informatics practices

a text book for class XII

Informatics Practices is a major subject at Senior Secondary level of CBSE with an aim of creating a league of students well-versed with computer practices that apply information science tools and methods in various fields. The subject 'Informatics Practices' focuses on concepts of Information Science applications and the basics of a Programming Language Python with an aim to use it for the management of information effectively.

Written strictly according to CBSE syllabus of subject code 065, this book covers :

- Data Handling
- Database Query using SQL
- Computer Networks
- Societal Impacts

This text book has been so designed that it contains adequate examples and practical questions so as to make the learning complete.

Keeping in sync with earlier editions, this book comes with a student-friendly presentation style so as to make learning process easier and fun to learn. This textbook – complete with numerous examples, OTQs (Objective Type Questions), solved problems, glossary and assignments – promises to guide you for the preparation for the *final examination*.

Publisher

Published by
DHANPAT RAI & CO. (P) LTD.
Educational & Technical Publishers
4576/15, Agarwal Road, Darya Ganj, New Delhi-110002
Ph.: 2324 7736, 37, 38 • E-mail: dhanpatrai@gmail.com

ISBN 978-81-7700-237-9

informatics practices XII

SUMITA ARORA

DHANPAT RAI

Sipo

Use SIPO app to scan the QR codes or to access the support material

informatics practices

Data Handling
Database Query using SQL
Computer Networks
Societal Impacts

SUMITA ARORA

a text book for class XII

DHANPAT RAI & Co.

Published by : GAGAN KAPUR
Dhanpat Rai & Co. (P) Ltd., Delhi

Sales Office : 1710, Nai Sarak, Delhi-110006
Phone : 2325 0251

Regd. Office : 4576/15, Agarwal Road
Darya Ganj, New Delhi-110002
Phone : 2324 7736, 37, 38, dhanpatrai@gmail.com

© Sumita Arora



All Trademarks Acknowledged

Disclaimer

Every effort has been made to avoid errors or omissions in this publication. In spite of this, some errors might have crept in. Any mistake, error or discrepancy noted may be brought to our notice which shall be taken care of in the next edition. It is notified that neither the publisher nor the author or seller will be responsible for any damage or loss of action to any one, of any kind, in any manner, therefrom.

EDITIONS : 2002, 2003, 2004, 2005, 2006, 2008, 2010, 2013, 2015, 2017, 2018
REPRINT : 2007, 2009, 2011, 2012, 2016
THIRTEENTH EDITION : 2020

PRICE : ₹ 495/-

Typesetting by : North Delhi Computer Convention, Delhi–110009,
ndcc.in@gmail.com
Printed at : Natraj Offset, Delhi

Preface

'Nothing is permanent but change'. Everyone agrees to this and this reflects in all fields around us. But the field of computers reflects this quote so prominently that every year we are surrounded by new technologies, updated technologies and what not. All this reflects in education too, especially the field of computer education. To keep abreast of the changing syllabi as per the changing technological requirements, we also have to come up with updated editions of the book every year and this is the reason behind this edition of the book.

This book adheres to the CBSE curriculumn for the subject 'Informatics Practices' (code 065) for Class XII. Based on the syllabus, this book has been divided into *four* units.

Unit 1 : Data Handling using Pandas and Data Visualization

The unit has been divided into *four* chapters (Chapters 1-4) and covers Python data handling and plotting libraries. This unit covers working with **Pandas** libraries of Python and **PyPlot** interface of Python's *matplotlib* library. In Pandas, it covers working with *two* basic data structures – **Series** and **Dataframes**. This unit also covers *importing/exporting of data from Pandas' dataframes to csv files and MySQL*.

Unit 2 : Database Query using SQL

This unit has been divided into *four* chapters (Chapters 5-8) and covers working with *MySQL databases and SQL queries. It covers simple SQL queries, MySQL functions, Joins and Set Operations.* The SELECT queries of SQL cover working with single-row functions, multiple rows' functions, grouping of records etc. The SQL also covers how to create various types of joins and perform various SET operations (UNION, INTERSECT, MINUS) on database tables.

Unit 3 : Introduction to Computer Networks

This unit has been divided into *two* chapters (Chapters 9-10) and covers *computers networks' basics,* along with the discussion on the *Internet and Web*. The computer networks chapter discusses various *network types, different network topologies*, factors for *topology selection* and various *network devices*. The *Internet and Web* chapter covers the distinct difference between the Internet and the Web, along with all things that make up the Internet and Web – *web servers, websites, web pages web browsers, Internet applications* etc.

Unit 4 : Societal Impacts

This unit has been divided into two chapters (Chapters 11-12) and covers *societal impacts of computers and technilogy, ethical issues, cyber crime, open source philosophy, health converns of technology use* and so on. It also covers various *threats to data and the solutions available for data protection.*

The SIPO app for Digital Content linked with this book



Dear Students

Get hold of Linked Learning Resources (*explanatory videos/ illustrations/images/text solutions*) for FREE, by scanning QR codes given in the chapters using our integrated learning app

Sipo

Linked resources include :

- explanatory videos
- animations
- illustrations
- images
- text solutions of selective assignment questions.

So, what are you waiting for ? Download SIPO now.

To
Kushagra and Mythili,
With Love

Syllabus

Distribution of Marks

Unit No.	Unit Name	Marks
1.	Data Handling using Pandas and Data Visualization	30
2.	Database Query using SQL	25
3.	Introduction to Computer Networks	07
4.	Societal Impacts	08
5.	Practicals	30
	Total	**100**

Unit 1 : Data Handling using Pandas and Data Visualization (50 Theory + 40 Practical)

Data Handling using Pandas – I

- Introduction to Python libraries – Pandas, Matplotlib.
- Data structures in Pandas – Series and Data Frames.
- Series : Creation of Series from – ndarray, dictionary, scalar value; mathematical operations ; Head and Tail functions ; Selection, Indexing and Slicing.
- Data Frames : Creation – from dictionary of Series, list of dictionaries, Text/CSV files ; display ; iteration ;
- Operations on rows and columns : add, select, delete, rename; Head and Tail functions ; Indexing using Labels, Boolean Indexing ; Joining, Merging and Concatenation.
- Importing/Exporting Data between CSV files and Data Frames.

Data Handling using Pandas – II

- Descriptive Statistics : max, min, count, sum, mean, median, mode, quartile, standard deviation, variance.
- DataFrame operations : Aggregation, group by, Sorting, Deleting and Renaming Index, Pivoting.
- Handling missing values – dropping and filling.
- Importing/Exporting Data between MySQL database and Pandas.

Data Visualization

- Purpose of plotting ; drawing and saving following types of plots using Matplotlib – line plot, bar graph, histogram, pie chart, frequency polygon, box plot and scatter plot.
- Customizing plots : color, style (dashed, dotted), width ; adding label, title, and legend in plots.

Unit 2: Database Query using SQL (30 Theory + 22 Practical)

- Math functions : POWER( ), ROUND( ), MOD( ).
- Text functions : UCASE( )/UPPER( ), LCASE( )/LOWER( ), MID( )/SUBSTRING( )/SUBSTR( ), LENGTH( ), LEFT( ), RIGHT( ), INSTR( ), LTRIM( ), RTRIM( ), TRIM( ).
- Date Functions : NOW( ), DATE( ), MONTH( ), MONTHNAME( ), YEAR( ), DAY( ), DAYNAME( ).
- Aggregate Functions : MAX( ), MIN( ), AVG( ), SUM( ), COUNT( ) ; using COUNT(*).
- Querying and manipulating data using Group by, Having, Order by.
- Operations on Relations – Union, Intersection, Minus, Cartesian Product, JOIN

Unit 3: Introduction to Computer Networks

(12 Theory + 2 Practical)

- Introduction to networks, Types of network : LAN, MAN, WAN.
- Network Devices : modem, hub, switch, repeater, router, gateway
- Network Topologies : Star, Bus, Tree, Mesh.
- Introduction to Internet, URL, WWW and its applications – Web, email, Chat, VoIP.
- Website : Introduction, difference between a website and webpage, static *vs.* dynamic web page, web server and hosting of a website.
- Web Browsers : Introduction, commonly used browsers, browser settings, add-ons and plug-ins, cookies.

Unit 4: Societal Impacts

(14 Theory)

- Digital footprint, net and communication etiquettes, data protection, intellectual property rights (IPR),
- plagiarism, licensing and copyright, free and open source software (FOSS), cybercrime and cyber laws, hacking, phishing, cyber bullying, overview of Indian IT Act.
- E-waste : hazards and management.
- Awareness about health concerns related to the usage of technology.

Project

The aim of the class project is to create tangible and useful IT application. The learner may identify a real-world problem by exploring the environment. e.g. Students can visit shops/business places, communities or other organizations in their localities and enquire about functioning of the organization, and how data are generated, stored and managed. The learner can take data stored in csv or database file and analyze using Python libraries and generate appropriate charts to visualize. If an organization is maintaining data offline, then the learner should create a database using MySQL and store the data in tables. Data can be imported in Pandas for analysis and visualization.

Learners can use Python libraries of their choice to develop software for their school or any other social good.

Learners should be sensitized to avoid plagiarism and violation of copyright issues while working on projects.

Teachers should take necessary measures for this. Any resources (data, image etc.) used in the project must be suitably referenced.

The project can be done individually or in groups of 2 to 3 students. The project should be started by students at least 6 months before the submission deadline.

Practical

Unit No.	Unit Name	Marks
1.	**Programs using Pandas and Matplotlib**	8
2.	SQL Queries	5
3.	Practical file (minimum of 20 programs based on Pandas, 5 based on Matplotlib and 20 SQL queries must be included)	5
4.	Project Work (using concepts learned in class XI and XII)	7
5.	Viva-Voce	5
	Total	30

Suggested Practical List

Data Handling

1. Create a pandas series from a dictionary of values and an ndarray
2. Given a Series, print all the elements that are above the 75th percentile.
3. Create a Data Frame quarterly sales where each row contains the item category, item name, and expenditure. Group the rows by the category, and print the total expenditure per category.
4. Create a data frame based on ecommerce data and generate descriptive statistics (mean, median, mode, quartile, and variance)
5. Create a data frame for examination result and display row labels, column labels data types of each column and the dimensions
6. Filter out rows based on different criteria such as duplicate rows.
7. Find the sum of each column, or find the column with the lowest mean.
8. Locate the 3 largest values in a data frame.
9. Subtract the mean of a row from each element of the row in a Data Frame.
10. Replace all negative values in a data frame with a 0.
11. Replace all missing values in a data frame with a 999.
12. Importing and exporting data between pandas and CSV file
13. Importing and exporting data between pandas and MySQL database

Visualization

14. Given the school result data, analyse the performance of the students on different parameters, *e.g.*, subject wise or class wise.
15. For the Data frames created above, analyze and plot appropriate charts with title and legend.
16. Take data of your interest from an open source (e.g. data.gov.in), aggregate and summarize it. Then plot it using different plotting functions of the Matplotlib library.

Data Management

17. Create a student table with the student id, name, and marks as attributes where the student id is the `primary key`.
18. Insert the details of a new student in the above table.
19. Delete the details of a particular student in the above table.
20. Use the select command to get the details of the students with marks more than 80.
21. Create a new table (order ID, customer Name, and order Date) by joining two tables (order ID, customer ID, and order Date) and (customer ID, customer Name, contact Name, country).
22. Create a foreign key in one of the two tables mentioned above
23. Find the min, max, sum, and average of the marks in a student marks table.
24. Find the total number of customers from each country in the table (customer ID, customer Name, country) using group by.
25. Create a new table (name, date of birth) by joining two tables (student id, name) and (student id, date of birth).
26. Write a SQL query to order the (student ID, marks) table in descending order of the marks.

Introduction to Computer Networks

27. Download, install and configure browser.

Contents

1 – 74

Python Pandas – I

1.1 Introduction 1
1.2 Using Pandas 1
1.3 Pandas Data Structures 2
1.4 Series Data Structure 4
1.4.1 Creating Series Objects 4
1.4.2 Creating Series Objects – Additional Functionality 8
1.4.3 Series Object Attributes 13
1.4.4 Accessing a Series Object and its Elements 16
1.4.5 Operations on Series Object 18
1.4.6 Difference between NumPy Arrays and Series Objects 26
1.5 DataFrame Data Structure 27
1.6 Creating and Displaying a DataFrame 29
1.7 DataFrame Attributes 38
1.8 Selecting or Accessing Data 41
1.8.1 Selecting/Accessing a Column 41
1.8.2 Selecting/Accessing Multiple Columns 41
1.8.3 Selecting/Accessing a Subset from a DataFrame using Row/Column Names 42
1.8.4 Selecting Rows/Columns from a DataFrame 44
1.8.5 Selecting/Accessing Individual Value 46
1.9 Adding/Modifying Rows'/Columns' Values in DataFrames 46
1.9.1 Adding/Modifying a Column 46
1.9.2 Adding/Modifying a Row 48
1.9.3 Modifying a Single Cell 49
1.10 Deleting/Renaming Columns/Rows 50
1.10.1 Deleting Rows/Columns in a DataFrame 50
1.10.2 Renaming Rows/Columns 51
1.11 More on DataFrame Indexing — Boolean Indexing 53
1.11.1 Creating Dataframes with Boolean Indexes 54
1.11.2 Accessing Rows from DataFrames with Boolean Indexes 55

75 – 160

Python Pandas – II

2.1 Introduction 75
2.2 Iterating Over a DataFrame 76
2.2.1 Using pandas.iterrows( ) Function 76
2.2.2 Using pandas.iteritems( ) Function 78
2.3 Binary Operations in a DataFrame 81
2.4 Descriptive Statistics with Pandas 86
2.4.1 Functions min( ) and max( ) 87
2.4.2 Functions mode( ), mean( ), median( ) 89
2.4.3 Functions count( ) and sum( ) 93

2.4.4 Functions quantile( ), std( ) and var( ) 94
2.4.5 The describe( ) Function 99
2.5 Some Other Essential Functions and Functionality 101
2.5.1 Inspection Function info( ) 101
2.5.2 DataFrame's Top and Bottom Rows using head( ) and tail( ) 102
2.5.3 Cumulative Calculations Functions 103
2.5.4 Applying Functions on a Subset of Dataframe 105
2.6 Advanced Operations on DataFrame 108
2.6.1 Pivoting 108
2.6.2 Sorting Dataframe Values 118
2.6.3 Aggregation 121
2.7 Handling Missing Data 122
2.7.1 Detecting/Filtering Missing Data 123
2.7.2 Handling Missing Data – Dropping Missing Values 124
2.7.3 Handling Missing Data – Filling Missing Values 124
Filling missing values from another DataFrame 126
2.8 Combining DataFrames 127
2.8.1 Combining Dataframes using concat( ) 127
2.8.2 Combining Dataframes using join( ) 128
2.8.3 Combining Dataframes using merge( ) 134
2.9 Function groupby( ) 137

161 – 244

Plotting with PyPlot

3.1 What is Data Visualization ? 161
3.2 Using Pyplot of Matplotlib Library 162
3.2.1 Installing and Importing matplotlib 162
3.2.2 Working with PyPlot Methods 163
3.3 Creating Line Charts and Scatter Charts 165
3.3.1 Line Chart using plot( ) Function 166
3.3.2 Specifying Plot Size and Grid 169
3.3.3 Applying Various Settings in plot( ) Function 169
3.3.4 Creating Scatter Charts 175
3.4 Creating Bar Charts and Pie Charts 181
3.4.1 Changing Widths of the Bars in a Bar Chart 184
3.4.2 Changing Colors of the Bars in a Bar Chart 186
3.4.3 Creating Multiple Bars Chart 188
3.4.4 Creating a Horizontal Bar Chart 191
3.4.5 Creating Pie Charts 191
3.5 Customizing the Plot 198
3.5.1 Anatomy of a Chart 198
3.5.2 Adding a Title 199
3.5.3 Setting X and Y Labels, Limits and Ticks 199
3.5.4 Adding Legends 205
3.5.5 Saving a Figure 207
3.6 Creating Histograms with PyPlot 208
3.7 Creating Frequency Polygons 213
3.8 Creating Box Plots 215

3.9 Plotting Data from a DataFrame 221
3.9.1 *Plotting a DataFrame's Data using PyPlot's Graph Functions* 221
3.9.2 *Plotting a DataFrame's Data using DataFrame's plot( )* 223

4 Importing/Exporting Data between CSV Files/MySQL and Pandas
245 – 277

4.1 Introduction 245
4.2 Transferring Data between .csv Files and DataFrames 246
4.2.1 *Loading Data from CSV to Dataframes* 246
4.2.2 *Storing Dataframe's Data to CSV File* 253
4.3 Transferring Data Between Dataframes and MySQL 257
4.3.1 *Bringing Data from MySQL Database into a Dataframe* 257
4.3.2 *Framing Flexible SQL Queries with User Data* 261
4.4 Exporting a Dataframe's Data as a Table in MySQL Database 264

5 MySQL SQL Revision Tour
279 – 303

5.1 Introduction 279
5.2 Relational Data Model 280
5.3 MySQL – A Popular Relational DataBase Management System 281
5.3.1 *MySQL and SQL* 281
5.3.2 *Common MySQL Data Types* 282
5.4 Accessing Database in MySQL 282
5.5 Creating Tables in MySQL 283
5.6 Inserting Data into Table 283
5.7 Making Simple Queries through Select Command 284
5.7.1 *Selecting All Data* 284
5.7.2 *Selecting Particular Rows* 284
5.7.3 *Selecting Particular Columns* 285
5.7.4 *Eliminating Redundant Data (with Keyword DISTINCT)* 285
5.7.5 *Selecting From All the Rows – ALL Keyword* 285
5.7.6 *Viewing Structure of a Table* 285
5.7.7 *Performing Simple Calculations* 286
5.7.8 *Using Column Aliases* 286
5.7.9 *Condition Based on a Range* 286
5.7.10 *Condition Based on a List* 286
5.7.11 *Condition Based on Pattern Matches* 286
5.7.12 *Searching for NULL* 287
5.8 Creating Tables with SQL Constraints 287
5.8.1 *SQL Constraints* 288
5.8.2 *Applying Table Constraints* 292
5.9 Viewing a Table Structure 292
5.10 Inserting Data Into another Table 293
5.11 Modifying Data in Tables 293
5.12 Deleting Data from Tables 293
5.13 Altering Tables 294
5.14 Dropping Tables 295

6 MySQL Functions
305 – 331

6.1 Introduction 305
6.2 Text Functions 306
6.3 Numeric Functions 315
6.4 Date/Time Functions 319

7 Querying Using SQL
333 – 362

7.1 Introduction 333
7.2 Ordering Records in Result – Order by Clause 333
7.2.1 *Recalling SQL SELECT ORDER BY Clause* 333
7.2.2 *Ordering Data on Multiple Columns* 335
7.2.3 *Ordering Data on the Basis of an Expression* 335
7.3 Aggregate Functions 337
7.4 Types of SQL Functions 340
7.5 Grouping Result – GROUP BY 340
7.5.1 *Nested Groups – Grouping on Multiple Columns* 341
7.5.2 *Placing Conditions on Groups – HAVING Clause* 343
7.5.3 *Non-Group Expressions with GROUP BY* 344

8 Joins and Set Operations
363 – 398

8.1 Introduction 363
8.2 Joining Tables 364
8.2.1 *Unrestricted Join (Cartesian Product)* 364
8.2.2 *Restricted Join (Join)* 364
8.2.3 *Using Table Aliases* 368
8.2.4 *Additional Search Conditions in Joins* 369
8.2.5 *Joining More Than Two Tables* 370
8.2.6 *Equi Join* 371
8.2.7 *Non-Equi-Joins* 371
8.2.8 *Natural Join* 371
8.2.9 *Joining Tables using JOIN Clauses of SQL SELECT* 372
8.3 Performing SET Operations on Relations 377
8.3.1 *UNION* 377
8.3.2 *MINUS Operator* 380
8.3.3 *INTERSECT Operator* 382

9 Introduction to Computer Networks
399 – 424

9.1 Introduction 399
9.2 Computer Networks – An Introduction 400
9.2.1 *Components of a Computer Network* 401
9.3 Types of Networks 402
9.3.1 *Types of Networks based on Geographical Spread* 402

9.4 Network Topologies 404
9.4.1 Point-to-Point Link 405
9.4.2 The Star Topology 405
9.4.3 The Bus or Linear Topology 406
9.4.4 The Ring or Circular Topology 407
9.4.5 The Tree Topology 408
9.4.6 Mesh Topology 408
9.4.7 Fully Connected 409
9.4.8 Factors to Consider for Topology Selection 409
9.5 Network Devices 410
9.5.1 Modem 410
9.5.2 Hub 410
9.5.3 Switch 411
9.5.4 Repeater 411
9.5.5 Bridge 411
9.5.6 Router 411
9.5.7 Gateway 412

10 425 – 454 Introduction to Internet and Web
10.1 Introduction 425
10.2 Introduction to Internet 425
10.3 World Wide Web (WWW) 426
10.3.1 History of World Wide Web 427
10.3.2 Difference between WWW and Internet 428
10.4 Web Servers 429
10.4.1 Services Provided by a Server 429
10.5 URLs 430
10.5.1 Elements of URL 430
10.5.2 URL Types 432
10.6 WebSite 432
10.6.1 Components of a Website 433
10.7 Web Pages 433
10.7.1 Components of a Webpage 434
10.7.2 Types of Web Page 435
10.7.3 Difference between a Webpage and Website 437
10.8 Web Browsers 438
10.8.1 Web Browser Toolbar 438
10.8.2 Browser Settings 439
10.9 Internet Applications 444
10.9.1 E-mail (Electronic Mail) 444
10.9.2 Protocols used in Email 446
10.10 Chat 447
10.10.1 Commonly used Type of Chat 447
10.11 Voice Over Internet Protocol (VoIP) 448

11 455 – 482 Societal Impacts
11.1 Introduction 455
11.2 Digital Footprint 456
11.2.1 Managing Digital Footprint 456
11.3 Net and Communications Etiquettes 457
11.3.1 Email Etiquettes 458
11.4 Ethical Essues 458
11.4.1 Intellectual Property Rights 459
11.4.2 Plagiarism 460
11.5 Open Source Philosophy 461
11.5.1 Terminology 461
11.5.2 Philosophy of Open Source 462
11.5.3 Definitions 463
11.6 Copyright and other Licenses 465
11.6.1 Licenses and Domains of Open Source Technology 465
11.7 Cyber Crime 467
11.8 Cyber Law and IT Act 470
11.9 E-Waste Management 471
11.9.1 E-Waste Disposal Process 472
11.9.2 Benefits of e-Waste Recycling 472
11.10 Health Concerns with Technology Usage 473

12 483 – 496 Data Protection
12.1 Introduction 483
12.2 Threats to Data 484
12.2.1 Computer Viruses 484
12.2.2 Spyware 485
12.2.3 Adware 485
12.2.4 Spamming 486
12.2.5 PC Intrusion 486
12.2.6 Eavesdropping 486
12.2.7 Phishing and Pharming 487
12.2.8 Cookies 487
12.3 Data Protection Solutions 488
12.3.1 Solutions to Viruses, Adware and Spyware 488
12.3.2 Solutions to Spam, Eavesdropping 489
12.3.3 Solutions to PC Intrusion 490
12.3.4 Solutions to Phishing and Pharming Attack 491
12.4 Firewall – An Important Solution for Computer Security 492

Appendix
Appendix A : Installing MySQL Python Connector 497
Appendix B : New Style : String Templates with { } formatting 498

Answers to Objective Type Questions (OTQs) (i) – (v)

LISTS OF QR CODES

Pandas' data structures in action 3
Series attributes in action 13
Vector operations in action 21
Arithmetic in series in action 21
DataFrame Anatomy in action 28
DataFrame Attributes in action 38
Subset from a dataframe in action 43
For selective assignment solutions 69
Working of iterrows in action 77
Working of iteritems in action 79
Operations on dataframe in action 81
min( ), max( ) functions in action 87
mode( ), mean( ), median( ) functions in action 89
count( ), sum( ) functions in action 93
quantile( ) function in action 95
cumprod( ), cummax( ), cummin( ) functions in action 104
pivot( ) function in action 110
pivot_table function in action 116
concat( ) in action 127
join( ) in action 129
merge( ) in action 134
For selective assignment solutions 156
Line, Scatter charts in action 165
marker types in action 173
scatter( ) function in action 179
Bar charts in action 183
Pie charts in action 195
Customizing charts in action 199
Histogram plot in action 209
boxplot( ) chart in action 218
For selective assignment solutions 242

LISTS OF QR CODES

Pandas' data structures in action 3
Series attributes in action 13
Vector operations in action 21
Arithmetic in series in action 21
DataFrame Anatomy in action 28
DataFrame Attributes in action 38
Subset from a dataframe in action 43
For selective assignment solutions 69
Working of iterrows in action 77
Working of iteritems in action 79
Operations on dataframe in action 81
min( ), max( ) functions in action 87
mode( ), mean( ), median( ) functions in action 89
count( ), sum( ) functions in action 93
quantile( ) function in action 95
cumprod( ), cummax( ), cummin( ) functions in action 104
pivot( ) function in action 110
pivot_table function in action 116
concat( ) in action 127
join( ) in action 129
merge( ) in action 134
For selective assignment solutions 156
Line, Scatter charts in action 165
marker types in action 173
scatter( ) function in action 179
Bar charts in action 183
Pie charts in action 195
Customizing charts in action 199
Histogram plot in action 209
boxplot( ) chart in action 218
For selective assignment solutions 242

1 Python Pandas - I

Outline

1.1 Introduction
1.2 Using Pandas
1.3 Pandas Data Structures
1.4 Series Data Structure
1.5 DataFrame Data Structure
1.6 Creating and Displaying a DataFrame
1.7 DataFrame Attributes
1.8 Selecting or Accessing Data
1.9 Adding/Modifying Rows'/Columns' Values in DataFrames
1.10 Deleting/Renaming Columns/Rows
1.11 More on DataFrame Indexing – Boolean Indexing

1.1 Introduction

Pandas or **Python Pandas** is Python's library for data analysis. **Pandas** has derived its name from **"panel data system"**, which is an ecometrics term for multi-dimensional, structured data sets. Today, Pandas has become a popular choice for data analysis. As you must be aware of that *data analysis* refers to process of evaluating big data sets using analytical and statistical tools so as to discover useful information and conclusions to support business decision-making.

Pandas makes available various tools for data analysis and makes it a simple and easy process as compared to other available tools. The main author of Pandas is **Wes McKinney**.

This chapter will introduce Python's Pandas library, its data structures *series* and *dataframes* and some useful functions with these data structures.

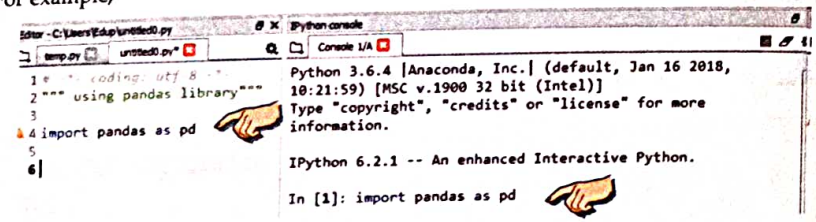
1.2 Using Pandas

Pandas is an open source, BSD library built for Python programming language. Pandas offers high-performance, easy-to-use data structures and data analysis tools.

In order to work with pandas in Python, you need to import **pandas** library in your python environment. You can do this in either on the shell prompt or in your script file (.py) by writing :

```
import pandas as pd
```

For example,



Please note that you can write code for data analysis using pandas library of Python in a .py file too as you have been writing programs earlier.

Why Pandas ?

Pandas is the most popular library in the *scientific Python ecosystem* for doing data analysis. Pandas is capable of many tasks including :

- It can read or write in many different data formats (*integer, float, double*, etc.).
- It can calculate in all the possible ways data is organized *i.e.*, across rows and down columns.
- It can easily select subsets of data from bulky data sets and even combine multiple datasets together. It has functionality to find and fill missing data.
- It allows you to apply operations to independent groups within the data.
- It supports reshaping of data into different forms.
- It supports advanced time-series functionality (*Time series forecasting is the use of a model to predict future values based on previously observed values.*)
- It supports visualization by integrating *matplotlib* and *seaborn* etc. libraries.

In other words, Pandas is best at handling huge tabular data sets comprising different data formats. Now that you know how to use Pandas on technical platform of your choice, let us learn the basics of *Python Pandas library*.

1.3 Pandas Data Structures

DATA STRUCTURES

> **Data Structures** refer to specialized way of storing data so as to apply a specific type of functionality on them.

In order to work with pandas, you need to learn the basic data structures of Pandas. But hey, do you know what the term ***data structure*** means ? What ? My mistake ? I did not tell you ? Hmm, ohh (Hope you will forgive me ;)) So here I am, correcting my mistake ☺.

A data structure is a particular way of storing and organizing data in a computer to suit a specific purpose so that it can be accessed and worked with in appropriate ways. Let us make it more clear. For instance, consider following cases :

- You want to store similar type of data items together and process them in identical way *e.g.*, you want to store marks scored by students – **arrays** are the solution for this. Or if you want to fix one end for insertions or deletions in an array, it will be termed as a **stack**-datastructure.
- There are many more other types of data structures suited for different types of functionality.

Depending upon the requirements of the situation, a data structure is decided for that situation.

Pandas offers many data structures to handle variety of data. At the very basic level, Pandas data structures can be thought of somewhat as enhanced versions of NumPy structured

arrays in which the *rows* and *columns* can be identified and accessed *with* ***labels*** *rather than simple integer indices.* You will learn about it in this chapter. Out of many data structures[1] of Pandas , two basic data structures – **Series** and **DataFrame** – are universally popular for their dependability.

Two Basic Data Structures Series and DataFrame

Series is 1-dimensional data structure of Python Pandas and **Dataframe** is a 2-dimesional data structure of Python Pandas. *Pandas* also supports another data structure called **Panel**, but that is beyond the scope of this book. Thus, our discussion will remain limited to Series and DataFrame data structures only. Following Fig. 1.1 shows a Series and a DataFrame object of Python Pandas.

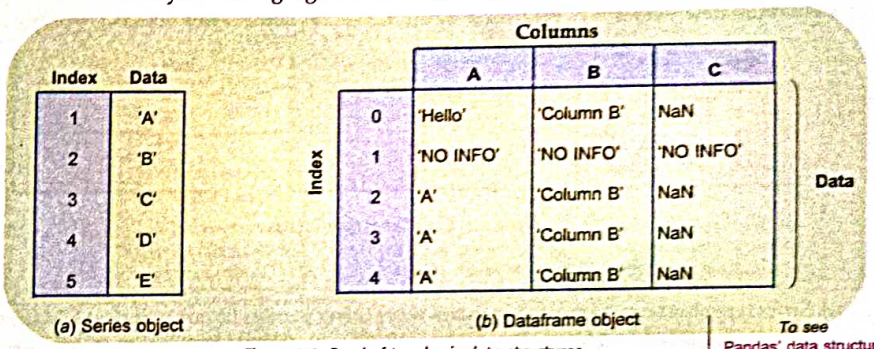


Figure 1.1 Pandas' two basic data structures

To see Pandas' data structures in action

Scan QR Code

Before we proceed to the detailed discussion of Pandas Series and Dataframe, let us talk about the basic difference between the two. Following table lists the key difference between a Series and DataFrame object.

Table 1.1 *Series vs. DataFrame objects*

Property	*Series*	*Dataframe*
Dimensions	1-dimensional	2-dimensional
Type of data	**Homogeneous**, *i.e.*, all the elements must be of same data type in a Series object.	**Heterogeneous**, *i.e.*, a DataFrame object can have elements of different data types.
Mutability	• **Value mutable**, *i.e.*, their elements' value can change. • **Size–immutable**, *i.e.*, **size** of a Series object, once created, **cannot change**. If you want to add/drop an element, internally a new Series object will be created.	• **Value mutable**, *i.e.*, their elements' value can change. • **Size–mutable**, *i.e.*, size of a Dataframe object, once created, can change in place. That is, you can add/drop elements in an existing dataframe object.

For working in Pandas, we generally import both *pandas* and *numpy* modules/libraries by giving following **import** statements on the Python shell or in the script or program code :

```
import pandas as pd
import numpy as np
```

You can use any identifier name in place of ***pd*** *and* ***np*** *but* ***pd*** *and* ***np*** *have been preferred choices generally*

After this basic introduction of Series and DataFrame objects, let us talk about these individually in details.

1. There are other data structures too such as **panel** but covering **panels** here is beyond the scope of the book.

1.4 Series Data Structure

Series is an important data structure of pandas. It represents a one-dimensional array of indexed data. A *Series type object* has *two* main components :

- an array of actual data
- an associated array of indexes or data labels.

SERIES

> "A Series is a Pandas data structure that represents a one-dimensional array-like object containing an array of data (of any NumPy data type) and an associated array of data labels, called its *Index*."

Both components are one-dimensional arrays with the same length. The index is used to access individual data values, *e.g.*, following figure 1.2 shows some series objects :

Index	Data
0	21
1	23
2	18
3	25

Index	Data
Jan	31
Feb	28
Mar	31
Apr	30

Index	Data
'A'	91
'B'	81
'C'	71
'D'	61

Figure 1.2 Some Series type objects

1.4.1 Creating Series Objects

A *Series type object* can be created in many ways using pandas library's **Series( )**. Make sure that you have imported **pandas** and **numpy** modules with **import** statements.

I. Create empty Series Object by using just the Series( ) with no parameter

To create an empty object *i.e.*, having no values, you can just use the Series( ) as :

```
<Series Object> = pandas.Series( )
```

S is in uppercase.

The above statement will create an *empty Series type object* with no value having default datatype which is ***float64***. Consider following statement :

```
In [94]: obj3 = pd.Series()

In [95]: obj3
Out[95]: Series([], dtype: float64)
```

Just Series() will create an empty Series type object

II. Creating non-empty Series objects

To create non-empty Series objects, you need to specify arguments for data and indexes as per following syntax :

```
<Series object> = pd.Series(data, index=idx)
```

where **idx** is a valid Numpy datatype and **data** is the data part of the Series object, it can be one of the following :

- A Python sequence
- An ndarray
- A Python dictionary
- A scalar value

Following subsections talk about the ways to create Series objects as per above syntax :

(i) Specify data as Python Sequence

Simplest way to create Series type object is to give a sequence of values as attribute to Series( ), *i.e.*, as :

```
<Series Object> = Series (<any Python sequence>)
```

It will return *an object of Series type*. For instance, consider following example statements that create two *Series type objects* using some Python sequences :

```
In [85]: obj1 = pd.Series(range(5))

In [86]: obj1
Out[86]:
0    0
1    1
2    2
3    3
4    4
dtype: int64
```

range(5) generates a sequence [0, 1, 2, 3, 4]

String representation of Series object **obj1** : *Left column* displays ***index*** and *right column* displays ***values***

```
In [87]: obj2 = pd.Series( [ 3.5, 5., 6.5, 8.] )

In [88]: obj2
Out[88]:
0    3.5
1    5.0
2    6.5
3    8.0
dtype: float64
```

Given list is a sequence with 4 values

String representation of Series object **obj2** : *Left column* displays ***index*** and *right column* displays ***values***

As you see above, if you just specify the sequence or just the values with Series( ) and no index, then by default index array consists of the integers *0 through* $N-1$ (where N is the length of data).

EXAMPLE 1 *Write code to create a Series object using the Python sequence [4, 6, 8, 10]. Assume that Pandas is imported as alias name pd.*

SOLUTION

```
import pandas as pd
s1 = pd.Series([4, 6, 8, 10])
print("Series object 1:")
print(s1)
```

Output

```
Series object 1:
0    4
1    6
2    8
3   10
dtype : int64
```

EXAMPLE 2 *Write code to create a Series object using the Python sequence (11, 21, 31, 41). Assume that Pandas is imported as alias name pd.*

SOLUTION

```
import pandas as pd
s2 = pd.Series((11, 21, 31, 41))
print("Series object 2:")
print(s2)
```

Output

```
Series object 2:
0    11
1    21
2    31
3    41
dtype: int64
```

EXAMPLE 3 *Write a program to create a Series object using individual characters 'o', 'h', and 'o'. Assume that Pandas is imported as alias name pd.*

SOLUTION

```
import pandas as pd
s3 = pd.Series(['o', 'h', 'o'])
print("Series object:")
print(s3)
```

Output

```
Series object:
0   o
1   h
2   o
dtype: object
```

EXAMPLE 4 *Write a program to create a Series object using a string : "So funny". Assume that Pandas is imported as alias name pd.*

SOLUTION

```
import pandas as pd
print("Series object:")
s4 = pd.Series("So funny")
print(s4)
```

Output

```
Series object:
0    So funny
dtype: object
```

EXAMPLE 5 *Write a program to create a Series object using three different words : "I", "am", "laughing". Assume that Pandas is imported as alias name pd.*

SOLUTION

```
import pandas as pd
s5 = pd.Series(["I", "am", "laughing"])
print("Series object:")
print(s5)
```

Output

```
Series object:
0          I
1         am
2   laughing
dtype: object
```

(ii) Specify data as an ndarray

The data attribute can be an ndarray also.

Consider following code :

```
nda1 = np.arange(3, 13, 3.5)
print(nda1)
ser1 = pd.Series(nda1)
print(ser1)
```

```
In [32]: nda1 = np.arange(3, 13, 3.5)

In [33]: print(nda1)
[ 3.   6.5 10. ]

In [34]: ser1 = pd.Series(nda1)

In [35]: ser1
Out[35]:
0     3.0
1     6.5
2    10.0
dtype: float64
```

You can also combine the code lines of above code. Carefully read the following :

```
ser1 = pd.series( np.arange(3, 13, 3.5) )
```

*This was stored as **nda1** in earlier code, but you can also create ndarray directly here too*

You can create a Series object from any ndarray created from any function. Consider following example that does the same.

EXAMPLE 6 *Write a program to create a Series object using an ndarray that has 5 elements in the range 24 to 64.*

SOLUTION

```
import pandas as pd
s6 = pd.Series (np.linspace(24, 64, 5))
print(s6)
```

Output

```
0    24.0
1    34.0
2    44.0
3    54.0
4    64.0
dtype: float64
```

EXAMPLE 7 *Write a program to create a Series object using an ndarray that is created by tiling a list [3,5] , twice).*

SOLUTION

```
import pandas as pd
s7 = pd.Series (np.tile([3,5],2))
print(s7)
```

Output

```
0    3
1    5
2    3
3    5
dtype: int32
```

(iii) Specify data as a Python Dictionary

The sequence that you provide with Series( ) can be any sequence, including ***dictionaries***. Let us see how you can create a Series object by specifying indexes and values through a dictionary. (see below)

```
In [25]: obj5 = pd.Series( { 'Jan' :31, 'Feb' : 28, 'Mar':31 } )

In [26]: obj5
Out[26]:
Feb    28
Jan    31
Mar    31
dtype: int64
```

See, left column took the keys of passed dictionary as ***indexes*** and values form the ***data values***

(Also, notice that indexes are not in the same order as given in the dictionary above)

See, Sequence passed to Series ( ) is a dictionary

Notice, curly brackets this time

Here, one thing is noteworthy that if you are creating a Series object from a dictionary object, then **keys** of the dictionary become **index** of the Series and the **values** of dictionary become the data of Series object. Also, the ***indexes*, which are created from *keys* may not be in the same order as you have typed them.**

EXAMPLE 8 *Write a program to create a Series object using a dictionary that stores the number of students in each section of class 12 in your school.*

SOLUTION

```
import pandas as pd
stu = { 'A':39, 'B':41, 'C':42, 'D': 44 }
s8 = pd.Series(stu)
print(s8)
```

Output

```
A    39
B    41
C    42
D    44
dtype: int64
```

(iv) Specify data as a Scalar Value

The **data** can be in the form of a single value or a scalar value. BUT if data is a scalar value, then the ***index*** argument to **Series( )** function must be provided. The scalar value (given as data) will be repeated to match the length of index. (The **index** argument has to be a sequence of numbers or labels of any type) *e.g.*, consider this :

NOTE

If data is a scalar value, then the ***index*** argument to Series( ) function must be provided.

This argument will specify the indexes for series object's elements.

```
medalsWon = pd.Series(10, index = range(0, 1))
medals2 = pd.Series(15, index = range(1, 6, 2))
ser2 = pd.Series('Yet to start', index = ['Indore', 'Delhi', 'Shimla'] )
```

See below the Series objects created as per above statements.

```
In [45]: medalsWon
Out[45]:
0    10
dtype: int64
```

```
In [46]: medals2
Out[46]:
1    15
3    15
5    15
dtype: int64
```

```
In [47]: ser2
Out[47]:
Indore    Yet to start
Delhi     Yet to start
Shimla    Yet to start
dtype: object
```

EXAMPLE **9** *Write a program to create a Series object that stores the initial budget allocated (50000/- each) for the four quarters of the year : Qtr1, Qtr2, Qtr3 and Qtr4.*

SOLUTION

```
import pandas as pd
s9 = pd.Series(50000, index = ['Qtr1', 'Qtr2', 'Qtr3', 'Qtr4'])
print(s9)
```

Output

```
Qtr1    50000
Qtr2    50000
Qtr3    50000
Qtr4    50000
dtype: int64
```

EXAMPLE **10** *Total number of medals to be won is 200 in the Inter University games held every alternate year. Write code to create a Series object that stores these medals for games to be held in the decade 2020-2029.*

SOLUTION

```
import pandas as pd
s10 = pd.Series(200, index = range(2020,2029,2))
print(s10)
```

Output

```
2020    200
2022    200
2024    200
2026    200
2028    200
dtype: int64
```

1.4.2 Creating Series Objects – Additional Functionality

Now that you have a fair idea about how to create Series type objects, let us talk about additional functionality of Series( ) that you can use to create Pandas Series objects.

(i) Specifying/Adding NaN values in a Series Object

Sometimes you need to create a series object of a certain size but you do not have complete data available at that time. In such cases, you can fill missing data with a NaN (Not a Number) value.

Legal empty value NaN is defined in *NumPy module* and hence you can use np.NaN to specify a missing value, or use *None e.g.*, see following code :

```
>>> obj3 = pd.Series( [ 6.5, np.NaN, 2.34] )
>>> obj3
0       6.50
1        NaN
2       2.34
dtype:  float64
```

You can specify missing or empty value using np.NaN

Use np.NaN or None to add missing data.

(ii) Specify index(es) as well as data with Series( )

While creating Series type object is that along with ***values***, you also provide ***indexes***. Both ***values*** and ***indexes*** are sequences.

The syntax for this is as follows :

These are default values, if you skip these parameters.

```
<Series Object> = pandas.Series(data = None, index = None)
```

Both ***data*** and ***index*** have to be sequences ; *None* is taken by default, if you skip these parameters.

NOTE

In place of **pandas.Series( )**, you may use **pd.Series( )** also if you have imported ***pandas*** as ***pd*** [*i.e.*, import pandas as pd] or with the name that you have used with import statement *e.g.*, if you have given statement as import pandas as pnd then you can use ***pnd.Series( )*** in place of ***pandas.Series( )***

Consider following example statements :

```
In [100]: arr = [31, 28, 31, 30]
In [101]: mon = ["Jan", "Feb", "Mar", "Apr"]
In [102]: obj3 = pd.Series(data = arr, index = mon)
In [103]: obj3
Out[103]:
Jan    31
Feb    28
Mar    31
Apr    30
dtype: int64
In [104]: obj4 = pd.Series( data = [32, 34, 35], index = ['A', 'B', 'C'] )
In [105]: obj4
Out[105]:
A    32
B    34
C    35
dtype: int64
```

A sequence of numbers

A sequence of strings

Series object created with both *values* and *indexes* given as sequences

Series object created with above statement

Another Series object created with both ***values*** and ***indexes*** given as sequences

datatype chosen by default

You could skip **keyword** data also, *i.e.*, following statement will also do the same as above :

```
obj3 = pd.Series(arr, index = mon)
```

You may use loop for defining index sequence also, *e.g.*,

```
s1 = pd.Series(range(1, 15, 3), index = [x for x in 'abcde'])
```

The above code will create a series object as shown below :

```
In [4]: s1 = pd.Series(range(1, 15, 3), index=[x for x in 'abcde'])

In [5]: s1
Out[5]:
a     1
b     4
c     7
d    10
e    13
dtype: int64
```

Loop used for specifying indexes

Caution !

If specifying indexes explicitly using an index sequence, you must provide indexes equal to the number of values in data array; providing too few or too many indices will lead to an error – the **ValueError**.

EXAMPLE **11** *A Python list namely* **section** *stores the section names ('A', 'B', 'C', 'D') of class 12 in your school. Another list* **contri** *stores the contribution made by these students to a charity fund endorsed by the school. Write code to create a Series object that stores the contribution amount as the values and the section names as the indexes.*

SOLUTION

```
import pandas as pd
section = ['A', 'B', 'C', 'D']
contri = [6700, 5600, 5000, 5200]
s11 = pd.Series(data = contri, index = section)
print(s11)
```

Output

```
A    6700
B    5600
C    5000
D    5200
dtype: int64
```

(iii) Specify Data Type along with data and index

You can also specify ***data type*** along with data and index with *Series( )* as per following syntax :

```
<Series Object> = pandas.Series(data = None, index = None, dtype = None)
```

None is the default value for different parameters taken in case no value is provided for a parameter. If you do not specify datatype, the nearest datatype to store the given values will be taken. But you can specify your own datatype by specifying a NumPy datatype with ***dtype*** attribute.

Consider following statements :

Series object's indexes are not necessarily to 0 to $n-1$ always.

```
In [106]: obj5 = pd.Series(data = arr, index = mon, dtype = np.float64)

In [107]: obj5
Out[107]:
Jan    31.0
Feb    28.0
Mar    31.0
Apr    30.0
dtype: float64
```

This time *dtype* provided. Compare it with obj3 object that is created above without specifying *dtype*

(iv) Using a Mathematical Function/Expression to Create Data Array in Series( )

The *Series( )* allows you to define a function or expression that can calculate values for data sequence. It is done in the following form :

```
<Series Object> = pandas.Series(index = None, data =<function|expression> )
```

To understand its functioning, consider following example statements.

```
In [126]: a = np.arange(9, 13)

In [127]: print(a)
[ 9 10 11 12]

In [128]: obj7 = pd.Series(index =a, data = a * 2)

In [129]: obj7
Out[129]:
9     18
10    20
11    22
12    24
dtype: int32

In [130]: obj8 = pd.Series(index = a, data = a ** 2)

In [131]: obj8
Out[131]:
9      81
10    100
11    121
12    144
dtype: int32
```

NumPy array *a* created as it supports vectorized operations

Data sequence given as a * 2 i.e., every value of a doubled

See, the index values taken from NumPy array a

Data sequence given as a **2 i.e., every value of a squared

The vectorized operations on a NumPy array (*e.g.*, a * 2 or a **2) will be applied on every element of NumPy array and stored as data part of ***Series object***. Now consider another code :

```
In [11]: Lst = [9, 10, 11, 12]

In [12]: obj8 = pd.Series( data = (2 * Lst) )

In [13]: obj8
Out[13]:
0     9
1    10
2    11
3    12
4     9
5    10
6    11
7    12
dtype: int64
```

Notice this time the expression for data array involves a list and see how it has impacted the final data array

data = (2 * Lst)

The above code tries to create data array for *Series object* **obj8** by giving expression 2 * *Lst* where **Lst** is a Python list and as you know a number multiplied with a list replicates the list those many times and hence the data array has the same list values replicated twice (∵ **2 * Lst**).

Carefully read the following example. It is **similar yet different** from the previous example.

EXAMPLE **12** *Sequences* **section** *and* **contri1** *store the section names ('A', 'B', 'C', 'D', 'E') and contribution made by them respectively (6700, 5600, 5000, 5200, nil) for a charity. Your school has decided to donate as much contribution as made by each section, i.e., the donation will be doubled.*

Write code to create a Series object that stores the contribution amount as the values and the section names as the indexes with datatype as ***float32****.*

SOLUTION

```
import pandas as pd
import numpy as np
section = ['A', 'B', 'C', 'D', 'E']
contri1 = np.array([6700, 5600, 5000, 5200, np.NaN])
s12= pd.Series(data = contri1 * 2, index = section, dtype = np.float32)
print(s12)
```

This time we have stored the contribution amount in an ndarray **contri1** *so that we can double its value through vectorized operation to add school's contribution*

This will double each value of the **contri1** *ndarray. This is done to reflect the school's contribution to charity.*

Output

```
A    13400.0
B    11200.0
C    10000.0
D    10400.0
E        NaN
dtype: float32
```

Compare the data type of this Series object with the previous object (s11). It is now different from default data type,

NOTE

When you store a NaN value in a series object, Pandas requires the data type to be of floating point type. Even if you specify an integer type, Pandas will promote it to a floating point type (automatically) because NaN is not supported by integer types.

IMPORTANT

While creating a Series object, when you give index array as a sequence then there is no compulsion for the uniqueness of indexes. That is, you can have duplicate entries in the index array and Python won't raise any error, *e.g.*, see figure below :

```
In [12]: val = np.arange(2.75, 50, 9.75)

In [13]: val
Out[13]: array([ 2.75, 12.5 , 22.25, 32.  , 41.75])

In [14]: ob3 = pd.Series (val, index = ['a', 'b', 'a', 'a', 'b' ] )

In [15]: ob3
Out[15]:
a     2.75
b    12.50
a    22.25
a    32.00
b    41.75
dtype: float64
```

See, index array has duplicate entries

No error ! Series object successfully converted

NOTE

Indices need not be unique in Pandas Series Object. This will only cause an error if/when you perform an operation that requires unique indices.

1.4.3 Series Object Attributes

When you create a Series type object, all information related to it is available through attributes. You can use these attributes in the following format to get information about the Series object.

```
<Series object>.<attribute name>
```

Some common attributes of Series object are listed in the table below. The code examples for the usage of these attributes follow the Table 1.2.

To see series attributes in action

Scan QR Code

Table 1.2 *Common attributes of Series objects*

Attribute	Description
<Series object>.index	The index (axis labels) of the Series.
<Series object>.values	Return Series as ndarray or ndarray-like depending on the dtype
<Series object>.dtype	return the dtype object of the underlying data
<Series object>.shape	return a tuple of the shape of the underlying data
<Series object>.nbytes	return the number of bytes in the underlying data
<Series object>.ndim	return the number of dimensions of the underlying data
<Series object>.size	return the number of elements in the underlying data
<Series object>.itemsize	return the size of the dtype of the item of the underlying data
<Series object>.hasnans	return True if there are any NaN values ; otherwise return False
<Series object>.empty	return True if the Series object is empty, false otherwise

Following examples show how to view various attributes of Series objects.

(a) Retrieving Index Array (index attribute) & Data Array (values attribute) of a Series Object

You can access the **index array** and **data values' array** of an existing Series object **obj5** as shown below :

```
In [51]: obj5.index
Out[51]: Index(['Feb', 'Jan', 'Mar'], dtype='object')

In [52]: obj5.values
Out[52]: array([28, 31, 31], dtype=int64)

In [53]: obj6.index
Out[53]: RangeIndex(start=0, stop=5, step=1)

In [54]: obj7.index
Out[54]: Int64Index([9, 10, 11, 12], dtype='int64')
```

See, <object>.index and <object>.values returned the index array and data values' array respectively

obj6 was created as :
obj6 = pd.Series(data = np.arange(11, 25, 3))

obj7 was created as :
obj7 = pd.Series(index = a, data = a * 2)
where a is a NumPy array ([9,10,11,12])

For the rest of the attributes (covered in points below), we shall be using following two objects (given as Reference 1.2)

```
>>>obj2 = pd.Series( [3.5, 5., 6.5, 8.] )
>>>obj2
0  3.5
1  5.0
2  6.5
3  8.0
dtype: f1  it64
```

```
>>>obj3 = pd.Series( [6.5, np.NaN, 2.34] )
>>>obj3
0  6.50
1   NaN
2  2.34
dtype: float64
```

Reference 1.2

(b) Retrieving Data Type (dtype) and Size of Type (itemsize)

To retrieve the data type of individual elements of a series object, use <objectname>.dtype.

To know about the type of *Series object* itself, you can use **type( )** of Python. You can use **itemsize** attribute to know the number of bytes allocated to each data item *e.g.*,

```
In [29]: obj2.dtype
Out[29]: dtype('float64')

In [30]: obj2.itemsize
Out[30]: 8

In [31]: type(obj2)
Out[31]: pandas.core.series.Series
```

The **dtype** attribute return the type of data values stored in Series object

The **itemsize** returns the size in bytes for each item

The **type()**, however, tells the type of the object passed to it

(c) Retrieving Shape

The *shape of a Series object (shape attribute) tells how big it is, i.e., how many elements it contains including missing or empty values (NaNs).* Since there is only one axis in Series object, it is shown as (<n>,) where *n* is the number of elements in the object *e.g.*,

NOTE

The shape of a series object tell how big it is, *i.e.*, how many elements it contains including missing or empty values (NaNs).

```
>>> print(obj2.shape , obj3.shape)
(4,) (3,)
```

See the number of elements in objects obj2 and obj3 are shown.

(d) Retrieving Dimension (number of axis : ndim attribute), Size (size attribute) and Number of Bytes (nbytes attribute)

To know about the dimension (number of axis), use <objectname>.ndim.

To know about the number of elements in the Series object, use <objectname>.size.

To know total number of bytes taken by Series object data, use <objectname>.nbytes (*nbytes* is equal to the **size * itemsize**)

```
>>> obj2.ndim
1
```

Series object is 1-dimesinal object.

```
>>> print (obj2.size, obj3.size)
4 3
```

obj2 has 4 elements and obj3 has 3 elements

```
>>> print (obj2.nbytes, obj3.nbytes)
32 24
```

*obj2 has 4 elements, hence 4*8 = 32 bytes and obj3 has 3 elements, hence 3 * 8 = 24 bytes*

(e) Checking Emptiness (empty attribute) and Presence of NaNs (hasnans attribute)

See we have created an empty Series object namely **obj1** also and then checked emptiness of **obj1** along with emptiness of objects **obj2** and **obj3** of Reference 1.2. (given on previous page).

```
In [37]: obj1=pd.Series()

In [38]: obj1.empty
Out[38]: True
```

The object **obj1** is empty but **obj2** and **obj3** (ref 10.2) are not empty as they contain some data

```
In [39]: obj2.empty
Out[39]: False

In [40]: obj3.empty
Out[40]: False
```


- Similarly, to check if a Series object contains some NaN value or not, you can use **hasnans** attribute as shown below (using objects of *reference 2.2*)
- You can use **len( )** to get total number of elements and **<series>.count( )** method with Series object to get the count of non-NaN values in a series object, *e.g.*, :

NOTE

If you use **len( )** on a Series object, then it returns total elements in it including NaNs but **<series>.count( )** returns only the count of non-NaN values in a Series object.

```
In [41]: obj2.hasnans
Out[41]: False

In [42]: obj3.hasnans
Out[42]: True
```

hasnans attrin\bute tells the presence of NaN values and count() returns count of non_NaN values

```
In [43]: obj2.count()
Out[43]: 4

In [44]: obj3.count()
Out[44]: 2
```

```
In [11]: len(obj3)
Out[11]: 3
```

len() gives total no. of elements

EXAMPLE 13 *Consider the two series objects **s11** and **s12** that you created in examples 11 and 12 respectively. Print the attributes of both these objects in a report form as shown below :*

Attribute name	Object s11	Object s12
Data type		
Shape		
No. of bytes		
No. of dimensions		
Item size		
Has NaNs ?		
Empty ?		

SOLUTION

```
import pandas as pd
:   # statements here to create objects s11 and s12 from previous examples
print("Attribute name  \t\t Object s11 \t Object s12 ")
print("————————        \t \t————— \t ——— ")
print("Data type(.dtype)        :\t", s11.dtype, '\t\t', s12.dtype)
print("Shape   (.shape)         :\t", s11.shape, '\t\t', s12.shape)
print("No. of bytes (.nbytes)   :\t", s11.nbytes, '\t\t', s12.nbytes)
print("No. of dimensions(.ndim) :\t", s11.ndim, '\t\t', s12.ndim)
print("Item size (.itemsize)    :\t", s11.itemsize, '\t\t', s12.itemsize)
print("Has NaNs? (.hasnans)     :\t", s11.hasnans, '\t\t', s12.hasnans)
print("Empty?  (.empty)         :\t", s11.empty, '\t\t', s12.empty)
```

Output

```
Attribute name                  Object s11      Object s12
————————
Data type(.dtype)          :    int64           float32
Shape      (.shape)        :    (4,)            (5,)
No. of bytes (.nbytes)     :    32              20
No. of dimensions(.ndim)   :    1               1
Item size (.itemsize)      :    8               4
Has Nan ? (.hasnans)       :    False           True
Empty?     (.empty)        :    False           False
```

1.4.4 Accessing a Series Object and its Elements

Once you have created *Series type object*, you can access it in many ways. You can access its indexes separately, its data separately and even access individual elements and slices. Let us see how. Consider following Series objects (**obj5, obj6, obj7** and **obj8**) (Fig. 1.3) :

```
In [35]: obj5
Out[35]:
Feb    28
Jan    31
Mar    31
dtype: int64
```

```
In [37]: obj6
Out[37]:
0    11
1    14
2    17
3    20
4    23
dtype: int32
```

```
In [28]: obj7
Out[28]:
9     18
10    20
11    22
12    24
dtype: int32
```

```
In [29]: obj8
Out[29]:
9      81
10    100
11    121
12    144
dtype: int32
```

Figure 1.3 Some sample panda Series Objects.

For all the following operations we shall be using the sample objects shown in Fig. 1.3.

1. Accessing Individual Elements

To access individual elements of a Series object, you can give its index in square brackets along with its name, *i.e.*, as :

```
<Series Object name>[<valid index>]
```

Following figure shows you elements accessed from the objects ***obj5, obj6, obj7*** and ***obj8.***

```
In [39]: obj6[3]
Out[39]: 20
```

```
In [32]: obj7[9]
Out[32]: 18
```

```
In [33]: obj8[11]
Out[33]: 121
```

```
In [38]: obj5['Feb']
Out[38]: 28
```

See all these objects' legal indexes are used to access individual elements of these objects.

As you see in above figure, we have use only valid or legal indexes (*i.e.*, which exist in series object) to access an element.

If the Series object has duplicate indexes, then giving an index with the Series object will return all the entries with that index, *e.g.*, see below :

```
In [15]: ob3
Out[15]:
a     2.75
b    12.50
a    22.25
a    32.00
b    41.75
dtype: float64
```

```
In [16]: ob3['b']
Out[16]:
b    12.50
b    41.75
dtype: float64
```

With duplicate indexes in a Series object, all entries with the same index are returned.

BUT if you try to give an index which is not a legal index for a **Series object**, it will give you an error.

See adjacent figure.

```
In [32]: obj7[2]
Traceback (most recent call last):

  File "<ipython-input-31-67505774af56>", line 1, in <module>
    obj7[2]

  File "pandas/_libs/hashtable_class_helper.pxi", line 817, in
pandas._libs.hashtable.Int64HashTable.get_item

KeyError: 2
```

2 is not a legal index for obj7 (its legal indexes are 9 to 12 refer above), hence the error.

2. Extracting Slices from Series Object

Like other sequences, you can extract slice too from a Series object to retrieve subsets. Let us see how you can extract slices from Series objects. Here, you need to understand **an important thing about slicing**, which is that :

Slicing takes place position wise and not the index wise in a series object.

To understand this, let us consider the same *Series objects* as given in Fig. 1.3. Internally there is a position associated with element – first element gets the position as 0, second element gets the position as 1 and so on. Irrespective of their indexes, positions always start with 0 and go on like 1, 2, 3 and so on. (*see* Fig. 1.4)

obj5

Position	Index	Data
0	Feb	28
1	Jan	31
2	Mar	31

obj6

Position	Index	Data
0	0	11
1	1	14
2	2	17
3	3	20
4	4	23

obj7

Position	Index	Data
0	9	18
1	10	20
2	11	22
3	12	24

All individual elements have position numbers starting from 0 onwards i.e., *0* for first element, *1* for 2nd element and so on

Figure 1.4 Position number associated with each element of Series object

When you have to extract slices, then you need to specify slices as [start : end : step] like you do for other sequences, but the ***start*** and ***stop*** signify the positions of elements not the indexes. Consider following examples :

```
In [67]: obj5[1: ]
Out[67]:
Jan    31
Mar    31
dtype: int64
```

```
In [68]: obj6[2:5]
Out[68]:
2    17
3    20
4    23
dtype: int32
```

```
In [70]: obj7[0: : 2]
Out[70]:
9     18
11    22
dtype: int32
```

Irrespective of the indexes, the slices have been extracted *position wise.*

All other rules of slices apply, *i.e.*, you can specify steps, you can reverse the elements, the range of slice can be outside the range of positions etc.

NOTE

A slice object is created from Series object using a syntax of <Object>[start : end : step], but the ***start*** and ***stop*** signify the positions of elements not the indexes. The **slice object** of a Series object is also a **panda Series type object.**

Even though **obj7** has indexes 10, 11, 12 but the slice object is empty, because slice is extracted *position wise* not *index wise*

```
In [71]: obj7[10:15]
Out[71]: Series([], dtype: int32)
```

```
In [72]: obj7[: : -1 ]
Out[72]:
12    24
11    22
10    20
9     18
dtype: int32
```

Slice object with values reversed

EXAMPLE **14** *Consider a Series object s8 that stores the number of students in each section of class 12 (as shown below).*

A	39
B	41
C	42
D	44

First two sections have been given a task of selling tickets @ 100/- per ticket as part of a social experiment. Write code to display how much they have collected.

Output

```
Tickets amount:
A    3900
B    4100
dtype: int64
```

SOLUTION

```
import pandas as pd
print("Tickets amount:")
print(s8[ :2] *100)
```

1.4.5 Operations on Series Object

Let us now talk about how you can perform various types of operations on Pandas Series objects.

1. Modifying Elements of Series Object

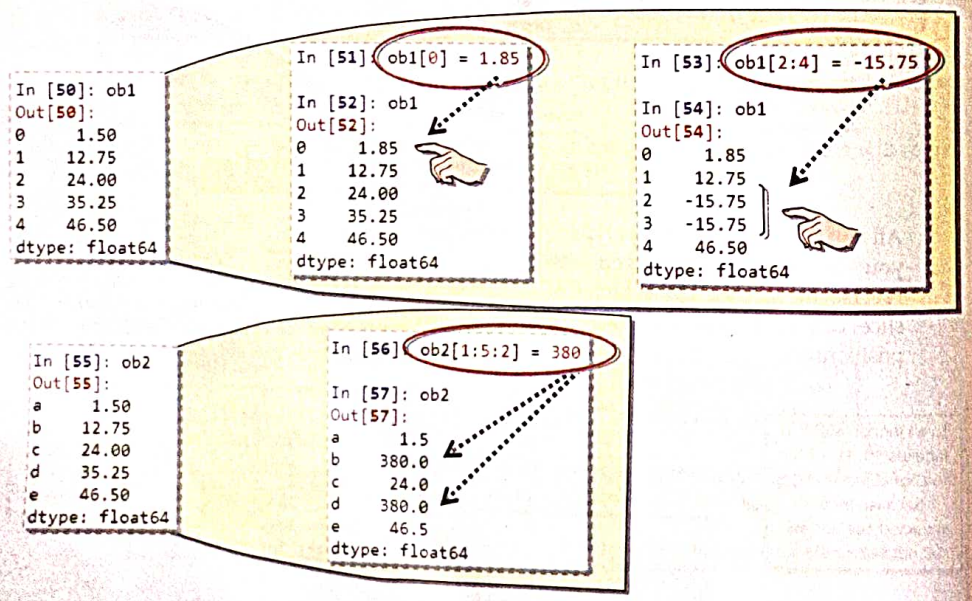
The data values of a Series object can be easily modified through item assignment, *i.e.,*

```
<SeriesObject>[<index> ] = <new_data_value>
```

Above assignment will change the data value of the given index in the Series object.

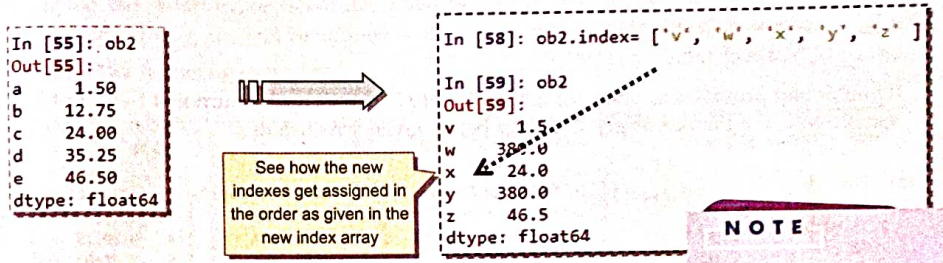
```
<SeriesObject>[start : stop] = <new_data_value>
```

Above assignment will replace all the values falling in given slice. Consider following screenshots :

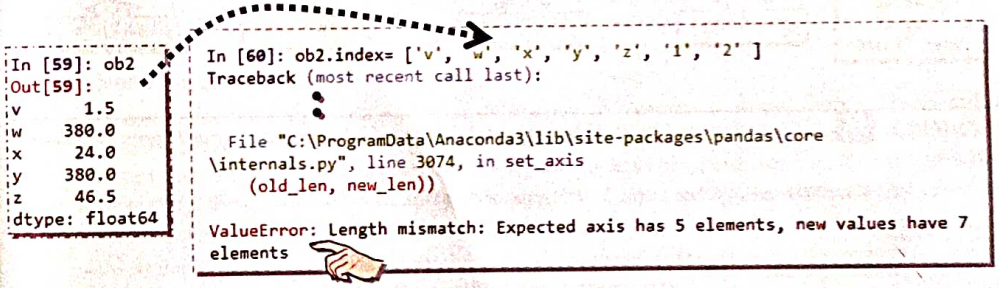


Renaming Indexes

You can even change or rename indexes of a Series object by assigning new index array to its index attribute, *i.e.,* : <Object>.index = <new index array>, *e.g.,* see below :



The size of new index array must match with existing index array's size. In other words, you cannot change the size of a Series object by assigning more or less number of indexes (see below) :



NOTE

Please note that Series object's values can be modified but size cannot. So you can say that **Series objects are *value-mutable* but *size-immutable* objects.**

EXAMPLE **15** *Consider the Series object s13 that stores the contribution of each section, as shown below :*

A	6700
B	5600
C	5000
D	5200

Write code to modify the amount of section 'A' as 7600 and for sections 'C' and 'D' as 7000. Print the changed object.

Output

```
Series object after modifying amounts:
A 7600
B 5600
C 7000
D 7000
```

SOLUTION

```
import pandas as pd
s13[0] = 7600                    # to modify section 'A''s amount
s13[2: ] = 7000                  # to modify sections 'C' and 'D''s amount
print("Series object after modifying amounts:")
print(s13)
```

2. The head( ) and tail( ) Functions

The ***head()*** function is used to fetch first *n* rows from a Pandas object and ***tail()*** function returns last *n* rows from a Pandas object. The syntax to use these functions is :

```
<pandas object>.head([n])
Or    <pandas object>.tail([n])
```

If you do not provide any value for *n*, then ***head()*** and ***tail()*** will return first 5 and last 5 rows respectively of a Pandas object. Consider below given screenshots :

```
In [42]: ob7
Out[42]:
0     2.75
1     7.60
2    12.45
3    17.30
4    22.15
5    27.00
6    31.85
7    36.70
8    41.55
9    46.40
10   51.25
dtype: float64
```

```
In [44]: ob7.head()
Out[44]:
0    2.75
1    7.60
2   12.45
3   17.30
4   22.15
dtype: float64
```

First 5 rows

```
In [45]: ob7.head(7)
Out[45]:
0    2.75
1    7.60
2   12.45
3   17.30
4   22.15
5   27.00
6   31.85
dtype: float64
```

First 7 rows

```
In [46]: ob7.tail()
Out[46]:
6    31.85
7    36.70
8    41.55
9    46.40
10   51.25
dtype: float64
```

Last 5 rows

```
In [47]: ob7.tail(7)
Out[47]:
4    22.15
5    27.00
6    31.85
7    36.70
8    41.55
9    46.40
10   51.25
dtype: float64
```

Last 7 rows

EXAMPLE 16 *A Series object* ***trdata*** *consists of around 2500 rows of data. Write a program to print the following details :*

(i) First 100 rows of data (ii) Last 5 rows of data

SOLUTION

```
import pandas as pd
:            # trdata object's creation or loading happens here
print(trdata.head(100))
print(trdata.tail())
```

3. Vector Operations on Series Object

Vector operations mean that if you apply a function or expression then it is individually applied on each item of the object. Since **Series objects** are built upon NumPy arrays (***ndarrays***), they also support vectorized operations, just like ***ndarrays***.

Following examples will make it clear to you.

Suppose we have a Pandas Series object **ob2** as shown here :

```
In [67]: ob2
Out[67]:
a     1.50
b    12.75
c    24.00
d    35.25
e    46.50
dtype: float64
```

A sample Series object

Then following all are legal operations :

```
ob2 + 2,   ob2 * 3,   ob8 = ob2 **2,   ob2 > 15   etc.
```

In all the above expressions, the given operation will be carried out in vectorized way, *i.e.*, will be applied to each item of the Series object.

Consider following examples :

To see vector operations in action

Scan QR Code

```
In [68]: ob2 + 2
Out[68]:
a     3.50
b    14.75
c    26.00
d    37.25
e    48.50
dtype: float64
```

```
In [69]: ob2 * 3
Out[69]:
a     4.50
b    38.25
c    72.00
d   105.75
e   139.50
dtype: float64
```

```
In [72]: ob2 > 15
Out[72]:
a    False
b    False
c     True
d     True
e     True
dtype: bool
```

```
In [73]: ob8 = ob2 ** 2

In [74]: ob8
Out[74]:
a       2.2500
b     162.5625
c     576.0000
d    1242.5625
e    2162.2500
dtype: float64
```

See each of the expression though applied on the Series type object but is carried out on each individual item of the Series object – **Vector Operations.**

Figure 1.5 Vector operations on Series objects

4. Arithmetic on Series Objects

You can perform arithmetic like *addition, subtraction, division* etc. with two Series objects and it will calculate result on two corresponding items of the two objects given in expression BUT it has a caveat – **the operation is performed only on the matching indexes**, *e.g.*, if first object has indexes 0,1,2 then it will perform arithmetic only with objects having 0,1,2 indexes ; for all other indexes, it will produce NaN (not a number).

Also, if the data items of the two matching indexes are not compatible for the operation, it will return NaN (***Not a Number***) as the result of those operations.

To understand this, consider below given (Fig. 1.6) *five* Series objects **ob1, ob2, ob3, ob4** and **ob5** (**ob1** and **ob3** have matching indexes ; **ob2** and **ob5** have matching indexes; **ob4** has some indexes matching with **ob1** and **ob3**) :

```
In [88]: ob1
Out[88]:
0     1.85
1    12.75
2   -15.75
3   -15.75
4    46.50
dtype: float64
```

```
In [75]: ob2
Out[75]:
a     1.50
b    12.75
c    24.00
d    35.25
e    46.50
dtype: float64
```

```
In [91]: ob3
Out[91]:
0     2.75
1    12.50
2    22.25
3    32.00
4    41.75
dtype: float64
```

```
In [98]: ob4
Out[98]:
0     1.255
1     5.530
2     9.805
3    14.080
4    18.355
5    22.630
6    26.905
7    31.180
dtype: float64
```

```
In [101]: ob5
Out[101]:
a     1.255
b     5.530
c     9.805
d    14.080
e    18.355
dtype: float64
```

Figure 1.6 (*a*) Some sample Series objects with matching and non-matching indexes

To see arithmetic in series in action

Scan QR Code

Now carefully look at the statements carrying out arithmetic operations on objects with matching indexes (see below) :

```
In [102]: ob1 * ob3
Out[102]:
0      5.0875
1    159.3750
2   -350.4375
3   -504.0000
4   1941.3750
dtype: float64
```

```
In [103]: ob1 / ob3
Out[103]:
0    0.672727
1    1.020000
2   -0.707865
3   -0.492188
4    1.113772
dtype: float64
```

```
In [104]: ob2 + ob5
Out[104]:
a     2.755
b    18.280
c    33.805
d    49.330
e    64.855
dtype: float64
```

```
In [104]: ob2 + ob5
Out[104]:
a     2.755
b    18.280
c    33.805
d    49.330
e    64.855
dtype: float64
```

Since objects **ob1** and **ob3** have matching indexes (both have indexes in the range 0 to 4), it successfully carries out given arithmetic operation on corresponding items of matching indexes, *i.e.*, items with index 0 of both the objects are performed the given operation and result given for the index 0, similarly corresponding items having index 1 of both the objects are performed the given operation and result given for index 1, and so on.

Same thing is applied for expression **ob2 + ob5**, *i.e.*, corresponding values of index 'a' are added, similarly corresponding values of index 'a' are added, and so on.

But if you try to perform operation on objects that have some or all non-matching indexes, then it will add values of matching indexes, if any and for non-matching indexes of both the objects, it will return the result as *Not a Number i.e.,* NaN. **NaN represents missing data.** (see below)

```
In [105]: ob1 + ob4
Out[105]:
0     3.105
1    18.280
2    -5.945
3    -1.670
4    64.855
5       NaN
6       NaN
7       NaN
dtype: float64
```

Computed the given operation for matching indexes (**0, 1, 2, 3, 4** in both objects) and **NaN** for non-matching indexes (**5, 6, 7** of *ob4*)

NaN (*Not a Number*) represents missing data

```
In [92]: ob1 +ob2
Out[92]:
0   NaN
1   NaN
2   NaN
3   NaN
4   NaN
a   NaN
b   NaN
c   NaN
d   NaN
e   NaN
dtype: float64
```

None of the indexes matched, because **(0, 1, 2, 3, 4)** of *ob1* do not match with indexes ('a', 'b', 'c', 'd', 'e') of *ob2*, hence **NaN** for all the indexes of both the objects

Figure 1.6 (*b*)

NOTE

When you perform arithmetic operations on two **Series** type objects , the data is aligned on the basis of matching indexes (this is called **Data Alignment in Pandas objects**) and then performed arithmetic; for non-overlapping indexes, the arithmetic operations result as a NaN (Not a Number).

You can store the result of object arithmetic in another object, which will also be a Series object, *i.e.,* if you give :

```
>>> ob6 = ob1 + ob3
```

Then **ob6** will also be a Series object (if **ob1** and **ob3** are Pandas Series objects).

EXAMPLE 17 *Number of students in classes 11 and 12 in three streams ('Science', 'Commerce' and 'Humanities') are stored in two Series objects c11 and 12. Write code to find total number of students in classes 11 and 12 , stream wise.*

SOLUTION

```
import pandas as pd
: # creating Series objects
c11 = pd.Series(data = [30, 40, 50], index = ['Science','Commerce','Humanities'])
c12 = pd.Series(data = [37, 44, 45], index = ['Science','Commerce','Humanities'])
# adding two objects to get total no. of students
print("Total no. of students")
print(c11+c12) # series objects arithmetic
```

Output

```
Total no. of students
Science      67
Commerce     84
Humanities   95
dtype: int64
```

EXAMPLE 18 *Object1* ***Population*** *stores the details of population in four metro cities of India and Object2* ***AvgIncome*** *stores the total average income reported in previous year in each of these metros. Calculate income per capita for each of these metro cities.*

SOLUTION

Statement continuation mark. Do not type it while typing code in a .py file, rather type the whole statement in single line

```
import pandas as pd
Population = pd.Series([10927986, 12691836, 4631392, 4328063 ], \
index = ['Delhi', 'Mumbai', 'Kolkata', 'Chennai'])
AvgIncome = pd.Series([72167810927986, 85087812691836, 4226784631392, 5261784328063 ], \
index = ['Delhi', 'Mumbai', 'Kolkata', 'Chennai'])
perCapita = AvgIncome / Population
print("Population in four metro cities ")
print(Population)
print("Avg. Income in four metro cities")
print(AvgIncome)
print("Per Capita Income in four metro cities ")
print(perCapita)
```

Expression to calculate per capita income

The output produced by above file is as shown on the right.

See how easy it has become to calculate per capita income if we have huge data stored about all the cities and average income etc.

```
In [113]: runfile('E:/Informatics Practi
perCapita.py', wdir='E:/Informatics Prac
Population in four metro cities
Delhi      10927986
Mumbai     12691836
Kolkata     4631392
Chennai     4328063
dtype: int64
Avg. Income in four metro cities
Delhi      72167810927986
Mumbai     85087812691836
Kolkata     4226784631392
Chennai     5261784328063
dtype: int64
Per Capita Income in four metro cities
Delhi      6.603944e+06
Mumbai     6.704137e+06
Kolkata    9.126381e+05
Chennai    1.215737e+06
dtype: float64
```

5. Filtering Entries

You can filter out entries from a Series object using expressions that are of Boolean type, (*i.e.*, the expressions that yield a Boolean value) as per following syntax :

```
<Series Object>[ [<Boolean expression on Series Object>]
```

Consider following examples (which are based on objects **ob1**, **ob2** and **ob3** from Fig. 1.6) :

Applying comparison operator directly on Series object works in vectorized way i.e., applies this check on each element and then returns True / False for each element

```
In [20]: ob1>5
Out[20]:
0    False
1     True
2    False
3    False
4     True
dtype: bool
```
(a)

```
In [21]: ob1[ob1>5]
Out[21]:
1    12.75
4    46.50
dtype: float64
```
(b)

```
In [23]: ob2>10
Out[23]:
a    False
b     True
c     True
d     True
e     True
dtype: bool
```
(c)

```
In [24]: ob2[ob2>10]
Out[24]:
b    12.75
c    24.00
d    35.25
e    46.50
dtype: float64
```
(d)

*But when this check is applied in the form <Series object> [<Boolean expression>], then it returns **filtered result** containing only the elements that return True for the given Boolean expression.*

When you apply a comparison operator directly on a Pandas Series object, then it works like vectorized operation and applies this check on each individual element of Series object (as you can see in (*a*) and (*c*) above) BUT when you apply this check with the Series object inside [ ] as per syntax given above then, you will find that it returns filtered result containing only the values that return ***True*** for the given Boolean expression (as you can see in (*b*) and (*d*) above).

EXAMPLE **19** *What will be the output produced by the following program ?*

```
import pandas as pd
info = pd.Series(data = [31, 41, 51])
print(info)
print(info > 40)
print(info[info > 40])
```

SOLUTION

```
0    31
1    41
2    51
dtype: int64
```
← *Series object*

```
0    False
1     True
2     True
dtype: bool
```
← *Vectorized operation result*

```
1    41
2    51
dtype: int64
```
← *Filtered result*

EXAMPLE **20** *Series object s11 stores the charity contribution made by each section (see below) :*

```
A    6700
B    5600
C    5000
D    5200
```

Write a program to display which sections made a contribution more than ₹5500/-

SOLUTION

```
import pandas as pd
:              # s11 created or loaded here
print("Contribution > 5500 by :")
print(s11 [s11 > 5500])
```

Output

```
Contribution > 5500 by :
A    6700
B    5600
dtype: int64
```

6. Sorting Series Values

You can sort the values of a Series object on the basis of values and indexes.

Sorting on the Basis of Values

To sort a Series object on the basis of values, you may use **sort_values( )** function as per the following index :

```
<Series object>.sort_values([ascending =True|False])
```
← *Optional argument*

The argument **ascending** is optional and if skipped, it takes the value ***True*** by default.

It means, the **sort_values( )** arranges the values in a Series object in ascending order by default.

For example consider the Series object **s11**,

```
>>> s11
A  6700
B  5600
C  5000
D  5200
dtype: int64
```

```
>>> s11.sort_values()
C  5000
D  5200
B  5600
A  6700
dtype: int64
```

Values sorted in ascending order (default setting)

```
>>> s11.sort_values(ascending = False)
A  6700
B  5600
D  5200
C  5000
dtype: int64
```

This time the values are sorted in descending order

To sort values in descending order, you may write **<series>.sort_values(ascending = False)**

Sorting on the Basis of Indexes

To sort a Series object on the basis of indexes, you may use **sort_index( )** function as per the following index :

Optional argument

```
<Series object>.sort_index([ascending = True|False])
```

The argument **ascending** is optional and if skipped, it takes the value *True* by default.

For example :

```
>>> s11.sort_index(ascending = False)
D  5200
C  5000
B  5600
A  6700
dtype: int64
```

Series sorted in descending order of indexes

```
>>> s11.sort_index()
A  6700
B  5600
C  5000
D  5200
dtype: int64
```

Series sorted in ascending order of indexes

Solved problems 13, 14 are based on sorting a Series object.

1.4.6 Difference between NumPy Arrays and Series Objects

The major differences in *ndarrays* and *Series objects* are listed below :

(*i*) In case of ndarrays, you can perform vectorized operations only if the shapes of two ndarrays match, otherwise it returns an error. But with Series objects, in case of vectorized operations, the data of two *Series objects* is aligned as per matching indexes and operation is performed on them and for non-matching indexes, NaN is returned. You have already read about this functionality in *section 1.4.5, point 4.*

(*ii*) In ndarrays, the indexes are always numeric starting from 0 onwards, BUT Series objects can have any type of indexes, including numbers (not necessarily starting from 0), letters, labels, strings etc.

```
In [88]: a1
Out[88]: array([1, 2, 3, 4])

In [89]: a3
Out[89]: array([11, 12, 13, 14, 15, 16, 17])

In [90]: a1+a3
Traceback (most recent call last):

  File "<ipython-input-90-f3952d2e37d2>", line 1, in <module>
    a1+a3

ValueError: operands could not be broadcast together with shapes (4,) (7,)
```

Ndarrays cannot perform vectorized operations on arrays with different shapes

Some Additional Operations on Series Objects

Reindexing

Sometimes you need to create a similar object but with a **different order of same** indexes. You can use reindexing for this purpose as per this syntax :

```
<Series Object > = <Object>.reindex(<sequence with new order of indexes>)
```

With this, the same datavalues and their indexes will be stored in the new object as per the defined order of index in the **reindex( )**. See below :

```
In [60]: ob2
Out[60]:
a     1.50
b    12.75
c    24.00
d    35.25
e    46.50
dtype: float64

In [61]: obj1 = ob2.reindex(['e', 'd', 'c', 'b', 'a'])

In [62]: obj1
Out[62]:
e    46.50
d    35.25
c    24.00
b    12.75
a     1.50
dtype: float64
```

Order of indexes is now changed

Dropping Entries from an Axis

Sometime, you do not need a data value at a particular index. You can remove that entry from series object using drop( ) as per this syntax :

```
<Series Object>.drop(<index to be removed>)
```

For example, see below :

```
In [60]: ob2
Out[60]:
a     1.50
b    12.75
c    24.00
d    35.25
e    46.50
dtype: float64

In [63]: obj2 = ob2.drop('c')

In [64]: obj2
Out[64]:
a     1.50
b    12.75
d    35.25
e    46.50
dtype: float64
```

Index 'c' dropped

1.5 DataFrame Data Structure

A **DataFrame** in another Pandas structure, which stores data in two-dimensional way. It is actually a **two-dimensional** (tabular and spreadsheet like) **labeled array**, which is actually an **ordered collection of columns** where columns may store different types of data, *e.g., numeric* or *string* or *floating point* or *Boolean* type etc.

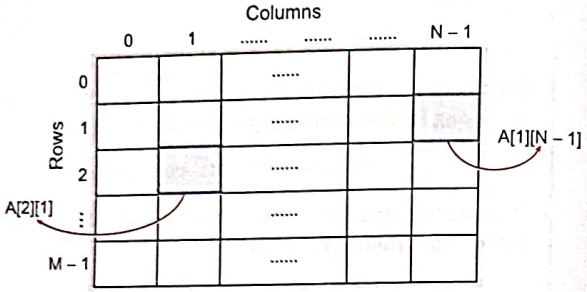
Since DataFrame data structure is like a two-dimensional array, let us first understand what a two-dimensional array is like.

DATAFRAME

> A **DataFrame** is a two-dimensional labeled array like Pandas data structure that stores an ordered collection columns that can store data of different types.

A two-dimensional array is an array in which each element is itself an array. For instance, an array *A* [m] [n] is an *M* by *N* table with *M* rows and *N* columns containing $M \times N$ elements.

The number of elements in a 2-D array can be determined by multiplying number of rows with number of columns. For example, the number of elements in an array A[7] [9] is calculated as $7 \times 9 = 63$.



Characteristics

Major **characteristics of a DataFrame data structure** can be listed as :

(*i*) It has two indexes or we can say that **two axes – a *row index* (*axis = 0*)** and a ***column index (axis = 1)***. [Fig. 1.7(*b*)]

(*ii*) Conceptually it is like a spreadsheet where each value is identifiable with the combination of *row index* and *column index*. The ***row index*** is known as index in general and the ***column index*** is called the column-name. [Fig. 1.7(*b*)]

(*iii*) The indexes can be of numbers or letters or strings. [Fig. 1.7(*a*)]

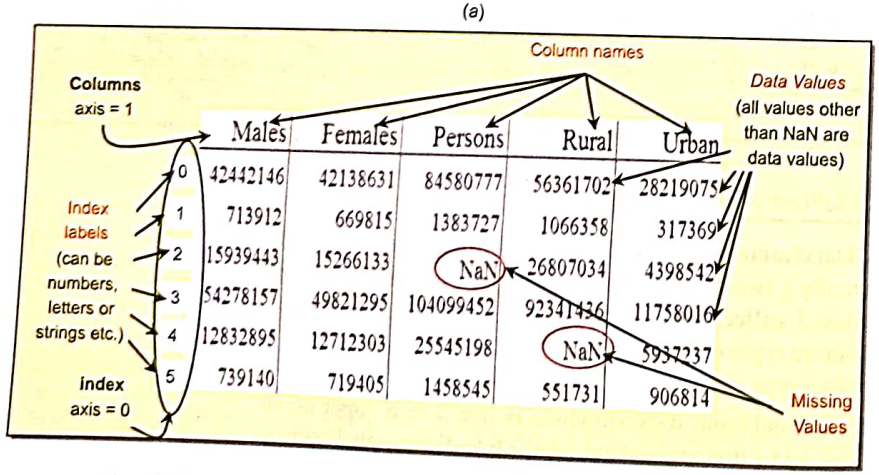
To see DataFrame Anatomy in action

Scan QR Code

	Statistic	N	Mean	St. Dev.	Min	Max
0	rating	30	64.633	12.173	40	85
1	complaints	30	66.600	13.315	37	90
2	privileges	30	53.133	12.235	30	83
3	learning	30	56.367	11.737	34	75
4	raises	30	64.633	10.397	43	88

(*a*)

	A	B	C	D	E	F
a	0.0	0.0	0.0	0.0	0.0	0.0
b	0.0	0.0	0.0	0.0	0.0	0.0
c	0.0	0.0	0.0	0.0	0.0	0.0
d	0.0	0.0	0.0	0.0	0.0	0.0
e	0.0	0.0	0.0	0.0	0.0	0.0



(*b*)

Figure 1.7 (*a*) Some sample DataFrame objects (*b*) Anatomy of a DataFrame object.

(*iv*) There is no condition of having all data of same type across columns; its columns can have data of different types. [Fig. 1.7(*a*)]

(*v*) You can easily change its values, *i.e.*, it is **value-mutable.**

(*vi*) You can add or delete rows/columns in a DataFrame. In other words, it is **size-mutable.**

NOTE

DataFrames are both, **value-mutable** and **size-mutable**, *i.e.*, you can change both its values and size.

Now that you have an idea about DataFrames, you can understand the terms associated with DataFrame as explained in following figure [Fig. 1.7(*b*)].

1.6 Creating and Displaying a DataFrame

A DataFrame object can be created by passing data in two-dimensional format. Like earlier, before you do anything with pandas module, make sure to import Pandas and NumPy modules, *i.e.*, give the following two import statements on your code or Python console :

```
import pandas as pd
import numpy as np
```

To create a DataFrame object, you can use syntax as :

*Both **D** and **F** are capital letters*

```
<datFrameObject> = panda.DataFrame(<a 2D datastructure>,\
[columns = <column sequence> ], [index = <index sequence>])
```

Command continuation mark

where the 2D data structure passed to it, contains the data values. If you have imported Pandas as **pd**, then you can create a DataFrame as **pd.DataFrame( )** also.

You can create a DataFrame object by passing data in many different ways, such as :

(*i*) Two-dimensional dictionaries *i.e.*, dictionaries having lists or dictionaries or *ndarrays* or Series objects etc.

(*ii*) Two-dimensional *ndarrays* (NumPy array)

(*iii*) Series type object

(*iv*) Another DataFrame object

1. Creating a DataFrame Object from a 2-D Dictionary

A two dimensional dictionary is a dictionary having items as (key : value) where value part is a data structure of any type : another dictionary, an ndarray, a Series object, a list etc. But here the value parts of all the keys should have similar structure and equal lengths.

Following sections explain these with the help of many examples.

(a) Creating a dataframe from a 2D dictionary having values as lists/ndarrays :

You can have a two-dimensional dictionary wherein the value part consists of either *lists* or *ndarrays*. Passing such a 2D dictionary to *DataFrame( )* will create a dataframe object as you can see yourself in the example below :

```
>>> dict1 = {'Students' : ['Ruchika', 'Neha', 'Mark', 'Gurjyot', 'Jamaal'],
'Marks' : [ 79.5, 83.75, 74, 88.5, 89 ],
'Sport' : ['Cricket', 'Badminton', 'Football', 'Athletics', 'Kabaddi'] }
>>> dict1
{'Marks': [79.5, 83.75, 74, 88.5, 89],
'Sport': ['Cricket', 'Badminton', 'Football', 'Athletics', 'Kabaddi'],
'Students': ['Ruchika', 'Neha', 'Mark', 'Gurjyot', 'Jamaal'] }
>>> dtf1 = pd.DataFrame(dict1)
>>> dtf1
```

	Marks	Sport	Students
0	79.50	Cricket	Ruchika
1	83.75	Badminton	Neha
2	74.00	Football	Mark
3	88.50	Athletics	Gurjyot
4	89.00	Kabaddi	Jamaal

The created DataFrame object has its index assigned automatically (0 onwards), and the columns created from keys, are placed in sorted order.

indexes are automatically generated

As you can see that the created DataFrame object has its index assigned automatically (0 onwards) just as it happens with Series objects, and the **columns are placed in sorted order.**

You can specify your own indexes too by specifying a sequence by the name **index** in the DataFrame( ) function, *e.g.*,

```
>>> dtf2 = pd.DataFrame(dict1, index = ['I', 'II', 'III', 'IV', 'V'])
>>> dtf2
```

*Specify the new indexes by giving **index sequence***

	Marks	Sport	Students
I	79.50	Cricket	Ruchika
II	83.75	Badminton	Neha
III	74.00	Football	Mark
IV	88.50	Athletics	Gurjyot
V	89.00	Kabaddi	Jamaal

See, the indexes are now as per the index sequence

If you specify the *index sequence*, Python will take indexes from the **index** sequence, BUT the number of indexes given in the index sequence MUST MATCH the length of dictionary's values, otherwise Python will give error (*see* below) :

```
In [32]: dtf2 = pd.DataFrame(dict1, index =['I', 'II', 'III', 'IV', 'V', 'VI'])
Traceback (most recent call last):
  .
  .
  .
  File "C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals.py", line
4608, in construction_error
    passed, implied))

ValueError: Shape of passed values is (3, 5), indices imply (3, 6)
```


EXAMPLE **21** *Given a dictionary that stores the section names' list as value for 'Section' key and contribution amounts' list as value for 'Contri' key :*

```
dict1 = {'Section':[ 'A', 'B', 'C', 'D'],
         'Contri': [ 6700, 5600, 5000, 5200] }
```

Write code to create and display the data frame using above dictionary.

SOLUTION

```
import pandas as pd
dict1 = {'Section':[ 'A', 'B', 'C', 'D'],
         'Contri':[ 6700, 5600, 5000, 5200] }
df1 = pd.DataFrame(dict1)
print(df1)
```

Output

	Contri	Section
0	6700	A
1	5600	B
2	5000	C
3	5200	D

(b) Creating a dataframe from a 2D dictionary having values as dictionary objects :

A 2D dictionary can have values as dictionary objects too. You can also create a data frame object using such 2D dictionary object, (*e.g., see* below) :

Given a 2D dictionary namely **people** :

```
people = {'Sales' : {'name': 'Rohit', 'age': '24', 'sex': 'Male' },
          'Marketing' : {'name': 'Neha', 'age': '25', 'sex': 'Female'}}
```

Here, the keys of inner dictionaries are exactly the same (*'name'*, *'age'*, and *'sex'*).

You can create a DataFrame by passing this dictionary :

```
>>> pd.DataFrame(people)
```

Inner dictionary keys as indexes

Out dictionary keys as columns

	Marketing	Sales
age	25	24
name	Neha	Rohit
sex	Female	Male

This time the keys of inner dictionaries make the indexes and the keys of outer dictionary make the columns.

EXAMPLE **22** *Create and display a DataFrame from a 2D dictionary, **Sales**, which stores the quarter-wise sales as inner dictionary for two years, as shown below :*

```
Sales = {'yr1' : { 'Qtr1' : 34500, 'Qtr2' : 56000, 'Qtr3' : 47000, 'Qtr4' : 49000 } ,
         'yr2' : { 'Qtr1' : 44900, 'Qtr2' : 46100, 'Qtr3' : 57000, 'Qtr4' : 59000 } }
```

SOLUTION

```
import pandas as pd
Sales = { 'yr1' : { 'Qtr1' : 34500, 'Qtr2' : 56000, 'Qtr3' : 47000, 'Qtr4' : 49000 },
          'yr2' : { 'Qtr1' : 44900, 'Qtr2' : 46100, 'Qtr3' : 57000, 'Qtr4' : 59000 } }
dfsales = pd.DataFrame(Sales)
print(dfsales)
```

Output

	yr1	yr2
Qtr1	34500	44900
Qtr2	56000	46100
Qtr3	47000	57000
Qtr4	49000	59000

In above example notice one thing : as the keys of all inner dictionaries (*yr1, yr2*) are exactly the same in number and names, the dataframe object **dfsales** also has the same number of indexes.

Now, had there been a situation **where inner dictionaries had non-matching keys**, then in that case Python would have done following things :

(*i*) There would have been **total number of indexes equal to sum of unique inner keys** in all the inner dictionaries.

(*ii*) For a key that has no matching keys in other inner dictionaries , value **NaN** would be used to depict the missing values.

```
>>> Collect1 = { 'yr1' :1500, 'yr2':2500}
>>> Collect2 = {'yr1':2200, 'Nil': 0}
```

Two dictionaries with dissimilar keys

```
>>> Collect = { 'I': Collect1, 'II':Collect2}
```

Dictionaries with dissimilar keys as inner dictionaries of a 2D dictionary

```
>>> df = pd.DataFrame(Collect)
```

DataFrame created with non-matching inner key dictionaries

```
>>> df
        I       II
Nil     NaN     0.0
yr1     1500.0  2200.0
yr2     2500.0  NaN
```

All the inner keys become indexes

NaN values added for non-matching keys of inner dictionaries

EXAMPLE 23 *Carefully read the following code.*

```
import pandas as pd
yr1 = { 'Qtr1' : 44900, 'Qtr2' : 46100, 'Q3' : 57000, 'Q4' : 59000 }
yr2 = { 'A' : 54500, 'B' : 51000, 'Qtr4' : 57000 }
diSales1 = { 1 : yr1 , 2 : yr2}
df3 = pd.DataFrame(diSales1)
```

(*i*) *list the index labels of the DataFrame df3*

(*ii*) *list the column names of DataFrame df3.*

SOLUTION

(*i*) The index labels of *df3* will include : A, B, Q3, Q4, Qtr1, Qtr2, Qtr4.
The total number of indexes is equal to total unique inner keys, *i.e.*, 7.

(*ii*) The column names of *df3* will be : 1, 2

NOTE

Total number of indexes in a DataFrame object are equal to total unique inner keys of the 2D dictionary passed to it and it would use NaN values to fill missing data *i.e.*, where the corresponding values for a key are missing in any inner dictionary.

2. Creating a DataFrame Object from a List of Dictionaries/Lists

If you pass a 2D list having dictionaries as its elements (**list of dictionaries**) to **pandas.DataFrame( )** function, it will create a DataFrame object such that the inner dictionary keys will become the columns and inner dictionary's values will make rows. *For example,*

```
>>> import pandas as pd
>>> topperA = { 'Rollno':115, 'Name': 'Pavni', 'Marks': 97.5}
>>> topperB = { 'Rollno':236, 'Name': 'Rishi', 'Marks': 98}
>>> topperC = { 'Rollno':307, 'Name' : 'Preet', 'Marks' : 98.5}
>>> topperD = { 'Rollno':422, 'Name': 'Paula', 'Marks': 98}
>>> toppers = [ topperA, topperB, topperC, topperD]
```

This is a list of dictionaries

```
>>> topDf = pd.DataFrame(toppers)
>>> topDf
```

	Rollno	Name	Marks
0	115	Pavni	97.5
1	236	Rishi	98.0
2	307	Preet	98.5
3	422	Paula	98.0

See, the keys of inner dictionaries have made the columns of the DataFrame.

See, the values of inner dictionaries have made the rows of the DataFrame.

When a DataFrame is created with a list of dictionaries, the columns are created from the keys of dictionary elements and default indexes are assigned to rows (0, 1, 2, 3 ....) as you can see above. You can specify your own index labels through the **index** argument containing the row/index labels (see below).

```
>>> topDf = pd.DataFrame(toppers, index = ["Sec A","Sec B","Sec C","Sec D"] )
>>> topDf
```

	Rollno	Name	Marks
Sec A	115	Pavni	97.5
Sec B	236	Rishi	98.0
Sec C	307	Preet	98.5
Sec D	422	Paula	98.0

See, the row labels are created from the sequence given with index argument of DataFrame() function.

EXAMPLE 24 *Write a program to create a dataframe from a list containing dictionaries of the sales performance of four zonal offices. Zone names should be the row labels.*

SOLUTION

```
import pandas as pd
zoneA = {'Target':56000, 'Sales':58000}
zoneB = {'Target':70000, 'Sales':68000}
zoneC = {'Target':75000, 'Sales':78000}
zoneD = {'Target':60000, 'Sales':61000}
sales = [zoneA, zoneB, zoneC, zoneD]
saleDf = pd.DataFrame(sales , index = ['zoneA', 'zoneB', 'zoneC', 'zoneD'])
print( saleDf )
```

Output

	Target	Sales
zoneA	56000	58000
zoneB	70000	68000
zoneC	75000	78000
zoneD	60000	61000

Passing a 2D list, *i.e.*, a **list having lists** will also create a dataframe where each inner list will form the row of the dataframe, *e.g.*,

```
>>> list2 = [ [ 25, 45, 60], [34, 67, 89], [88, 90, 56] ]
>>> df1 = pd.DataFrame( list2 )
>>> df1
    0   1   2
0   25  45  60
1   34  67  89
2   88  90  56
```

See, the inner lists have formed a row each, in the created dataframe.

With 2D lists, the created dataframe by default names the columns and indexes as 0, 1, 2... unless you specify the index and columns arguments. You can specify own index names/labels with **index** argument and column labels with **columns** argument in the DataFrame( ) as illustrated in following two examples.

EXAMPLE **25** *Write a program to create a dataframe from a 2D list. Specify own index labels.*

SOLUTION

```
import pandas as pd
list2 = [ [ 25, 45, 60], [34, 67, 89], [88, 90, 56] ]
df2 = pd.DataFrame(list2, index = ['row1', 'row2', 'row3'] )
print ( df2 )
```

Output

	0	1	2
row1	25	45	60
row2	34	67	89
row3	88	90	56

EXAMPLE **26** *Write a program to create a dataframe from a list containing 2 lists, each containing Target and actual Sales figures of four zonal offices. Give appropriate row labels.*

SOLUTION

```
import pandas as pd
Target = [56000, 70000, 75000, 60000 ]
Sales = [58000, 68000, 78000, 61000]
ZoneSales = [Target, Sales]
zsaleDf = pd.DataFrame(ZoneSales, columns = ['ZoneA','ZoneB','ZoneC','ZoneD'],
                                  index = ['Target', 'Sales'])
print( zsaleDf )
```

Output

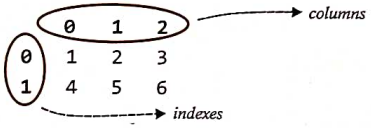
	ZoneA	ZoneB	ZoneC	ZoneD
Target	56000	70000	75000	60000
Sales	58000	68000	78000	61000

3. Creating a DataFrame Object from a 2-D ndarray

You can also pass a two-dimensional NumPy array (*i.e.*, having ***shape*** as (<*n*>, <*n*>)) to DataFrame( ) to create a dataframe object.

Let us see how :

```
>>> narr1 = np.array([[1, 2, 3], [4, 5, 6]], np.int32)
>>> narr1.shape
(2, 3)
>>> dtf1 = pd.DataFrame(narr1)
>>> dtf1
```



As you can see in above code example that Python created **indexes** or **axis = 0** from the first dimension of the passed ndarray and **columns (axis = 1)** from the second dimension of the passed ndarray. As no **keys** are there, hence default names are given to indexes and columns, *i.e.*, 0 onwards.

You can however, specify your own column names and /or index names by giving a **columns** sequence and/or index sequence :

```
>>> dtf2 = pd.DataFrame(narr1, columns = ['One', 'Two', 'Three'])
>>>dtf2
```

By giving a **columns** sequence, you can specify your own column names.

	One	Two	Three
0	1	2	3
1	4	5	6

See, this time the columns have the names given in columns sequence

```
>>> narr2 = np.array([[11.5, 21.2, 33.8], [40, 50, 60], [212.3, 301.5, 405.2]] )
>>> dtf3 = pd.DataFrame(narr2, columns = ['First', 'Second', 'Third'], index = ['A', 'B', 'C'])
>>> dtf3
```

By giving an **index** sequence, you can specify your own index names or labels .

You can specify either columns or index or both the sequences

	First	Second	Third
A	11.5	21.2	33.8
B	40.0	50.0	60.0
C	212.3	301.5	405.2

This time the columns and indexes have names or labels as per the given **columns** and **index** sequences respectively.

In above examples, the ndarrays that are passed to DataFrame have same number of elements in each of the rows. If, however, the rows of ndarrays differ in length, *i.e.*, if number of elements

in each row differ, then Python will create just single column in the dataframe object and the type of the column will be considered as **object** (see below).

```
>>> narr3 = np.array([[101.5, 201.2], [400, 50, 600, 700], [212.3, 301.5, 405.2]] )
>>> narr3
array([list([101.5, 201.2]), list([400, 50, 600, 700]),
       list([212.3, 301.5, 405.2])], dtype = object)
>>> dtf4 = pd.DataFrame(narr3)
>>> dtf4
```

2 values, 4 values, 3 values

See the datatype of the ndarray is **object** this time

Single column created this time because the lengths of rows of ndarray did not match

```
                0
0       [101.5, 201.2]
1       [400, 50, 600, 700]
2       [212.3, 301.5, 405.2]
```

EXAMPLE **27** *What will be the output of following code ?*

```
import pandas as pd
import numpy as np
arr1 = np.array([ [11, 12], [13, 14 ], [15, 16] ], np.int32)
dtf2 = pd.DataFrame(arr1)
print(dtf2)
```

SOLUTION

```
    0   1
0  11  12
1  13  14
2  15  16
```

EXAMPLE **28** *Write a program to create a DataFrame from a 2D array as shown below :*

101	113	124
130	140	200
115	216	217

SOLUTION

```
import pandas as pd
import numpy as np
arr2 = np.array([ [101, 113, 124], [130, 140, 200 ], [115, 216, 217] ] )
dtf3 = pd.DataFrame(arr2)
print(dtf3)
```

Output

```
     0    1    2
0  101  113  124
1  130  140  200
2  115  216  217
```


4. Creating a DataFrame Object from a 2D Dictionary with Values as Series Objects

You can also create a DataFrame object by using multiple Series objects. In a 2D dictionary, you can have the values parts as Series objects and then you can pass this dictionary as argument to create a DataFrame object.

We have following Series objects :

```
>>> staff = pd.Series([20, 36, 44])
>>> salaries = pd.Series([166000, 246000, 563000])
```

And a dictionary that stores these Series objects as values :

```
>>> school = {'people':staff, 'Amount':salaries }
```

You can create a DataFrame object from above 2D dictionary by passing its name as argument :

```
>>> dtf4 = pd.DataFrame(school)
```

Now the DataFrame **dtf4**, created above, will be like :

```
>>> dtf4
   Amount     people
0  166000         20
1  246000         36
2  563000         44
```

EXAMPLE **29** *Consider two series objects* ***staff*** *and* ***salaries*** *that store the number of people in various office branches and salaries distributed in these branches, respectively.*

Write a program to create another Series object that stores average salary per branch and then create a DataFrame object from these Series objects.

SOLUTION

```
import pandas as pd
import numpy as np
staff= pd.Series([20, 36, 44])
salaries = pd.Series([279000, 396800, 563000])
avg = salaries / staff          # it will create avg series object
org= {'people':staff, 'Amount':salaries, 'Average':avg }
dtf5 = pd.DataFrame(org)
print(dtf5)
```

Output

```
   Amount        Average  people
0  279000  13950.000000      20
1  396800  11022.222222      36
2  563000  12795.454545      44
```

5. Creating a DataFrame Object from another DataFrame Object

You can pass an existing DataFrame object to **DataFrame( )** and it will create another dataframe object having similar data. Consider the code shown below that passes to **DataFrame( )** another dataframe object.

Given a DataFrame object **dtf1** :

```
>>> dtf1
   0  1  2
0  1  2  3
1  4  5  6
```

You can create an identical dataframe by passing its name (dtf1) to DataFrame( ) :

```
dfnew = pd.DataFrame(dtf1)
```

You can display the new DataFrame object to confirm that it is identical to dtf1:

```
>>> dfnew
   0  1  2
0  1  2  3
1  4  5  6
```

Please note there are some other methods too for creating dataframe objects but covering those will be beyond the scope of the book.

Displaying a DataFrame

Displaying a dataframe is the same as the way you display other variables and objects, *i.e.*, on the console prompt, either type its name or give **print( )** command with the *dataframe* object as you have been doing till now.

NOTE

DataFrames can also be created from text/CSV files, which we shall cover in **Chapter 4 — Data Transfer between DataFrames and Flat files/MySQL.**

1.7 DataFrame Attributes

To see DataFrame Attributes in action

Scan QR Code

When you create a DataFrame object, all information related to it (such as its *size*, its *datatype* etc.) is available through its *attributes*. You can use these attributes in the following format to get information about the dataframe object.

```
<DataFrame object>.<attribute name>
```

Some common attributes of DatFrame object are listed in table below. The code examples for the usage of these attributes follow the Table 1.3

Table 1.3 *Common attributes of DataFrame Objects*

Attribute	Description
index	The index (row labels) of the DataFrame.
columns	The column labels of the DataFrame.
axes	Return a list representing both the axes (axis 0 *i.e.*, index and axis 1, *i.e.*, columns) of the DataFrame.
dtypes	Return the dtypes of data in the DataFrame.
size	Return an int representing the number of elements in this object.
shape	Return a tuple representing the dimensionality of the DataFrame.
values	Return a Numpy representation of the DataFrame.
empty	Indicator whether DataFrame is empty.
ndim	Return an int representing the number of axes/array dimensions.
T	Transpose index and columns.

We are using the following DataFrame (**dfn**) to display various attributes, counting, transpose etc.

```
>>> dfn
      Marketing   Sales
age      25         24
name     Neha       Rohit
sex      Female     Male
```

(a) Retrieving Various Properties of a DataFrame Object

To view the value of an attribute, just give its name with the dataframe's name as depicted in following examples :

```
>>> dfn.index
Index(['age', 'name', 'sex'], dtype = 'object')
>>> dfn.columns
Index(['Marketing', 'Sales'], dtype = 'object')
>>> dfn.axes
[Index(['age', 'name', 'sex'], dtype = 'object'),
Index(['Marketing', 'Sales'], dtype = 'object')]
>>> dfn.dtypes
Marketing   object
Sales       object
dtype:      object
```

```
>>> dfn.size
6
>>> dfn.shape
(3, 2)
>>> dfn.ndim
2
>>> dfn.empty
False
```

(b) Getting Number of Rows in a DataFrame

The **len(<DF object>)** will return the number of rows in a dataframe *e.g.*,

```
>>> len(dfn)
    3
```

(c) Getting Count of non-NA Values in DataFrame

Like Series, you can use **count( )** with dataframe too to get the count of non-NaN or non-NA values, but **count( )** with dataframe is little elaborate :

(*i*) If you do not pass any argument or pass 0 (default is 0 only), then it returns count of non-NA values for each column, *e.g.*,

```
>>> dfn.count()
Marketing  3
Sales      3
dtype: int64
```

dfn.count( ) or dfn.count(0) or dfn.count(axis = 'index') will produce the same result

```
>>> dfn.count(axis = 'index')
Marketing  3
Sales      3
dtype: int64
```

You may also pass argument **axis = 'index'** to get the same result as above.

(*ii*) If you pass argument as 1, then it returns count of non-NA values for each row, *e.g.*,

```
>>> dfn.count(1)
age   2
name  2
sex   2
dtype: int64
```

dfn.count(1) or dfn.count(axis='columns') will produce the same result

```
>>> dfn.count(axis = 'columns')
age   2
name  2
sex   2
dtype: int64
```

You may also pass argument **axis = 'columns'** to get the same result as above.

(d) Transposing a DataFrame

You can transpose a dataframe by swapping its indexes and columns by using attribute T as shown below :

```
>>> dfn.T
           age  name   sex
Marketing   25  Neha   Female
Sales       24  Rohit  Male
```

Compare the transpose(dfn.T) with original dfn :

```
>>> dfn
          Marketing  Sales
age          25        24
name         Neha      Rohit
sex          Female    Male
```

EXAMPLE 30 *Write a program to create a DataFrame to store weight, age and names of 3 people. Print the DataFrame and its transpose.*

SOLUTION

```
import pandas as pd
# Creating the DataFrame
df = pd.DataFrame({'Weight' :[42, 75, 66],
                   'Name':['Arnav', 'Charles', 'Guru'],
                   'Age' :[15, 22, 35]})
print('Original Dataframe')
print(df)
print('Transpose:')
print(df.T)
```

Output

```
Original Dataframe
   Age    Name  Weight
0   15   Arnav      42
1   22 Charles      75
2   35    Guru      66
Transpose:
            0        1     2
Age        15       22    35
Name     Arnav  Charles  Guru
Weight     42       75    66
```

NOTE

You can also use **shape[0]** to see the number of rows and **shape[1]** for getting number of columns, *i.e.*,

```
df.shape[0]
df.shape[1]
```

(e) Numpy Representation of DataFrame

You can represent the values of a dataframe object in numpy way using *values* attribute :

```
>>> dfn.values
array([['25', '24'],
       ['Neha', 'Rohit'],
       ['Female', 'Male']], dtype = object)
```


1.8 Selecting or Accessing Data

From a DataFrame object, you can extract or select desired rows and columns as per your requirement. Let us see how.

For all the examples in this section, in coming lines, we are using the following DataFrame **dtf5** :

DataFrame : dtf5

	Population	Hospitals	Schools
Delhi	10927986	189	7916
Mumbai	12691836	208	8508
Kolkata	4631392	149	7226
Chennai	4328063	157	7617

1.8.1 Selecting/Accessing a Column

Selecting a column is easy, just use the following syntax :

```
<DataFrame object> [ <column name>]
```
← *Using square brackets*

Or

```
<DataFrame object>.<column name>
```
← *Using dot notation*

Now, consider the following example accessing columns ***Population, Schools*** from dataframe **dtf5**.

```
>>> dtf5['Population']
Delhi       10927986
Mumbai      12691836
Kolkata      4631392
Chennai      4328063
Name: Population, dtype: int64
```

```
>>> dtf5['Schools']
Delhi       7916
Mumbai      8508
Kolkata     7226
Chennai     7617
Name : Schools, dtype : int64
```

In the dot notation, make sure not to put any quotation marks around the column name.

For example,

```
>>> dtf5.Population
Delhi       10927986
Mumbai      12691836
Kolkata      4631392
Chennai      4328063
Name: Population, dtype: int64
```

While using dot nation, do not put any quotes around the column name

1.8.2 Selecting/Accessing Multiple Columns

To select multiple columns, you can give a list having multiple column names inside the square brackets with dataframe object, *i.e.*, as follows :

```
<DataFrame object> [[<column name>, <column name>, <column name>, ...]]
```

For example,

Notice double square brackets

```
>>> dtf5[ ['Schools', 'Hospitals'] ]
          Schools    Hospitals
Delhi     7916       189
Mumbai    8508       208
Kolkata   7226       149
Chennai   7617       157
```

List having multiple column names given inside the square brackets

Columns appear in the order of column names given in the **list inside square brackets** (see below). Compare it with above result too.

```
>>> dtf5[ ['Hospitals', 'Schools'] ]
          Hospitals  Schools
Delhi     189        7916
Mumbai    208        8508
Kolkata   149        7226
Chennai   157        7617
```

Columns appear in the order of column names given in the list inside the square brackets

EXAMPLE 31 *Given a DataFrame namely* ***aid*** *that stores the aid by NGOs for different states :*

	Toys	Books	Uniform	Shoes
Andhra	7916	6189	610	8810
Odisha	8508	8208	508	6798
M.P.	7226	6149	611	9611
U.P.	7617	6157	457	6457

Write a program to display the aid for

(i) Books and Uniform only *(ii) Shoes only*

SOLUTION

```
import pandas as pd
:       # DataFrame aid created or loaded
print("Aid for books and uniform:")
print(aid[['Books', 'Uniform'] ])
print("Aid for shoes:")
print(aid.Shoes)
```

Output

```
Aid for books and uniform:
          Books  Uniform
Andhra    6189   610
Odisha    8208   508
M.P.      6149   611
U.P.      6157   457
Aid for shoes:
Andhra    8810
Odisha    6798
M.P.      9611
U.P.      6457
Name: Shoes, dtype: int64
```

1.8.3 Selecting/Accessing a Subset from a DataFrame using Row/Column Names

To access row(s) and/or a combination of rows and columns, you can use following syntax to select/access a subset from a dataframe object :

```
<DataFrameObject>.loc [<startrow> : <endrow>, <startcolumn> : <endcolumn>]
```


The above syntax is a general syntax through which you can single/multiple rows /columns. Let us see some examples :

- **To access a row**, just give the row name/label as this : <DF object>.loc [<row label> , :]. Make sure not to miss the COLON AFTER COMMA.

```
>>> dtf5.loc['Delhi', :]
Population    10927986
Hospitals       189
Schools        7916
Name: Delhi
```

Use this syntax to access one row : <DF>.loc [<row label> , :]

```
>>> dtf5.loc['Chennai', :]
Population    4328063
Hospitals       157
Schools        7617
Name: Chennai
```

- **To access multiple rows**, use : <DF object>.**loc** [<start row > : <end row> , :]. Make sure not to miss the COLON AFTER COMMA.

```
>>> dtf5.loc['Mumbai' :'Kolkata', :]
          Population  Hospitals  Schools
Mumbai    12691836    208        8508
Kolkata   4631392     149        7226
```

Give start and end row indexes with **<DFobject>.loc.** Make sure not to forget comma and colon after comma

Please note that when you specify <start row> :<end row>, the Python will return all rows falling between ***start row*** and ***end row***, along with ***start row*** and ***endrow***. (see below)

```
>>> dtf5.loc['Mumbai' :'Chennai', :]
          Population  Hospitals  Schools
Mumbai    12691836    208        8508
Kolkata   4631392     149        7226
Chennai   4328063     157        7617
```

To see subset from a dataframe in action

Scan QR Code

- **To access selective columns**, use : <DF object>.**loc** [: , <start column > : <end column>]. Make sure not to miss the COLON BEFORE COMMA. Like rows, all columns falling between start and end columns, will also be listed :

```
>>> dtf5.loc[ : , 'Population' :'Schools']
          Population  Hospitals  Schools
Delhi     10927986    189        7916
Mumbai    12691836    208        8508
Kolkata   4631392     149        7226
Chennai   4328063     157        7617
```

All columns falling between start and end columns, are listed

```
>>> dtf5.loc[ : , 'Population' :'Hospitals']
          Population  Hospitals
Delhi     10927986    189
Mumbai    12691836    208
Kolkata   4631392     149
Chennai   4328063     157
```

◈ To access range of columns from a range of rows, use :

```
<DF object>.loc [<startrow> : <endrow>, <startcolumn> : <endcolumn>].
```

```
>>>dtf5.loc['Delhi' : 'Mumbai' , 'Population' :'Hospitals']
        Population  Hospitals
Delhi     10927986        189
Mumbai    12691836        208
```

Selecting a range of columns from a range of rows

EXAMPLE 32 *Given a DataFrame namely* **aid** *that stores the aid by NGOs for different states :*

	Toys	Books	Uniform	Shoes
Andhra	7916	6189	610	8810
Odisha	8508	8208	508	6798
M.P.	7226	6149	611	9611
U.P.	7617	6157	457	6457

Write a program to display the aid for states 'Andhra' and 'Odisha' for Books and Uniform only.

SOLUTION

```
import pandas as pd
:       # DataFrame aid created or loaded
print( aid.loc['Andhra':'Odisha', 'Books':'Uniform'])
```

Output

	Books	Uniform
Andhra	6189	610
Odisha	8208	508

1.8.4 Selecting Rows/Columns from a DataFrame

Sometimes your dataframe object does not contain row or column labels or even you may not remember them. In such cases, you can extract subset from dataframe using the row and column ***numeric index/position,*** but this time you will use **iloc** instead of **loc. iloc** means ***integer location*** .

```
<DF object>.iloc[<start row index> : <end row index>, <start col index> : <end column index>]
```

When you use **iloc**, then **<startindex> :<end index>** given for rows and columns work like slices, and the end index is excluded (unlike **loc**), just as in the slices.

Consider the following example :

```
>>> dtf5.iloc[0:2, 1:3]
        Hospitals   Schools
Delhi         189      7916
Mumbai        208      8508

>>> dtf5.iloc[0:2, 1:2]
        Hospitals
Delhi         189
Mumbai        208
```

With iloc, the end index is excluded in result

NOTE

With **loc**, both start label and end label are included when given as start : end, but with **iloc**, like slices end index/position is excluded when given as start : end.

The **loc** and **iloc** are very flexible and can be used in variety of ways as shown below.

EXAMPLE 33 *Consider a dataframe* **df** *as shown below.*

Sample DataFrame *df* (Reference 1.7*a*)

```
    Item Type Sales Channel  Order Date   Order ID  Total Revenue  Total Cost  Total Profit
0     Cosmetics      Online   5/22/2017       8985      793518.00   477943.95     315574.05
1        Cereal      Online   5/20/2017       5559     1780539.20  1013704.16     766835.04
2  Personal Care     Offline    5/8/2017       4567      523807.57   363198.03     160609.54
3  Personal Care      Online   3/11/2017       6992      246415.95   170860.05      75555.90
4        Snacks      Online   2/25/2017       7562     1117953.66   713942.88     404010.78
5     Household     Offline    2/8/2017      52284     5997054.98  4509793.96    1487261.02
6          Meat      Online   1/14/2017       8253     2011149.63  1738477.23     272672.40
7       Clothes      Online   1/13/2017       1873      902980.64   296145.92     606834.72
8     Cosmetics      Online  12/31/2016  331438481     3876652.40  2334947.11    1541705.29
9  Office Supplies     Offline  12/6/2016       6213      617347.08   497662.08     119685.00
10    Cosmetics     Offline  11/19/2016  419123971     3039414.40  1830670.16    1208744.24
11    Cosmetics      Online  11/15/2016  286959302     2836990.80  1708748.37    1128242.43
12    Beverages     Offline  10/23/2016  345718562      221117.00   148141.40      72975.60
13      Clothes     Offline   7/25/2016  807025039      600821.44   197048.32     403773.12
14       Snacks      Online   6/30/2016  795490682      339490.50   216804.00     122686.50
```

Write statements to do the following :

(*i*) *Display rows 2 to 4 (both inclusive)*

(*ii*) *From rows 2 to 4 (both inclusive), display columns, 'Item Type' and 'Total Profit'.*

(*iii*) *From rows 2 to 4 (both inclusive), display first four columns.*

SOLUTION

(*i*)

```
>>> df [2:5]
```

```
In [14]: df[ 2:5 ]
Out[14]:
      Item Type Sales Channel Order Date  Order ID  Total Revenue  Total Cost  Total Profit
2  Personal Care      Offline   5/8/2017      4567      523807.57   363198.03     160609.54
3  Personal Care       Online  3/11/2017      6992      246415.95   170860.05      75555.90
4         Snacks       Online  2/25/2017      7562     1117953.66   713942.88     404010.78
```

(*ii*)

```
>>> df.loc [ 2:5 , ['Item Type', 'Total Profit']]
```

```
In [18]: df.loc [ 2:5 , ['Item Type', 'Total Profit'] ]
Out[18]:
      Item Type  Total Profit
2  Personal Care     160609.54
3  Personal Care      75555.90
4         Snacks     404010.78
5      Household    1487261.02
```

(*iii*)

```
>>> df.iloc[2:5,0:4]
```

```
In [19]: df.iloc[2:5,0:4]
Out[19]:
      Item Type Sales Channel Order Date  Order ID
2  Personal Care      Offline   5/8/2017      4567
3  Personal Care       Online  3/11/2017      6992
4         Snacks       Online  2/25/2017      7562
```

Recall that when you use iloc, then <startindex> :<end index> given for rows and columns work like slices, and the end index is excluded

1.8.5 Selecting/Accessing Individual Value

To select/access an individual data value from a dataframe, you can use any of the following methods :

(*i*) Either give name of row or numeric index in square brackets with, *i.e.*, as this :

```
<DF object>.<column> [<row name or row numeric index>]
```

Consider the following examples :

```
>>> dtf5.Population['Delhi']
10927986
>>> dtf5.Population[1]
12691836
```

Either give name of row or numeric index with **DFobject.column[ ]** in square brackets. Both will yield same result

(*ii*) You can use **at** or **iat** attributes with DF object as shown below :

Use	*To*
`<DFobject>.at[<row label>, col label>]`	Access a single value for a row/ column label pair.
`<DFobject>.iat[<row index no.>, col index no.>,]`	Access a single value for a row/ column pair by integer position.

Consider examples given below :

```
>>> dtf5.at['Chennai', 'Schools']
7617
>>> dtf5.iat[3, 2]
7617
```

You can give *row and column names* with **at** attribute and *numeric row index and numeric column index* with **iat** attribute to access individual data value

1.9 Adding/Modifying Rows'/Columns' Values in DataFrames

You can assign or modify data in a dataframe in the same way as you do with other objects. All you need to do is to specify the row name and/or column name along with the dataframe's name. The process of adding and modifying rows'/columns' value is similar, as you will see in the following sub-sections.

1.9.1 Adding/Modifying a Column

Columns in a dataframe can be referred to in multiple ways. Assigning a value to a column :

- will modify it, if the column already exists.
- will add a new column, if it does not exist already.

To change or add a column, use syntax :

```
<DF object >.<column name> = <new value>
```

or

```
<DF object >.[<column>] = <new value>
```


If the given column name does not exist in dataframe then a new column with this name is added :

```
>>> dtf5['Density'] = 1219
>>> dtf5
         Population  Hospitals  Schools  Density
Delhi      10927986        189     7916     1219
Mumbai     12691836        208     8508     1219
Kolkata     4631392        149     7226     1219
Chennai     4328063        157     7617     1219
```

Since there is no column with name 'Density', Python added a new column with this name which has given value for all its rows

Although the above method adds a column BUT here the catch is that all the rows of this new column have the same given value. If you want to add a proper new column that has different values for all its rows, then you can assign the data values for each row of the column in the form of a list, *i.e.*, as shown below :

NOTE

If you assign something to <DF>[<columnname>] or using **at** or **loc** and if column name does not exist, then a new column is created and that has the same value for all its rows.

```
>>> dtf5['Density'] = [ 1500, 1219, 1630, 1050, 1100]
>>> dtf5
           Population  Hospitals  Schools  Density
Delhi      10927986.0      189.0   7916.0   1500.0
Mumbai     12691836.0      208.0   8508.0   1219.0
Kolkata     4631392.0      149.0   7226.0   1630.0
Chennai     4328063.0      157.0   7617.0   1050.0
Banglore    5678097.0     1200.0   1200.0   1100.0
```

This time Python has added new column with different values for all its row as per the list of values

Same way, you can modify an existing column by assigning a new list of values to it. That is, for existing column, it will change the data values and for non-existing column, **it will add a new column**. There are some other ways of adding a column to a dataframe. These are :

```
<DF object>.at[ : , <column name> ] = <values for column>
```

Or

```
<DF object>.loc[ : , <column name> ] = <values for column>
```

Or

```
<DF object>.= <DF object>.assign(<column name> =<values for column>)
```

For example, given below are some statements that will add a new column if the mentioned column name does not exist in a dataframe :

```
dtf5.loc[:, "Density"] = [1500, 1219, 1630, 1050, 1100]
dtf5 = dtf4.assign( Density = [1500, 1219, 1630, 1050, 1100] )
```

NOTE

When you assign something to a column of dataframe, then for existing column, it will change the data values and for non-existing column, it will add a new column.

You just need to make sure in above examples that the sequence which contains values for the new column must have values equal to number of rows in the dataframe otherwise Python will give error (error : **ValueError**).

1.9.2 Adding/Modifying a Row

Like columns, you can change or add rows to a DataFrame using **at** or **loc** attributes as explained below :

To change or add a row, use syntax :

```
<DF object>.at[<row name>, : ] = <new value>
```

Or

```
<DF object>.loc[<row name>, : ] = <new value>
```

Likewise, if there is no row with such row label, then Python adds new row with this *row label* and assigns given values to all its columns :

```
>>> dtf5.at['Banglore', :] = 1200
>>> dtf5
           Population  Hospitals  Schools  Density
Delhi      10927986.0      189.0   7916.0   1219.0
Mumbai     12691836.0      208.0   8508.0   1219.0
Kolkata     4631392.0      149.0   7226.0   1219.0
Chennai     4328063.0      157.0   7617.0   1219.0
Banglore       1200.0     1200.0   1200.0   1200.0
```

Since there is no row with label 'Banglore', Python added new row with this label which has same given value for all its columns

As you can see in the above output that if a mentioned row label with **at** or **loc** attributes does not exist in the DataFrame, Python will create a new row for it and this is how a new row is added to a DataFrame. But there is a catch – if you specify only a single value, then all the values in the newly added row will have the same value as it did in the above output.

You can add a new row by specifying individual values for each column. For this, specify all the values of the new row to be added as a sequence such as a list etc., *e.g.*,

```
>>> dtf5.at['Bangalore', :] = [10002980, 171, 7311, 1200]
>>> dtf5
           Population  Hospitals  Schools  Density
Delhi      10927986.0      189.0   7916.0   1219.0
Mumbai     12691836.0      208.0   8508.0   1219.0
Kolkata     4631392.0      149.0   7226.0   1219.0
Chennai     4328063.0      157.0   7617.0   1219.0
Banglore   10002980.0      171.0   7311.0   1200.0
```

See, this time values are specified for all the columns in the form of a list

See, this time the newly added row contains the values given in the list

While adding a row this way, make sure that the sequence containing values for different columns has **values for all the columns**, otherwise Python will raise **ValueError** (*see* below).

If you try to add a row having 4 values to the above DataFrame dtf5 having 5 columns, Python will give you error :

```
>>> dtf5.loc['Mohali', :] = [452980, 71, 211]
:
ValueError: cannot copy sequence with size 3 to array axis with dimension 4
```

This statement is trying to insert a row having 3 values in the dataframe with four columns and this the error.

NOTE

You can use **at** or **loc** attributes of a DataFrame to add/modify a row, column or individual cell.

EXAMPLE **34** *Consider the following dataframe* ***saleDf*** :

	Target	Sales
zoneA	56000	58000
zoneB	70000	68000
zoneC	75000	78000
zoneD	60000	61000

Write a program to add a column namely **Orders** *having values 6000, 6700, 6200 and 6000 respectively for the zones A, B, C and D. The program should also add a new row for a new zone* **ZoneE**. *Add some dummy values in this row.*

SOLUTION

```
import pandas as pd
:   # saleDf created or loaded here
saleDf['Orders'] = [6000, 6700, 6200, 6000 ]
saleDf.loc['zoneE', :]= [ 50000, 45000, 5000]
print(saleDf)
```

Output

	Target	Sales	Orders
zoneA	56000.0	58000.0	6000.0
zoneB	70000.0	68000.0	6700.0
zoneC	75000.0	78000.0	6200.0
zoneD	60000.0	61000.0	6000.0
zoneE	50000.0	45000.0	5000.0

1.9.3 Modifying a Single Cell

You may use <DataFrame>.iat[<row position>, <column position>] to modify values using row and column position. (*Refer Multiple Choice Question 22 given in Objective type questions at the end of this chapter.*)

To change or modify a single data value, use syntax :

```
<DF>.<columnname>[<row name/label>] = <new/modified value>
```

```
>>> dtf5.Population['Banglore'] = 5678097
>>> dtf5
           Population  Hospitals  Schools  Density
Delhi      10927986.0      189.0   7916.0   1219.0
Mumbai     12691836.0      208.0   8508.0   1219.0
Kolkata     4631392.0      149.0   7226.0   1219.0
Chennai     4328063.0      157.0   7617.0   1219.0
Banglore    5678097.0     1200.0   1200.0   1200.0
```

See, this time only this cell got modified

1.10 Deleting/Renaming Columns/Rows

Python Pandas provides two ways to delete rows and columns — **del** statement and **drop( )** function. Pandas also provides **rename( )** function to rename rows and columns. In this section, we shall talk about how rows/columns can be deleted or renamed.

Let us now talk about how you can delete or rename columns and rows.

1.10.1 Deleting Rows/Columns in a DataFrame

To delete a column, you use **del** statement as this :

```
del <Df object>[<column name>]
```

For example,

```
>>> del dtf5['Density']
>>> dtf5
          Population   Hospitals   Schools
Delhi     10927986.0   189.0       7916.0
Mumbai    12691836.0   208.0       8508.0
Kolkata   4631392.0    149.0       7226.0
Chennai   4328063.0    157.0       7617.0
Banglore  5678097.0    1200.0      1200.0
```

Column 'Density' deleted

To delete rows from a dataframe, you can use **<DF>.drop(index or sequence of indexes)**, *e.g.*,

Both these commands will delete the rows with indexes 2, 4, 6, 8, 12 from dataframe **df** :

```
df.drop(range(2, 13, 2))
df.drop([2, 4, 6, 8, 12])
```

Argument to drop( ) should be either an index, or a sequence containing indexes

You can also give **axis = 1** along with indexes/labels then **drop( )** will drop the columns, *i.e.*, the following command will drop the mentioned columns from dataframe df :

```
df.drop(["Total Cost", "Order ID"], axis = 1)
```

This argument signifies to delete columns

EXAMPLE **35** *From the dtf5 used above, create another DataFrame and it must not contain the column 'Population' and the row Bangalore.*

SOLUTION

```
import pandas as pd
:       # DataFrame dtf5 created or loaded
dtf6 = pd.DataFrame(dtf5)
del dtf6['Population']
dtf6 = dtf6.drop(['Bangalore])
print(dtf6)
```

Output

```
          Hospitals   Schools
Delhi     189.0       7916.0
Mumbai    208.0       8508.0
Kolkata   149.0       7226.0
Chennai   157.0       7617.0
```


1.10.2 Renaming Rows/Columns

To change the name of any row/column individually, you can use the rename( ) function of DataFrame as per the syntax given below.

```
<DF>.rename( index = {<names dictionary>}, columns = {<names dictionary>}, inplace = False )
```

where,

- The **index** argument is for index names (row labels). If you want to rename rows only then specify only **index** argument.
- The **columns** argument is for the **columns** names. If you want to rename **columns** only then specify only **columns** argument.
- For both **index** and **columns** arguments, specify the **names-change dictionary** containing original names and the new names in a form like {**old name: new name**}
- Specify **inplace** argument as ***True*** if you want to rename the rows/columns in the same dataframe as if you skip this argument, then a new dataframe is created with new indexes/columns' names and original remains unchanged.

Let us now practically see how **rename( )** works.

Consider the dataframe **topDf** shown below :

```
        Rollno   Name   Marks
Sec A   115      Pavni  97.5
Sec B   236      Rishi  98.0
Sec C   307      Preet  98.5
Sec D   422      Paula  98.0
```

To change the row labels as *'A', 'B', 'C', 'D'*, you can write :

```
>>> topDf.rename( index = {'Sec A':'A', 'Sec B':'B', 'Sec C':'C', 'Sec D':'D'} )
   Rollno   Name   Marks
A  115      Pavni  97.5
B  236      Rishi  98.0
C  307      Preet  98.5
D  422      Paula  98.0
```

See, the names dictionary for index argument is storing the old and new index names

*The output of **rename()** has shown the changed indexes but are these changed in original **topDf** as well ?*

The above statement will show the changed indexes but when you display the dataframe **topDf** after executing above statement, it will show you the original dataframe (*see* below) because by default **rename( )** creates a new dataframe with changed names.

```
>>> topDf
        Rollno   Name   Marks
Sec A   115      Pavni  97.5
Sec B   236      Rishi  98.0
Sec C   307      Preet  98.5
Sec D   422      Paula  98.0
```

See, the rename( ) function by default does not make changes in the original dataframe; it creates a new dataframe with the changes and the original dataframe remains unchanged

*See the original dataframe **topDf** is unchanged*

To make changes in the index/columns' names original dataframe, you need to specify additional argument **inplace = True** in the **rename( )**, *e.g.,*

```
>>>topDf.rename(index = {'Sec A':'A', 'Sec B':'B', 'Sec C':'C', 'Sec D':'D'}, inplace = True)
>>> topDf
```

	Rollno	Name	Marks
A	115	Pavni	97.5
B	236	Rishi	98.0
C	307	Preet	98.5
D	422	Paula	98.0

By giving inplace =True argument, it will now make the changes in the original dataframe

*With **inplace** as True, **rename()** has changed the indexes in original **topDf***

You can use **rename( )** to change columns names too, e.g., following statement will change the columns names in the **topDf** dataframe. (Consider the original **topDf** shown earlier in the beginning of this section) You can choose to change selective columns' names or all the columns' names. Only the **names** given in the **name-change dictionary** will get changed. *For example,*

```
>>> topDf.rename(columns ={'Rollno':'Rno'})
```

	Rno	Name	Marks
Sec A	115	Pavni	97.5
Sec B	236	Rishi	98.0
Sec C	307	Preet	98.5
Sec D	422	Paula	98.0

the name-change dictionary containing old and new names.

You can combine any of these arguments together : **index, columns, inplace** arguments.

Following examples are illustrating the same.

EXAMPLE **36** *Consider the **saleDf** shown below.*

	Target	Sales
zoneA	56000	58000
zoneB	70000	68000
zoneC	75000	78000
zoneD	60000	61000

Write a program to rename indexes of 'zoneC' and 'zoneD' as 'Central' and 'Dakshin' respectively and the column names 'Target and 'Sales' as 'Targeted' and 'Achieved' respectively.

SOLUTION

```
import pandas as pd
:  # saleDf created or loaded here
print(saleDf.rename(index = {'zoneC':'Central', 'zoneD':'Dakshin'}, \
   columns = {'Target':'Targeted', 'Sales':'Achieved'}) )
```

Output

	Targeted	Achieved
zoneA	56000	58000
zoneB	70000	68000
Central	75000	78000
Dakshin	60000	61000

EXAMPLE **37** *Will the previous program reflect the renamed indexes and columns' names in the dataframe saleDf ? Make changes in the previous program so that dataframe **saleDf** has the changed indexes and columns.*

SOLUTION

The previous program will not reflect the changed indexes and columns' names in the dataframe saleDf on which rename( ) is applied because **inplace = True** argument is missing. The changed program to make changes in saleDf would be :

```
import pandas as pd
:   # saleDf created or loaded here
saleDf.rename( index = {'zoneC':'Central', 'zoneD':'Dakshin'}, \
               columns = {'Target':'Targeted', 'Sales':'Achieved'}), inplace = True )
print(saleDf)
```

1.11 More on DataFrame Indexing — BOOLEAN INDEXING

Till now you have learnt to create and use DataFrames in variety of ways. You have also learnt to change *indexes, column names, rename* them etc. Let us now talk about an interesting feature of DataFrames – **Boolean Indexing**.

Boolean Indexing, as the name suggests, means having Boolean Values [(*True or False*) or (1 or 0) sometimes] as indexes of a dataframe. You might be wondering – Why? What is the need for having indexes as *True* or *False* ?

Well, your question is genuine and so is the answer. Actually, in some situations, we may need to divide our data in two subsets – *True* or *False*, *e.g.,* your school has decided to launch online classes for you. But some days of the week are designated for it. So, a dataframe related to this information might look like :

The Boolean indexes divide the dataframe in two groups – *True* rows and *False* rows.

	Day	No. of Classes
True	Monday	6
False	Tuesday	0
True	Wednesday	3
False	Thursday	0
True	Friday	8

The Boolean indexes divide the dataframe in *two* groups – *True rows* and *False rows*. This is useful division in situations where you find out things like – On which days, the *online classes* are held ? Or Which ones are *offline classes* days ? And so on.

Boolean indexes can either be in True/False form or in (1 or 0) form.

BOOLEAN INDEXING

Boolean Indexing refers to having Boolean Values [(True or False) or (1 or 0)] as indexes of a dataframe.

Check Point 1.1

1. What is the use of Python Pandas library ?
2. Name the Pandas object that can store one dimensional array like object and can have numeric or labelled indexes.
3. Can you have duplicate indexes in a series object ?
4. What do these attributes of series signify ? (*a*) size (*ii*) itemsize (*iii*) nbytes
5. If S1 in a series object then how will *len(S1)* and *S1.count( )* behave ?
6. What are NaNs ? How do you store them in a datastructures ?
7. True/False. Series objects always have indexes 0 to $n-1$.
8. What is the use of del statement ?
9. What does drop( ) function do ?
10. What is the role of **inplace** argument in rename( ) function.

1.11.1 Creating Dataframes with Boolean Indexes

Let us first create a dataframe with Boolean indexes *True* and *False*. While creating a dataframe with Boolean indexes ***True*** and ***False***, make sure that *True* and *False* are not enclosed in quotes (*i.e.*, like 'true' or 'False'), otherwise it will give you error (**KeyError**) while accessing data with Boolean indexes using **.loc**, because 'True' and 'False' are string values, not Boolean values.

The ***solved problem*** 27 depicts the same problem.

Let us first create above shown dataframe containing online classes' information, through the code given below.

```
import pandas as pd
Days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday' ]
Classes = [ 6, 0, 3, 0, 8]
dc = {'Days':Days, 'No. of Classes':Classes}
clasDf = pd.DataFrame(dc, index = [True, False, True, False, True] )
```

Boolean indexes provided for each row. Notice True and False values are Boolean values not strings (not enclosed in quotes

```
>>> clasDf
```

The Boolean values as indexes

	Days	No. of Classes
True	Monday	6
False	Tuesday	0
True	Wednesday	3
False	Thursday	0
True	Friday	8

You can also provide Boolean indexing to dataframes as ***1s*** and ***0s***. Let us create another dataframe (***clasDf1***) similar to the above shown dataframe (***clasDf***) having indexes as **1s** and **0s**, through the code given below.

```
import pandas as pd
Days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday' ]
Classes = [6, 0, 3, 0, 8]
dc = {'Days':Days, 'No. of Classes':Classes}
clasDf1 = pd.DataFrame(dc, index = [1, 0, 1, 0, 1] )
```

Boolean indexes provided for each row as 1s and 0s this time

```
>>> clasDf1
```

	Days	No. of Classes
1	Monday	6
0	Tuesday	0
1	Wednesday	3
0	Thursday	0
1	Friday	8

1.11.2 Accessing Rows from DataFrames with Boolean Indexes

Boolean indexing is very useful for filtering records, *i.e.*, for finding or extracting the *True* or *False* indexed rows. You can filter out records from dataframes with Boolean indexes using the **.loc** attribute as depicted below :

```
<DF>.loc[True]     # to display all records with True index
<DF>.loc[False]    # to display all records with False index
<DF>.loc[1]        # to display all records with index as 1
<DF>.loc[0]        # to display all records with index as 0
```

Use **.loc[ ]** with dataframes having Boolean indexes to filter out rows.

For example,

```
>>> clasDf.loc[True]
```

See, the result now contains only the rows having index as True

	Days	No. of Classes
True	Monday	6
True	Wednesday	3
True	Friday	8

```
>>> clasDf.loc[False]
```

See, the result now contains only the rows having index as False

	Days	No. of Classes
False	Tuesday	0
False	Thursday	0

```
>>> clasDf1.loc[0]
```

See, the result now contains only the rows having index as 0

	Days	No. of Classes
0	Tuesday	0
0	Thursday	0

With this we have come to the end of this chapter. Let us quickly revise what we have learnt in this chapter.

LET US REVISE

- *Pandas is the most popular library in the scientific Python ecosystem for doing data analysis.*
- *A Series type object is pandas data structure that represents a one-dimensional array-like object containing an array of data (of any NumPy supported data type) and an associated array of data labels, called its index.*
- *A Series type object can be created using panda.Series( ) where data may be specified as a sequence of any type or ndarrays.*
- *Series object's indexes are not necessarily from 0 to n – 1 always.*
- *You can use* ***numpy.NaN*** *to add missing data.*
- *Common Series attributes are : index, values, dtype, shape, nbytes, ndim, size, itemsize, hasnans, empty etc.*

- *A slice object is created from Series object using a syntax of <Object>[start : end : step].*
- *Series objects are value-mutable but size-immutable objects.*
- *Vector operations mean that if you apply a function or expression then it is individually applied on each item of the object.*
- *When you perform arithmetic operations on two Series type objects, the data is aligned on the basis of matching indexes and then performed arithmetic; for non-overlapping indexes, the arithmetic operations result as a NaN (Not a Number) – this is called Data Alignment in panda objects.*
- *A DataFrame is a two-dimensional labeled array like, pandas data structure that stores an ordered collection columns that can store data of different types.*
- *A dataframe object is created using pandas.DataFrame( ) and the values to it are passed in a two dimensional sequence such as a 2D ndarray or dictionary having lists/ndarrays/dictionary/Series etc. as values or another dataframe object.*
- *While creating a dataframe with a nested or 2d dictionary, Python interprets the outer dict keys as the columns and the inner keys as the row indices.*
- *Common dataframe objects include: index , columns, axes, dtypes, size, shape, values, empty, ndim, T.*
- *Transpose of a dataframe is available through <DF>.T*
- *When you assign something to a column of dataframe, then for existing column, it will change the data values and for non-existing column, it will add a new column.*
- *A column can be deleted using **del** command.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. To create an empty Series object, you can use :
   (a) pd.Series(empty) (b) pd.Series(np.NaN)
   (c) pd.Series( ) (d) all of these
2. To specify datatype *int16* for a Series object, you can write :
   (a) pd.Series(data = array, dtype = int16)
   (b) pd.Series(data = array, dtype = numpy.int16)
   (c) pd.Series(data = array.dtype = pandas.int16)
   (d) all of the above
3. To get the number of dimensions of a Series object, _____ attribute is displayed.
   (a) index (b) size (c) itemsize (d) ndim
4. To get the size of the datatype of the items in Series object, you can display _____ attribute.
   (a) index (b) size (c) itemsize (d) ndim
5. To get the number of elements in a Series object, _____ attribute may be used.
   (a) index (b) size (c) itemsize (d) ndim
6. To get the number of bytes of the Series data, _____ attribute is displayed.
   (a) hasnans (b) nbytes (c) ndim (d) dtype
7. To check if the Series object contains NaN values, _____ attribute is displayed.
   (a) hasnans (b) nbytes (c) ndim (d) dtype

8. To display third element of a Series object S, you will write _____ .
   (a) S[:3] (b) S[2] (c) S[3] (d) S[:2]
9. To display first three elements of a Series object S, you may write _____ .
   (a) S[:3] (b) S[3] (c) S[3rd] (d) all of these
10. To display last five rows of a Series object S, you may write _____ .
    (a) head( ) (b) head(5) (c) tail( ) (d) tail(5)
11. Missing data in Pandas object is represented through :
    (a) Null (b) None (c) Missing (d) NaN
12. If a Dataframe is created using a 2D dictionary, then the indexes/row labels are formed from _____ .
    (a) dictionary's values (b) inner dictionary's keys
    (c) outer dictionary's keys (d) none of these
13. If a dataframe is created using a 2D dictionary, then the column labels are formed from _____ .
    (a) dictionary's values (b) inner dictionary's keys
    (c) outer dictionary's keys (d) none of these
14. The axis 0 identifies a dataframe's _____ .
    (a) rows (b) columns (c) values (d) datatype
15. The axis 1 identifies a dataframe's _____ .
    (a) rows (b) columns (c) values (d) datatype
16. To get the number of elements in a dataframe, _____ attribute may be used.
    (a) size (b) shape (c) values (d) ndim
17. To get NumPy representation of a dataframe, _____ attribute may be used.
    (a) size (b) shape (c) values (d) ndim
18. To get a number representing number of axes in a dataframe, _____ attribute may be used.
    (a) size (b) shape (c) values (d) ndim
19. To get the transpose of a dataframe D1, you can write _____ .
    (a) D1.T (b) D1.Transpose (c) D1.Swap (d) All of these
20. To extract row/column from a dataframe, _____ function may be used.
    (a) row( ) (b) column( ) (c) loc( ) (d) All of these
21. To display the 3rd, 4th and 5th columns from the 6th to 9th rows of a dataframe DF, you can write _____ .
    (a) DF.loc[6:9, 3:5] (b) DF.loc[6:10, 3:6]
    (c) DF.iloc[6:10, 3:6] (d) DF.iloc[6:9, 3:5]
22. To change the 5th column's value at 3rd row as 35 in dataframe DF, you can write _____ .
    (a) DF[4, 6] = 35 (b) DF[3, 5] = 35
    (c) DF.iat[4, 6] = 35 (d) DF.iat[3, 5] = 35
23. Which among the following options can be used to create a DataFrame in Pandas ?
    (a) A scalar value (b) An ndarray (c) A python dict (d) All of these
24. Identify the correct statement :
    (a) The standard marker for missing data in Pandas is NaN.
    (b) Series act in a way similar to that of an array
    (c) Both (a) and (b) (d) None of the above

25. To delete a column from a DataFrame, you may use _____ statement.
(a) remove (b) del (c) drop (d) cancel
26. To delete a row from a DataFrame, you may use ____ statement.
(a) remove (b) del (c) drop (d) cancel

Fill in the Blanks

1. _____ is a popular data-science library of Python.
2. A _____ is a Pandas data structure that represents a 1 D array like object.
3. A _____ is a Pandas data structure that represents a 2 D array like object.
4. You can use numpy._____ for missing data.
5. To specify datatype for a Series object, _____ argument is used.
6. The _____ function on Series object returns total elements in it including NaNs.
7. The _____ function on Series object returns only the count of non-NaN values in it.
8. Series is _____ mutable.
9. Series is not _____ mutable.
10. Dataframe is _____ mutable as well as _____ mutable.
11. To access values using row labels you can use DF._____ .
12. To access individual value, you can use DF._____ using row/column index labels.
13. To access individual value, you can use DF._____ using row/column integer position.
14. The rename( ) function requires _____ argument to make changes in the original dataframe.

True/False Questions

1. A Pandas Series object can be thought of as a column or a row, essentially.
2. Both Series and DataFrame are one dimensional data structure objects.
3. While series is a one dimensional datastructure object, dataframe is a multi-dimensional data structure object.
4. A Series object is value mutable.
5. A Series object is size mutable.
6. A DataFrame object is value mutable.
7. A DataFrame object is size mutable.
8. There is no difference between a NumPy array and a Series object.
9. A DataFrame can be thought of as a group of multiple Series objects.
10. A DataFrame has similar properties as a Series object.
11. A Series object can store only homogeneous (same type of) elements.
12. A DataFrame object can store only homogeneous elements.
13. The del statement can remove the rows as well as columns in a dataframe.
14. The rename( ) always makes changes in the default dataframe.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *What is Pandas library of Python ? What is its significance ?*

Solution. Pandas is a Python Data Analysis Library that provides data structures and functions for data manipulation and analysis. It provides fast, flexible, and expressive data structures designed to make working with *labeled data* in an easy and intuitive manner. It is capable of handling huge amounts of data and at the same time it provides multiple ways to handle missing data thereby making data analysis more accurate and reliable.

2. *What do you understand by axes in a NumPy array ? Define axes for a 2d ndarray.*

Solution. In a Numpy array (also called *ndarray*), dimensions are called axes. The number of axes is called rank. A 2-dimensional ndarray has two corresponding axes :

- the first axis running vertically downwards across rows (axis 0), and
- the second axis running horizontally across columns (axis 1).

3. *How is Series data structure different from a dataframe data structure ?*

Solution. A Series is a one-dimensional object that can hold any data type such as integers, floats and strings. It has only one axis (maxis = 0).

A DataFrame is a two dimensional object that can have columns with potential different types. A datafrme can be created different kind of inputs which may include dictionaries, lists, series, and evén another DataFrame. Individual columns of a dataframe can act as separate Series objects.

It is the most commonly used pandas object. It has two axes (axis 0 and axis 1).

4. *Given are two objects, a list object namely* ***lst1*** *and a Series object namely* ***ser1****, both are having similar values i.e., 2, 4, 6, 8. . Find out the output produced by following statements:*

(i) `print(lst1 * 2)` (ii) `print(ser1 * 2)`

Solution. The output produced by given statement (i) will be :

```
[2, 4, 6, 8, 2, 4, 6, 8]
```

The output produced by given statement (ii) will be :

```
0    4
1    8
2   12
3   16
```

where 0, 1, 2, 3 are the indexes.

5. *Can you specify the reason behind the output produced by the code statements of previous question ?*

Solution. The reason behind the output produced by statement (i) is that :

When some scalar value is multiplied with a list object, then the list is replicated those many times, thus the result is having the list elements repeated twice.

The reason behind the output produced by statement (ii) is that :

When some scalar value is multiplied with a Series object, then the given operation gets performed on every single element of the Series because for a Series, the operations are carried in a vectorized way, i.e., applied on each element one by one.

6. *Given a Series object* **s13** *as shown below :*

```
A  7600
B  5600
C  7000
D  7000
dtype: int64
```

Why is following code producing Error while working on Series object **s13** *?*

```
import pandas as pd
s13.index = range(0, 5)
print(s13)
```

Solution. The given code is producing error because the s13 object has 4 elements only and the code line:

```
s13.index = range(0, 5)
```

is trying to assign 5 new indexes to it : **range(0, 5)** produces 5 values as **0, 1, 2, 3, 4.**

We can change the indexes of a Series object only if the new indexes are same in count as original and that is why the given code is producing an error.

7. *Correct the error reported by the code of previous question. Write corrected code for the same.*

Solution.

```
import pandas as pd
s13.index = range(1, 5)        #it will now produce 4 values : 1, 2, 3, 4
print(s13)
```

8. *Consider the below given two code fragments. Will they produce the same output ? Why/why not ?*

(*i*)
```
fst = [9, 10, 11]
ob1 = pd.Series(data = fst * 2)
print(ob1)
```

(*ii*)
```
fst = pd.Series(data =[9, 10, 11])
ob2 = pd.Series(data = fst * 2)
print(ob2)
```

Solution. No, both codes will produce different results.

This is because, in the part (*i*) Series object ob1's data contains **a Python list (fst)*2** which will repeat the values of the list two times and the data will contain : **9, 10, 11, 9, 10, 11.**

In part (*ii*), Series object ob2's data contains a **Series object (fst) *2**, which will perform the vectorized operation on the values of ***fst*** and then make it as data of ob2 , *i.e.*, **18, 20, 22.**

9. *What will be the output of the following program ?*

```
import pandas a pd
fst = [9, 10, 11]
scd = pd.Series(fst)
ob1 = pd.Series(data = fst * 2)
ob2 = pd.Series(data = scd * 2)
print("ob1")
print(ob1)
print("ob2")
print(ob2)
```


Solution.

```
ob1
0    9
1   10
2   11
3    9
4   10
5   11
ob2
0   18
1   20
2   22
```

10. *What will be the output of the following code ?*

```
import pandas as pd
import numpy as np
data = np.array(['a1','b1','c1','d1','e1','f1'])
s = pd.Series(data)
print("I.")
print (s[:3])
print("II.")
print(s[-3:])
```

Solution.
```
I.
0   a1
1   b1
2   c1
II.
3   d1
4   e1
5   f1
```

11. *What will be the output of the following code ?*

```
data = np.array(['a1','b1','c1','d1','e1'])
s = pd.Series(data,index = [1001,1002,1003,1004,1005])
# retrieve multiple elements with labels or index
print(s[[1002,1003,1004]])
```

Solution.
```
1002  b1
1003  c1
1004  d1
```

12. *Consider the following code to create two dataframes with similar values. What will be printed by the code given below ? Justify your answer.*

```
df1 = pd.DataFrame([1, 2, 3])
df2 = pd.DataFrame([[1, 2, 3]])
print("df1")
print(df1)
print("df2")
print(df2)
```

Solution. The output produced by above code will be as shown here.

```
df1
   0
0  1
1  2
2  3
df2
   0  1  2
0  1  2  3
```

The reason being that the first dataframe df1 is created with a 1D list hence all its values are taken as new row for dataframe df1 and only one column is created. **df1** is like the one shown on left below. Dataframe df2 , is created with a 2D list where inner list contains values 1,2, 3. Since inner list forms the column values, the three columns are formed from the three values of inner list; there is just one inner list, hence one row only.

Dataframe **df2** is like the one shown on the right\below :

	0
0	1
1	2
2	2

df1

	0	1	2
0	1	2	3

df2

13. *How does dataframe object specify indexes to its data rows ?*

Solution. If the indexes are not specified with the DataFrame( ), then these are generated from the input data in an intuitive fashion, for example, from the keys of dictionary (in case of column labels) or by using np.range(*n*) in the case of row labels, where *n* corresponds to the number of rows.

14. *Given a data frame namely* **data** *as shown in adjacent figure (fruit names are row labels). Write code statement to :*

```
       Color  Count  Price
Apple    Red      3    120
Apple  Green      9    110
 Pear    Red     25    125
 Pear  Green     26    150
 Lime  Green     99     70
```

(*a*) *Find all rows with the label "Apple". Extract all columns.*

(*b*) *List fruits with count more than 25.*

(*c*) *List single True or False to signify if all prices are more than 100 or not.*

(*d*) *List 2nd, 3rd and 4th rows.*

Solution.

(*a*) `data.loc['Apple', :]`

(*b*) `data[data['Count'] >25]`

(*c*) `(data['Price'] > 100).all( )`

(*d*) `data.iloc[0:3, :]`

15. *Consider the dataframe* **data** *given in Q. 14 above. Using the same dataframe* **data**, *answer the following:*

(*a*) *List only the columns Color and Price using* **loc.**

(*b*) *List only columns 0 and 2 (column indexes) using* **iloc.**

(*c*) *List only rows with labels 'Apple' and 'Pear' using* **loc.**

(*d*) *List only rows 1, 3, 4 using* **iloc.**

Solution.

(*a*) `data.loc[:, ['Color', 'Price']]`

(*b*) `data.iloc[:, [0, 2]]`

(*c*) `data.loc[['Apple', 'Pear']]`

(*d*) `data.iloc[[1, 3, 4]]`

16. *Consider the dataframe* **data** *given in Q. 14 above. Using the same dataframe data, answer the following :*

(*a*) *Write statement to delete rows with labels 'Apple' and 'Lime'.*

(*b*) *Write statement to delete columns with labels 'Color' and 'Count'.*

Solution.

(*a*) `data.drop(['Apple', 'Lime'] )`

(*b*) `data.drop(['Color', 'Count'], axis = 1)`

17. *Given a DataFrame* **mdf** *as shown below :*

```
   A  B  C
0  1  2  3
1  4  5  6
```

What will be the output produced by the following code ?

```
print("I  :", mdf.iloc[0][0])
print("II :", mdf.loc[0]['C'])
print("III :", mdf.at[1,'A'])
print("IV :", mdf.iat[1,2])
```

Solution.

```
I   : 1
II  : 3
III : 4
IV  : 6
```

18. *What is following statement doing ?*

```
df.drop(["Total", "Order"], axis = 1)
```

Solution. The given statement is deleting the columns 'Total' and 'Order' from the DataFrame df.

19. *Given a DataFrame* **mdf** *as shown below :*

```
   A  B  C
0  1  2  3
1  4  5  6
```

Find out the errors in following statements.

(*i*) `mdf.drop(["Total", "Order"], axis = 1)`

(*ii*) `mdf.drop(["A", "D"])`

(*iii*) `mdf.drop(["A", "D"], axis = 1)`

Solution.

(*i*) Axis 1 means columns. There are no columns by the name "Total", and "order" in DataFrame **mdf**. Hence the error.

(*ii*) No axis given, which means the default axis 0, which is row. There are no rows in DataFrame **mdf** with labels as "A" and "D". Hence the error.

(*iii*) Axis 1 means columns. There are no columns by the name "D" in DataFrame **mdf**. Hence the error.

20. *Explain what the following statements are doing ?* ***df*** *is the name of a DataFrame.*

(*i*) `df.iloc[:5,]`

(*ii*) `df.iloc[1:5,]`

(*iii*) `df.iloc[5,0]`

(*iv*) `df.iloc[1:5,0]`

(*v*) `df.iloc[1:5,:5]`

(*vi*) `df.iloc[2:7,1:3]`

Solution.

(*i*) It will display first 5 rows of DataFrame **df.**

(*ii*) It will display Second (which has index as 1) to Fifth row of DataFrame **df.**

(iii) It will display Sixth row's 1st column value of DataFrame **df**.

(iv) It will display Second to Fifth rows', first column's values of DataFrame **df**.

(v) It will display Second to Fifth rows', first 5 columns' values of DataFrame **df**.

(vi) It will display Third to Seventh rows', 2nd and 3rd columns' values of DataFrame **df**.

21. *Trying to extract the first five rows of DataFrame x, Nia has given code as :*

```
x.loc[0:5]
```

But it is returning 6 rows. Why ? Suggest the solution.

Solution. It is returning 6 rows because **loc** does not produce output like slices. It includes both starting and end point. To correct the above problem, Nia needs to change the code to :

```
x.loc[0:4]
```

22. *What is following statement doing ?*

```
df.drop(df.columns[0], axis = 1)
```

Solution. Axis 1 means columns. Given statement will remove the first column because of **df.columns[0]**.

23. *What is the output of the following code ?*

```
import pandas as pd
import numpy as np
x = pd.DataFrame({"var1" : np.arange(1,20,2)}, index=[9,8,7,6,10, 1, 2, 3, 4, 5])
x1 = x.iloc[4:7]
x2 = x.head(2)
x3= x.tail(4)
print("x1:")
print(x1)
print("x2:")
print(x2)
print("x3:")
print(x3)
```

Solution.

```
x1:
    var1
10     9
1     11
2     13

x2:
   var1
9     1
8     3

x3:
   var1
2    13
3    15
4    17
5    19
```


24. *What will the following code do ?*

```
x = pd.DataFrame({"var1" : np.arange(1,20,2)}, index=[9,8,7,6,10, 1, 2, 3, 4, 5])
x1 = x.iloc[4:7]
x2 = x.head(2)
x3= x.tail(4)
```

Solution. The given code is creating a DataFrame x from the given data and indexes. (*First line of code*)

Second line of code is creating a DataFrame x1 which will store 4, 5, 6 rows of x.

Third line of code is creating a DataFrame x2 which will store first 2 rows of x.

Fourth line of code is creating a DataFrame x3 which will store last 4 rows of x.

25. *Why is the following code not renaming the index and columns even when code is executing without any error, considering that the saleDf dataframe is as shown on the right.*

```
>>> saleDf
       Target  Sales
zoneA   56000  58000
zoneB   70000  68000
zoneC   75000  78000
zoneD   60000  61000
```

The code :

```
saleDf.rename(index = {'zoneC':'Central', 'zoneD':'Dakshin'},\
      columns = {'Target':'Targeted', 'Sales':'Achieved'} )
print( saleDf )
```

What output would be produced by the above code and what is the problem with the code?

Solution. The above code will produce the output as :

```
       Target  Sales
zoneA   56000  58000
zoneB   70000  68000
zoneC   75000  78000
zoneD   60000  61000
```

which means the given data frame is shown unchanged.

There is no syntax error in the code and internally the line 1 of the code does rename the indexes and columns but since **inplace** argument is not given , it made the changes in a new dataframe and the original dataframe remained unchanged . So when the line 2 printed the original dataframe **saleDf**, it showed the unchanged dataframe.

26. *What correction/modification would you suggest for the problem of the previous question which wanted to change the index and column name in the original data frame saleDf ?*

Solution. To change the index and columns' names in the original data frame, we must give argument **inplace = True** in the **rename( )** method. Thus, the modified code of the previous question should be :

```
saleDf.rename(index = {'zoneC':'Central', 'zoneD':'Dakshin'}, \
              columns = {'Target': 'Targeted', 'Sales' : 'Achieved'}, inplace = True)
print( saleDf )
```

27. *Consider the following code*

```
Section = ['A', 'B', 'C']
Classes = [ 6, 4, 3]
dc = {'Section':Section, 'No. of Classes': Classes}
clasDf = pd.DataFrame(dc, index = ['True', 'False', 'True'] )
print(clasdf.loc[True])
```

(i) *Why is it giving KeyError if you run the given code ?*

(ii) *Suggest the correction /solution for the above problem.*

Solution.

(i) The above code is giving **KeyError** because it has created the dataframe with string indexes (values 'True' and 'False' are enclosed in quotes) and it is trying to access rows considering the indexes as Boolean.

(ii) The solution for the above problem is to remove the quotes from the indexes while creating the dataframe, *i.e.*, 4th line of the code should be changed to :

```
clasDf = pd.DataFrame(dc, index = [True, False, True] )
```

Not enclosed in quotes this time

Practical Questions

28. *Given a Series that stores the area of some states in km^2. Write code to find out the biggest and smallest three areas from the given Series. Given series has been created like this :*

```
Ser1 = pd.Series( [34567, 890, 450, 67892, 34677, 78902, 256711, 678291,
                   637632, 25723, 2367, 11789, 345, 256517])
```

Solution.

```
import pandas as pd
Ser1 = pd.Series( [34567, 890, 450, 67892, 34677, 78902,
                   256711, 678291, 637632, 25723,
                   2367, 11789, 345, 256517])
print("Top 3 biggest areas are :")
print(Ser1.sort_values().tail(3))
print("3 smallest areas are :")
print(Ser1.sort_values().head(3))
```

```
3 biggest areas are :
7    678291
8    637632
6    256711
dtype: int64
3 smallest areas are :
1      890
2      450
12     345
dtype: int64
```

An alternative code for above problem would be :

```
import pandas as pd
Ser1 = pd.Series( [34567, 890, 450, 67892, 34677, 78902, 256711, 678291, 637632,
                   25723, 2367, 11789, 345, 256517])
print("Top 3 biggest areas are :")
print(Ser1.sort_values(ascending = False).head(3))
print("3 smallest areas are :")
print(Ser1.sort_values(ascending = False).tail(3))
```

29. *From the series Ser1 of areas (given earlier that stores areas of states in km^2), find out the areas that are more than 50000 km^2.*

Solution.

```
import pandas as pd
Ser1 = pd.Series([34567, 890, 450, 67892, 34677, 78902,
256711, 678291, 637632, 25723, 2367, 11789, 345, 256517])
print(Ser1[Ser1 > 50000])
```

```
In [90]: print(Ser1[Ser1>50000])
3     67892
5     78902
6    256711
7    678291
8    637632
13   256517
dtype: int64
```


30. *Write a program to create a Series object with 6 random integers and having indexes as : ['p', 'q', 'r', 'n', 't','v']*

Solution.

```
import pandas as pd
import numpy as np
s = pd.Series(np.random.randint(6), index =['p', 'q', 'r', 'n', 't','v'])
print(s)
```

31. *Write a program to create data series and then change the indexes of the Series object in any random order.*

Solution.

```
import pandas as pd
import numpy as np
s1 = pd.Series(data = [100,200,300,400,500], index = ['I', 'J', 'K','L','M'])
print("Original Data Series:")
print(s)
s1 = s1.reindex(index = ['K','I','M','L','J'])
print("Data Series after changing the order of index:")
print(s)
```

32. *Write a program to sort the values of a Series object s1 in ascending order of its values and store it into series object s2.*

Solution.

```
import pandas as pd
:    # Series object s1 created or loaded
s2 = s1.sort_values()
print("Series object s1:")
print(s1)
print("Series object s2:")
print(s2)
```

Output

```
Series object s1:
1    6700
2    5600
3    5000
4    5200
dtype: int64
Series object s2:
3    5000
4    5200
2    5600
1    6700
dtype: int64
```

33. *Write a program to sort the values of a Series object s1 in descending order of its indexes and store it into series object s3.*

Solution.

```
import pandas as pd
:    # Series object s1 created or loaded
s3 = s1.sort_index(ascending = False)
print("Series object s1:")
print(s1)
print("Series object s3:")
print(s3)
```

Output

```
Series object s1:
1    6700
2    5600
3    5000
4    5200
dtype: int64
Series object s3:
4    5200
3    5000
2    5600
1    6700
dtype: int64
```

34. *Given a Series object s4. Write a program to change the values at its 2nd row(index1) and 3rd row to 8000.*

Solution.

```
import pandas as pd
:  # Series object s4 created or loaded
print("Original Series object s4:")
print(s4)
s4[1:3]=8000
print("Series object s4 after changing value:")
print(s4)
```

Output

```
Original Series object s4:
1    6700
2    5600
3    5000
4    5200
dtype: int64
Series object s4 after changing value:
1    6700
2    8000
3    8000
4    5200
dtype: int64
```

35. *Given a Series object s5. Write a program to calculate the cubes of the Series values.*

Solution.

```
import pandas as pd
:  # Series object s4 created or loaded
print("Series object s5:")
print(s5)
print("Cubes of s5 values:")
print(s5 ** 3)
```

Output

```
Series object s5:
0    5
1    7
2    9
dtype: int64
Cubes of s5 values:
0    125
1    343
2    729
dtype: int64
```

36. *Given a Series object s5. Write a program to store the squares of the Series values in object s6. Display s6's values which are > 15.*

Solution.

```
import pandas as pd
:  # Series object s5 created or loaded
print("Series object s5 :")
print(s5)
s6 = s5 * 2                # s6 created
print("Values in s6 > 15 :")
print(s6[ s6>15])
```


37. *Write a program to display number of rows and number of columns in DataFrame df.*

Solution.

```
import pandas as pd
:  # Series object df created or loaded
print("No. Of rows : ", df.shape[0])
print("No. Of columns : ", df.shape[1])
```

38. *Write a program to display number of rows and number of columns in DataFrame df without using shape attribute.*

Solution.

```
import pandas as pd
:  # Series object df created or loaded
rows = len(df.axes[0])
cols = len(df.axes[1])
print("No. of rows:", rows)
print("No. of columns: ", cols)
```

39. *Given a DataFrame df :*

	Age	Name	Weight
0	15	Arnav	42
1	22	Charles	75
2	35	Guru	66

Write a program to display only the Weight of first and third rows.

Solution.

```
import pandas as pd
:  # Series object df created or loaded
print(df.iloc[ [0,2], [2] ])
```

Output

```
   Weight
0      42
2      66
```

GLOSSARY

Series	Pandas data structure storing one-dimensional array of elements.
DataFrame	Pandas data structure storing two-dimensional labelled array.

Assignment

For selective assignment solutions

Scan QR Code

Type A : Very Short Answer Questions

1. What is the significance of Pandas library ?
2. Name some common data structures of Python's Pandas library.
3. How is a Series object different from and similar to ndarrays ? Support your answer with examples.

4. Write commands to print following details of a Series object seal.
   (*a*) if the series is empty (*b*) indexes of the series
   (*c*) The data type of underlying data (*d*) if the series stores any NaN values
5. Given following Series objects :

S1	
0	3
1	5
2	6
4	10
5	12

S2	
0	12
2	10
3	15
4	20
6	27

   (*a*) What will be the result of S1 + S2 ?
   (*b*) What will be the result of S1 – S2 ?
6. Given a dataframe df as shown below :

	A	B	D
0	15	17	19
1	16	18	20
2	20	21	22

   What will be the result of following code statements ?
   (*a*) `df['C'] = np.NaN` (*b*) `df['C'] = [2, 5]` (*c*) `df['C'] = [12, 15, 27]`
7. Write code statements to list the following, from a dataframe namely sales.
   (*a*) List only columns 'Item' and 'Revenue'.
   (*b*) List rows from 3 to 7.
   (*c*) List the value of cell in 5th row, 'Item' column.
8. How would you add a new column namely 'val' to a dataframe df that has 10 rows in it and has columns as 'Item', 'Qty', 'Price' ? You can choose to put any values of your choice.
9. Write code statements for a dataframe df for the following :
   (*a*) delete an existing column from it.
   (*b*) delete rows from 3 to 6 from it.
   (*c*) Check if the dataframe has any missing values.
   (*d*) fill all missing values with 999 in it.
10. Write statement(s) to delete a row from a DataFrame.
11. Write statement(s) to delete a column from a DataFrame.
12. Write statement(s) to change the value at 5th row, 6th column in a DataFrame df.
13. Write statement(s) to change the values to 750 at 4th row to 9th row, 7th column in a DataFrame df.
14. What is the difference between **iloc** and **loc** with respect to a DataFrame ?
15. What is the difference between **iat** and **at** with respect to a DataFrame ?
16. How would you delete columns from a dataframe ?
17. How would you delete rows from a dataframe ?
18. Which function would you use to rename the index/column names in a dataframe ?

Type B : Short Answer Questions/Conceptual Questions

NOTE : For the questions here, consider that both Pandas and NumPy modules have been imported as :

```
import pandas as pd
import numpy as np
```

1. Consider following Series object namely S :

```
0    0.430271
1    0.617328
2   -0.265421
3   -0.836113
dtype:float64
```

   What will be returned by following statements ?
   (*a*) S * 100 (*b*) S > 0 (*c*) S1 = pd.Series(S) (*d*) S3 = pd.Series(S1) + 3
   What will be the values of Series objects S1 and S2 created above ?
2. Consider the same Series object, S, given in the previous question. What output will be produced by following code fragment ?

```
S.index = ['AMZN', 'AAPL', 'MSFT', 'GOOG']
print(S)
print(s['AMZN'])
s['AMZN'] = 1.5
print( s['AMZN'])
print(S)
```

3. What will be the output produced by following code ?

```
Stationery = ['pencils', 'notebooks', 'scales', 'erasers']
S = pd.Series([20, 33, 52, 10], index = Stationery)
S2 = pd.Series([17, 13, 31, 32], index = Stationery)
print (S + S2)
S = S + S2
print(S + S2)
```

4. What will be the output produced by following code, considering the Series object S given above ?
   (*a*) `print(S[1:1])` (*b*) `print(S[0:1])` (*c*) `print(S[0:2])`
   (*d*) `S[0:2] = 12` `print(S)` (*e*) `print(S.index)` `print(S.values)`
5. Find the error in following code fragment :
   (*a*)
```
S2 = pd.Series( [101, 102, 102, 104] )
print(S2.index)
S2.index = [0, 1, 2, 3, 4, 5]
S2[5] = 220
print(S2)
```
   (*b*) `S = pd.Series(2, 3, 4, 5, index = range (4) )`
   (*c*) `S1 = pd.Series(1, 2, 3, 4, index = range (7) )`
   (*d*) `S2 = pd.Series([1, 2, 3, 4], index = range (4) )`
6. Find the Error :

```
data = np.array(['a','b','c','d','e','f'])
s = pd.Series(data, index = [100,101,102,103,104,105])
print(s[102,103,104])
```

   Can you correct the error ?

7. Why does following code cause error ?

```
s1 = pd.Series(range(1, 15, 3), index = list('abcd'))
```

8. Why does following code cause error ?

```
s1 = pd.Series(range(1, 15, 3), index = list('ababa'))
print(s1['ab'])
```

9. If **Ser** is a Series type object having 30 values, then how are statements (*a*), (*b*) and (*c*), (*d*) similar and different ?

   (*a*) print(Ser.head( )) (*b*) print(Ser.head(8))

   (*c*) print(Ser.tail( )) (*d*) print(Ser.tail(11))

10. What advantages does dataframe offer over series datastructure ? If you have similar data stored in multiple series and a single dataframe, which one would you prefer and why ?

11. Given :

```
import pandas as pd
d = {'one' : pd.series([1., 2., 3.], index = ['a', 'b', 'c']),
     'two' : pd.series([1., 2., 3., 4.], index = ['a', 'b', 'c', 'd'])}
df = pd.Dataframe(d)
df1 = pd.Dataframe(d, index = ['d', 'b', 'a'])
df2 = pd.Dataframe(d, index = ['d', 'a'], columns = ['two', 'three'])
print(df)
print(df1)
print(df2)
```

What will Python show the result as if you execute above code ?

12. From the dataframes created in previous question, write code to

   (*a*) display only column 'a' from dataframes df, df1, and df2.

   (*b*) display only rows 0 and 1 from dataframes df, df1, and df2.

   (*c*) display only column 'a' and 'b' for rows 1 and 2 from dataframes df, df1 and df2.

   (*d*) add an empty column 'x' to all dataframes

Questions 13 and 14 make use of this dictionary :

```
my_di = {"name" : ["Jiya", "Tim", "Rohan"],"
         "age" : np.array([10,15,20]),
         "weight" : (75,123,239),
         "height" :[4.5, 5, 6.1],
         "siblings" : 1,
         "gender" : "M"}
```

13. *Predict the output of following code* (it uses above given dictionary **my_di**)

```
df = pd.DataFrame(my_di)
print(df)
```

14. *Consider the same dictionary* **my_di** *in the previous question, what will be the output produced by following code ?*

```
df2 = pd.DataFrame(my_di, index = my_di ["name"] )
print(df2)
```


Using the **df2** created in **Question 14**, answer the outputs of questions (**15 to 19**) given below.

15. Assume that required libraries (panda and numpy) are imported and dataframe df2 has been created as per questions 13 and 14 above. Predict the output of following code fragment :

```
print( df2["weight"] )
print( df2.weight['Tim' ] )
```

16. Assume that required libraries (panda and numpy) are imported and dataframe df2 has been created as per questions 13 and 14 above. Predict the output of following code fragment :

```
df2["IQ"] = [130, 105, 115]
df2["Married"] = False
print ( df2 )
```

17. Assume that required libraries (panda and numpy) are imported and dataframe df2 has been created as per questions 13 and 14 above. Predict the output produced by following code fragment :

```
df2["College"] = pd.Series(["IIT"], index=["Rohan"] )
print ( df2 )
```

18. Assume that required libraries (panda and numpy) are imported and dataframe df2 has been created as per questions 13 and 14 above. Predict the output produced by following code fragment :

```
print ( df2.loc["Jiva"] )
print ( df2.loc["Jiva","IQ"] )
print ( df2.loc["Jiva":"Tim" , "IQ":"College"] )
print ( df2.iloc[0] )
print ( df2.iloc[0, 5] )
print ( df2f2.ilocloc[0:2, 5:8] )
```

19. What is the output of the following code ?

```
d = {'col1': [1, 4, 3 ], 'col2': [6, 7, 8], 'col3': [ 9, 0, 1]}
df = pd.DataFrame(d)
print("Original DataFrame")
print(df)
print("New DataFrame :")
dfn = df.drop(df.index[[1,2]])
print(dfn)
```

20. What is the output of the following code ?

```
data = {'age': [20, 23, 22], 'name': ['Ruhi', 'Ali', 'Sam']}
df1 = pd.DataFrame(data, index=[1, 2, 3])
print("Before")
print(df1)
df1['Edu'] = ['BA', 'BE', 'MBA']
print('After')
print(df1)
```

21. Find the error in the following code? Suggest the solution.

```
>>> topDf
       Rollno Name  Marks
Sec A    115  Pavni  97.5
Sec B    236  Rishi  98.0
Sec C    307  Preet  98.5
Sec D    422  Paula  98.0
>>> topDf.del['Sec D']
```

22. Find the error in the following code considering the same dataframe **topDf** given in the previous question.
    (*i*) `topDf.rename(index=['a','b','c','d'])`
    (*ii*) `topDf.rename(columns = {})`

Type C : Long Answer Questions

1. Write Python code to create a Series object ***Temp1*** that stores temperatures of seven days in it. Take any random seven temperatures.
2. Write Python code to create a Series object ***Temp2*** storing temperatures of seven days of week. Its indexes should be 'Sunday', 'Monday',... 'Saturday'.
3. A series object (say T1) stores the average temperature recorded on each day of a month. Write code to display the temperatures recorded on : (*i*) first 7 days (*ii*) last 7 days.
4. Series objects *Temp1, Temp2, Temp3, Temp4* store the temperatures of days of week1, week2, week3, week4 respectively. Write a script to
   (*a*) print the average temperature per week. (*b*) print average temperature of entire month.
5. Write a program that stores the sales of 5 fast moving items of a store for each month in 12 Series objects, *i.e.*, S1 Series object stores sales of these 5 items in 1st month, S2 stores sales of these 5 items in 2nd month, and so on.
   The program should display the summary sales report like this :
   Total Yearly Sales, item-wise (should display sum of items' sales over the months)
   Maximum sales of item made : <name of item that was sold the maximum in whole year>
   Maximum sales for individual items
   Maximum sales of item 1 made : <month in which that item sold the maximum>
   Maximum sales of item 2 made : <month in which that item sold the maximum>
   Maximum sales of item 3 made : <month in which that item sold the maximum>
   Maximum sales of item 4 made : <month in which that item sold the maximum>
   Maximum sales of item 5 made : <month in which that item sold the maximum>
6. Three Series objects store the marks of 10 students in three terms. Roll numbers of students form the index of these Series objects. The Three Series objects have the same indexes.
   Calculate the total weighted marks obtained by students as per following formula :
   Final marks = 25% Term 1 + 25% Term 2 + 50% Term 3
   Store the Final marks of students in another Series object.
7. Write code to print all the information about a Series object.
8. Write a program to create three different Series objects from the three columns of a DataFrame df.
9. Write a program to create three different Series objects from the three rows of a DataFrame df.
10. Write a program to create a Series object from an ndarray that stores characters from '*a*' to '*g*'.
11. Write a program to create a Series object that stores the table of number 5.
12. Write a program to create a Dataframe that stores two columns, which store the Series objects of the previous **two questions** (**10** and **11**).
13. Write a program to create a Dataframe storing salesmen details (*name, zone, sales*) of five salesmen.
14. Four dictionaries store the details of four employees-of-the-month as (*empno, name*). Write a program to create a dataframe from these.
15. A list stores three dictionaries each storing details, (*old price, new price, change*). Write a program to create a dataframe from it.

2 Python Pandas – II

Outline

2.1 Introduction
2.2 Iterating Over a DataFrame
2.3 Binary Operations in a DataFrame
2.4 Descriptive Statistics with Pandas
2.5 Some Other Essential Functions and Functionality
2.6 Advanced Operations on DataFrame
2.7 Handling Missing Data
2.8 Combining DataFrames
2.9 Function groupby( )

2.1 Introduction

By now you have an idea of Python Pandas' two most popular data structures **Series** and **DataFrames**. In previous chapter, you learnt to work with Series object and how to create dataframes in many ways. Recall that Series is a 1D datastructure in Pandas whereas DataFrame is a 2D datastructure.

Series

	Apples
0	3
1	2
2	0
3	1

Series

	Oranges
0	0
1	3
2	7
3	2

DataFrame

	Apples	Oranges
0	3	0
1	2	3
2	0	7
3	1	2

In this chapter, we shall talk more about dataframes, basic operations of dataframe, descriptive statistics, pivoting, handling missing data, combining/merging etc. So, let's get started.

2.2 Iterating Over a DataFrame

Sometimes, you need to process all the data values of a dataframe. In such a case, writing separate statements for assigning/accessing individual values will make the process cumbersome. Thus, for such a thing, you need to iterate over a dataframe. There are many ways using which you can iterate over a dataframe. Most common of which are **<DFobject>.iterrows( )** and **<DFobject>.iteritems( )**. Let us see how these function.

The method **<DF>.iterrows( )** views a dataframe in the form of ***horizontal subsets*** *i.e.*, row-wise and the method **<DF>.iteritems( )** views a dataframe in the form of ***vertical subsets*** *i.e.*, column-wise (as shown in Fig 2.1). Each horizontal subset is in the form of **(row-index, Series)** where Series contains all column values for that row-index ; Each vertical subset is in the form of **(col-index, Series)** where Series contains all row values for that column-index.

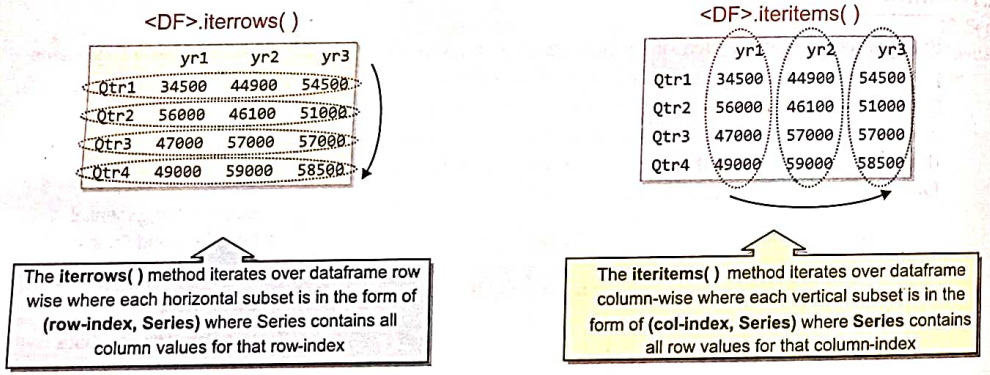


Figure 2.1 Iteration in a dataframe

2.2.1 Using pandas.iterrows( ) Function

Following examples demonstrate the usage of function iterrows( ) practically using the same dataframe (**df1**) shown in Fig. 2.1 above.

NOTE

<DF>.iteritems( ) iterates over vertical subsets in the form of (col-index, Series) pairs and **<DF>.iterrows( )** iterates over horizontal subsets in the form of (row-index, Series) pairs.

EXAMPLE 1 *Using iterrows( ) to extract data from dataframe row wise :*

```
import pandas as pd
diSales = { 'yr1': {'Qtr1': 34500, 'Qtr2': 56000, 'Qtr3': 47000, 'Qtr4': 49000}, \
'yr2': {'Qtr1': 44900, 'Qtr2': 46100, 'Qtr3': 57000, 'Qtr4': 59000}, \
'yr3': {'Qtr1': 54500, 'Qtr2': 51000, 'Qtr3': 57000, 'Qtr4': 58500} }
df1 = pd.DataFrame(diSales)
```

Command continuation marks


```
for (row, rowSeries) in df1.iterrows() :
    print("Row index :", row)
    print("Containing :")
    print(rowSeries)
```

*Each row is taken one at a time in the form of **(row, rowSeries)** where **row** would store the row-index and **rowSeries** will store all the values of the row in form of a Series object*

Row-index

All values in a row are in form of a Series object

To see working of iterrows in action

Scan QR Code

The output produced by above code is :

```
Row index :  Qtr1
Containing :
yr1    34500
yr2    44900
yr3    54500
Name: Qtr1, dtype: int64
Row index :  Qtr2
Containing :
yr1    56000
yr2    46100
yr3    51000
Name: Qtr2, dtype: int64
Row index :  Qtr3
Containing :
yr1    47000
yr2    57000
yr3    57000
Name: Qtr3, dtype: int64
Row index :  Qtr4
Containing :
yr1    49000
yr2    59000
yr3    58500
Name: Qtr4, dtype: int64
```

*Row-index **Qtr1***

*All values in rowindex **Qtr1** in form of a Series object*

*Row-index **Qtr2***

*All values in rowindex **Qtr2** in form of a Series object*

*Row-index **Qtr3***

*All values in rowindex **Qtr3** in form of a Series object*

*Row-index **Qtr4***

*All values in rowindex **Qtr4** in form of a Series object*

So, with **iterrows( )**, you can iterate over dataframe row-wise where each row's values are returned in form of a Series type object. You can also iterate over a Series object just as you iterate over other sequences. Following code is iterating over the rows' Series objects too.

EXAMPLE 2 *Using iterrows( ) to extract row-wise Series objects :*

```
import pandas as pd
diSales = {'yr1': {'Qtr1': 34500, 'Qtr2': 56000, 'Qtr3': 47000, 'Qtr4': 49000}, \
'yr2': {'Qtr1': 44900, 'Qtr2': 46100, 'Qtr3': 57000, 'Qtr4': 59000}, \
'yr3': {'Qtr1': 54500, 'Qtr2': 51000, 'Qtr3': 57000, 'Qtr4': 58500} }
df1 = pd.DataFrame(diSales)
```

Command continuation marks

```
for (row, rowSeries) in df1.iterrows() :
    print("Row index :", row)      # row-index of each row : Stment 1
    print("Containing :")
    i = 0
    for val in rowSeries :
        print("At", i, "position :", val)
        i = i + 1
```

See, we can iterate over the Series object obtained from iterrows()

The output produced by above code is :

```
Row index : Qtr1          <-- Result of Stment1 of code above
Containing :
At 0 position : 34500
At 1 position : 44900
At 2 position : 54500
```

Output of inner loop (row 1)

```
Row index : Qtr2          <-- Result of Stment2 of code above
Containing :
At 0 position : 56000
At 1 position : 46100
At 2 position : 51000
```

Output of inner loop (row 2)

```
Row index : Qtr3          <-- Result of Stment3 of code above
Containing :
At 0 position : 47000
At 1 position : 57000
At 2 position : 57000
```

Output of inner loop (row 3)

```
Row index : Qtr4          <-- Result of Stment4 of code above
Containing :
At 0 position : 49000
At 1 position : 59000
At 2 position : 58500
```

Output of inner loop (row 4)

You may also print the ***column names*** which act as the indexes for each row-wise Series object (*e.g.*, **rowSeries** in above code). In other words, you can use **row Series.index** to print individual column names along with row data.

The statement **print(rowSeries.index)**, if used with above code, will print :

```
Index[['yr1', 'yr2', 'yr3']]
```

which are the indexes of the object stored in **rowSeries**.

2.2.2 Using pandas.iteritems( ) Function

Unlike iterrows( ), ***iteritems( )*** brings you vertical subsets from a dataframe in the form of ***column index*** **and a** ***series object*** containing values for all rows in that column. Following examples illustrate the practical use of **iteritems( )**.

EXAMPLE 3 *Using iteritems( ) to extract data from dataframe column wise :*

```
import pandas as pd
diSales = {'yr1': {'Qtr1': 34500, 'Qtr2': 56000, 'Qtr3': 47000, 'Qtr4': 49000}, \
'yr2': {'Qtr1': 44900, 'Qtr2': 46100, 'Qtr3': 57000, 'Qtr4': 59000}, \
'yr3': {'Qtr1': 54500, 'Qtr2': 51000, 'Qtr3': 57000, 'Qtr4': 58500} }
df1 = pd.DataFrame(diSales)
for (col, colSeries) in df1.iteritems() :
    print("Column index : ", col)
    print("Containing : ")
    print(colSeries)
```

Command continuation marks

All values in a column are in form of a Series object

Col-index

Each row is taken one at a time in the form of ***(col, colSeries)*** *where* ***col*** *would store the row-index and* ***colSeries*** *will store all the values of the row in form of a Series object*

```
Column index : yr1          <-- Col-index yr1
Containing :
Qtr1    34500
Qtr2    56000
Qtr3    47000
Qtr4    49000               <-- All values in col index yr1 are in form of a Series object
Name: yr1, dtype: int64

Column index : yr2          <-- Col-index yr2
Containing :
Qtr1    44900
Qtr2    46100
Qtr3    57000
Qtr4    59000               <-- All values in col index yr2 are in form of a Series object
Name: yr2, dtype: int64

Column index : yr3          <-- Col-index yr3
Containing :
Qtr1    54500
Qtr2    51000
Qtr3    57000
Qtr4    58500               <-- All values in col index yr3 are in form of a Series object
Name: yr3, dtype: int64
```

To see working of iteritems in action

Scan QR Code

EXAMPLE 4 *Using iteritems( ) to extract column wise Series objects :*

```
import pandas as pd
diSales = { 'yr1': {'Qtr1': 34500, 'Qtr2': 56000, 'Qtr3': 47000, 'Qtr4': 49000}, \
'yr2': {'Qtr1': 44900, 'Qtr2': 46100, 'Qtr3': 57000, 'Qtr4': 59000}, \
'yr3': {'Qtr1': 54500, 'Qtr2': 51000, 'Qtr3': 57000, 'Qtr4': 58500} }
df1 = pd.DataFrame(diSales)
for (col, colSeries) in df1.iteritems() :
    print("Column index :", col)
    print("Containing :")
    print(colSeries)
```

Command continuation marks

See, we can iterate over the Series object obtained from iteritems()

The output produced by above code is :

```
Column index :  yr1
Containing :
At row 0  :  34500
At row 1  :  56000
At row 2  :  47000
At row 3  :  49000
```

Result of ***Stment1*** *of code above*

Output of inner loop (col 1)

```
Column index :  yr2
Containing :
At row 0  :  44900
At row 1  :  46100
At row 2  :  57000
At row 3  :  59000
```

Result of ***Stment1*** *of code above*

Output of inner loop (col 2)

```
Column index : yr3
Containing :
At row 0  :  54500
At row 1  :  51000
At row 2  :  57000
At row 3  :  58500
```

Result of ***Stment1*** *of code above*

Output of inner loop (col 3)

You may also print the **row labels** which act as the indexes for each column-wise Series object (*e.g.*, **colSeries** in above code). In other words, you can use **colSeries.index** to print individual row labels along with column's data. **colSeries.index** will print **['Qtr1', 'Qtr2', 'Qtr3', 'Qtr4']**.

Consider some more examples given below, which will iterate over a DataFrame **df** as shown below :

	Marks	Name
Rno.1	70	Ram
Rno.2	95	Pam
Rno.3	80	Sam

EXAMPLE 5 *Write a program to print the DataFrame* ***df****, one row at a time.*

SOLUTION

```
import pandas as pd
dict = {'Name':["Ram", "Pam", "Sam"],
        'Marks':[70, 95, 80]}
df = pd.DataFrame(dict, index = ['Rno.1', 'Rno.2', 'Rno.3'])
for i, j in df.iterrows():
    print(j)
    print("----------")
```

Output

```
Marks  70
Name   Ram
Name:  Rno.1, dtype: object
----------
Marks  95
Name   Pam
Name:  Rno.2, dtype: object
----------
Marks  80
Name   Sam
Name:  Rno.3, dtype: object
----------
```


EXAMPLE 6 *Write a program to print the DataFrame* ***df****, one column at a time.*

SOLUTION

```
import pandas as pd
dict = { 'Name':["Ram", "Pam", "Sam"],
         'Marks':[70, 95, 80]}
df = pd.DataFrame(dict, index = ['Rno.1', 'Rno.2', 'Rno.3'])
for i, j in df.iteritems():
    print(j)
    print("----------")
```

Output

```
Rno.1   70
Rno.2   95
Rno.3   80
Name: Marks, dtype: int64
----------
Rno.1   Ram
Rno.2   Pam
Rno.3   Sam
Name: Name, dtype: object
----------
```

Printing Individual Column from a Row

When accessing rows of a DataFrame using *iterrows()*, then by using the *rowSeries[<column>]*, you can print individual column value from that row, *i.e.*, after the line :

```
for r, Row in df.iterrows():
```

you can print individual column value as :

```
Row[<column name>]
```

EXAMPLE 7 *Write a program to print only the values from marks column, for each row.*

SOLUTION

```
import pandas as pd
dict = {'Name':["Ram", "Pam", "Sam"],
        'Marks':[70, 95, 80]}
df = pd.DataFrame(dict, index = ['Rno.1', 'Rno.2', 'Rno.3'])
for r, row in df.iterrows():
    print(row['Marks'])
    print("----")
```

Output

```
70
----
95
----
80
----
```

2.3 Binary Operations in a DataFrame

Binary operations mean operations requiring two values to perform and these values are picked elementwise. In a binary operation, the data from the two dataframes are aligned on the bases of their row and column indexes and for the matching row, column index, the given operation is performed and for the nonmatching row, column indexes NaN value is stored in the result (just as the way **data alignment** happens in Series objects, refer section 2.4.3, Chapter 2).

So you can say that like Series objects, data is aligned in two dataframes, the data is aligned on the basis of matching row and column indexes and then arithmetic is performed for non-overlapping indexes, the arithmetic operations result as a NaN for non-matching indexes.

In order to understand binary operations on DataFrames, consider the following DataFrames (**df1, df2, df3, df4**) on which we shall perform various binary operations like *addition, subtraction, multiplication* and *division*.

To see operations on dataframe in action

Scan QR Code

DataFrames (df1, df2, df3, df4)

```
In [50]: df1
Out[50]:
   A  B  C
0  1  2  3
1  4  5  6
2  7  8  9
```

```
In [51]: df2
Out[51]:
    A   B   C
0  10  20  30
1  40  50  60
2  70  80  90
```

```
In [52]: df3
Out[52]:
     A    B    C
0  100  200  300
1  400  500  600
```

```
In [53]: df4
Out[53]:
      A     B
0  1000  2000
1  4000  5000
2  7000  8000
```

Addition Binary operation [using +, add( ) and radd( )]

You can perform **add** binary operation on two dataframe objects using either + operator or using add( ) as per syntax : **<DF1>.add(<DF2>)** which means <DF1>+<DF2> or by using radd( ) *i.e., reverse add* as per syntax : **<DF1>.radd(<DF2>)** which means <DF2>+<DF1>

```
>>> df1 + df2
    A   B   C
0  11  22  33
1  44  55  66
2  77  88  99
   ≡
>>> df1.add(df2)
    A   B   C
0  11  22  33
1  44  55  66
2  77  88  99
   ≡
>>> df1.radd(df2)
    A   B   C
0  11  22  33
1  44  55  66
2  77  88  99
```

```
>>> df1 + df3
      A     B     C
0   101.0 202.0 303.0
1   404.0 505.0 606.0
2     NaN   NaN   NaN
   ≡
>>> df1.add(df3)
      A     B     C
0   101.0 202.0 303.0
1   404.0 505.0 606.0
2     NaN   NaN   NaN
   ≡
>>> df1.radd(df3)
      A     B     C
0   101.0 202.0 303.0
1   404.0 505.0 606.0
2     NaN   NaN   NaN
```

```
>>> df1 + df4
      A     B     C
0  1001  2002   NaN
1  4004  5005   NaN
2  7007  8008   NaN
   ≡
>>> df1.add(df4)
      A     B     C
0  1001  2002   NaN
1  4004  5005   NaN
2  7007  8008   NaN
   ≡
>>> df1.radd(df4)
      A     B     C
0  1001  2002   NaN
1  4004  5005   NaN
2  7007  8008   NaN
```

See, + and <DF>.add and <DF>.radd produce the same result

See NaN values for non-matching indexes but sum calculated for matching values after data alignment

<DF>.add( ) and **<DF>.radd( )** work identically if the datatypes in aligned cells are the same.

NOTE

When + is performed on string values, it concatenates them.

IMPORTANT

Carefully notice the results and their data types. You will notice that the results of **df1+df3** have the sum values displayed as *float* numbers. Do you know why? The reason behind it is that whenever any column of a dataframe has a NaN value in it, then its type is automatically changed to floating point because in Python integer types cannot store NaN values. Thus to accommodate the NaN values, the dtype of a column is automatically upgraded from integer to floating point. Hence you can see that a column having some numbers as well as NaN values has a floating point representation.

NOTE

Python integer types cannot store NaN values. To store a NaN value in a column, the datatype of a column is changed to non-integer suitable type.

Subtraction Binary operation [Using – , sub( ) and rsub( )]

You can perform **subtract** binary operation on two dataframe objects using either – (minus) operator or using sub( ) as per syntax : **<DF>.sub(<DF>)** which means <DF1> – <DF2> or by using rsub( ) *i.e., reverse subtract* as per syntax : **<DF1>.rsub(<DF2>)** which means <DF2> – <DF1>

```
>>> df1 - df2
    A   B   C
0  -9 -18 -27
1 -36 -45 -54
2 -63 -72 -81
   ≡
>>> df1.sub(df2)
    A   B   C
0  -9 -18 -27
1 -36 -45 -54
2 -63 -72 -81
```

```
>>> df1 - df3
      A      B      C
0   -99.0  -198.0  -297.0
1  -396.0  -495.0  -594.0
2     NaN    NaN    NaN
   ≡
>>> df1.sub(df3)
      A      B      C
0   -99.0  -198.0  -297.0
1  -396.0  -495.0  -594.0
2     NaN    NaN    NaN
```

```
>>> df1 - df4
      A      B      C
0   -999   -1998   NaN
1  -3996   -4995   NaN
2  -6993   -7992   NaN
   ≡
>>> df1.sub(df4)
      A      B      C
0   -999   -1998   NaN
1  -3996   -4995   NaN
2  -6993   -7992   NaN
```

```
>>> df2 - df1
    A   B   C
0   9  18  27
1  36  45  54
2  63  72  81
   ≡
>>> df1.rsub(df2)
    A   B   C
0   9  18  27
1  36  45  54
2  63  72  81
```

```
>>> df3 - df1
      A      B      C
0    99.0   198.0  297.0
1   396.0   495.0  594.0
2     NaN    NaN    NaN
   ≡
>>> df1.rsub(df3)
      A      B      C
0    99.0   198.0  297.0
1   396.0   495.0  594.0
2     NaN    NaN    NaN
```

```
>>> df4 - df1
      A      B      C
0   999    1998   NaN
1  3996    4995   NaN
2  6993    7992   NaN
   ≡
>>> df1.rsub(df4)
      A      B      C
0   999    1998   NaN
1  3996    4995   NaN
2  6993    7992   NaN
```

See, – and <DF>.sub and <DF>.rsub perform subtraction on dataframes

Operations that produce the identical result are shown with same shade

As you can see that :

df1 – df2 is equal to **df1.sub(df2)**

but

df2 – df1 is equal to **df1.rsub(df2)**

NOTE

If you are performing subtraction on two dataframes, make sure the data types of values are subtraction compatible (*e.g.*, you cannot subtract two strings) otherwise Python will raise an error. Check Solved problem 10.

Multiplication Binary operation [using * and (mul)]

You can perform **multiplication** binary operation on two dataframe objects using either * operator or using **mul( )** as per syntax : <DF>.mul(<DF>)

```
>>> df1 * df2
     A    B    C
0   10   40   90
1  160  250  360
2  490  640  810
```

≡

```
>>>df1.mul(df2)
     A    B    C
0   10   40   90
1  160  250  360
2  490  640  810
```

```
>>> df1 * df3
        A       B       C
0   100.0   400.0   900.0
1  1600.0  2500.0  3600.0
2     NaN     NaN     NaN
```

≡

```
>>> df1.mul(df3)
        A       B       C
0   100.0   400.0   900.0
1  1600.0  2500.0  3600.0
2     NaN     NaN     NaN
```

```
>>> df1 * df4
        A       B    C
0    1000    4000  NaN
1   16000   25000  NaN
2   49000   64000  NaN
```

≡

```
>>> df1.mul(df4)
        A       B    C
0    1000    4000  NaN
1   16000   25000  NaN
2   49000   64000  NaN
```

See, * and <DF>.mul perform multiplication on dataframes

Please note that method **rmul**, *i.e.*, **<.DF>.rmul(<DF>)** works the same as **mul** method as multipli- cation is commutative, hence we are not showing it here.

Division Binary operation [using / and div( )]

You can perform **division** binary operation on two dataframe objects using either/ operator or using **div( )** as per syntax : **<DF>.div(<DF>)**

```
>>> df1 / df2
     A    B    C
0  0.1  0.1  0.1
1  0.1  0.1  0.1
2  0.1  0.1  0.1
```

≡

```
>>> df1.div(df2)
     A    B    C
0  0.1  0.1  0.1
1  0.1  0.1  0.1
2  0.1  0.1  0.1
```

```
>>> df2 / df1
      A     B     C
0  10.0  10.0  10.0
1  10.0  10.0  10.0
2  10.0  10.0  10.0
```

≡

```
>>> df1.rdiv(df2)
      A     B     C
0  10.0  10.0  10.0
1  10.0  10.0  10.0
2  10.0  10.0  10.0
```

```
>>> df1 / df3
      A     B     C
0  0.01  0.01  0.01
1  0.01  0.01  0.01
2   NaN   NaN   NaN
```

≡

```
>>> df1.div(df3)
      A     B     C
0  0.01  0.01  0.01
1  0.01  0.01  0.01
2   NaN   NaN   NaN
```

```
>>> df3 / df1
       A      B      C
0  100.0  100.0  100.0
1  100.0  100.0  100.0
2    NaN    NaN    NaN
```

≡

```
>>> df1.rdiv(df3)
       A      B      C
0  100.0  100.0  100.0
1  100.0  100.0  100.0
2    NaN    NaN    NaN
```

See, / and <DF>.div and <DF>.rdiv perform division on dataframes

Operations that produce the identical result are shown with same shade

In an online contest, two 2-player teams' points in 4 rounds are stored in two DataFrames as shown below :

Team 1's points df1

	p1	p2
1	700	490
2	975	460
3	970	570
4	900	590

Team 2's points df2

	p1	p2
1	1100	1400
2	1275	1260
3	1270	1500
4	1400	1190

Following programs are based on the above given two DataFrames.

EXAMPLE 8 *Write a program to calculate total points earned by both the teams in each round.*

SOLUTION

Note : performing binary addition using + operator

```
import pandas as pd
d1 = { 'p1': {'1': 700, '2': 975, '3': 970, '4': 900}, \
       'p2': {'1': 490, '2': 460, '3': 570, '4': 590} }
d2 = { 'p1': {'1' : 1100, '2': 1275, '3': 1270, '4': 1400}, \
       'p2': {'1': 1400, '2': 1260, '3': 1500, '4': 1190} }
df1 = pd.DataFrame(d1)
df2 = pd.DataFrame(d2)

print("Team 1's performance")
print(df1)
print("Team 2's performance")
print(df2)
print("Points earned by both teams")
print(df1 + df2)
```

Output

```
Team 1's performance
    p1   p2
1  700  490
2  975  460
3  970  570
4  900  590
Team 2's performance
     p1    p2
1  1100  1400
2  1275  1260
3  1270  1500
4  1400  1190
Points earned by both teams
     p1    p2
1  1800  1890
2  2250  1720
3  2240  2070
4  2300  1780
```

EXAMPLE 9 *Consider the points earned by two teams, display how much point difference* ***Team2*** *has with* ***Team1****.*

SOLUTION

Note : performing binary subtraction using a function

```
import pandas as pd
:  # df1 and df2 created or loaded
print("Team2's points' difference")
print(df1.rsub(df2))    # or you can write df2.sub(df1)
```

Output

```
Team2's points' difference
    p1   p2
1  400  910
2  300  800
3  300  930
4  500  600
```

Using the operators you can create expressions, just as you normally do with other variables, e.g., see what following program is doing to calculate average points.

EXAMPLE 10 *Given two DataFrames storing points of a 2-player teams in four rounds, write a program to calculate average points obtained by each player in each round.*

SOLUTION

```
import pandas as pd
:       # df1 and df2 created or loaded
print("Team 1's performance")
print(df1)
print("Team 2's performance")
print(df2)
print("Average points scored by each player")
av = (df1 + df2)/2    #av stores average points now
print(av)
```

An expression using binary operations

Output

```
Team 1's performance
   p1   p2
1  700  490
2  975  460
3  970  570
4  900  590
Team 2's performance
     p1    p2
1  1100  1400
2  1275  1260
3  1270  1500
4  1400  1190
Average points scored by each player
     p1    p2
1   900   945
2  1125   860
3  1120  1035
4  1150   890
```

2.4 Descriptive Statistics with Pandas

Python Pandas is a widely used data science library and it offers many useful functions. Among many other functions, the functions offered by Pandas also include many useful statistical functions. In this section, we are going to talk about the same. Here we shall talk about common statistical and aggregate functions as suggested by the syllabus.

For all the functions being covered in this section, we shall be using the following (reference) dataframe :

Reference dataframe namely *prodf*

```
>>> iprod = { 'Rice': {'Andhra P.':7452.4, 'Gujarat':1930.0, 'Kerala':2604.8,\
                       'Punjab':11586.2, 'Tripura':814.6, 'Uttar P.':13754.0},
           'Wheat': { 'Andhra P.':np.NaN, 'Gujarat':2737.0, 'Kerala':np.NaN,\
                       'Punjab':16440.5, 'Tripura':0.5, 'Uttar P.':30056.0},
           'Pulses': { 'Andhra P.':931.0, 'Gujarat':818.0, 'Kerala':1.7,\
                       'Punjab':33.0, 'Tripura':23.2, 'Uttar P.':2184.4},
           'Fruits': { 'Andhra P.':7830.0, 'Gujarat':11950.0, 'Kerala':113.1,\
                       'Punjab':7152.0, 'Tripura':44.1, 'Uttar P.':140169.2} }

# Dictionary iprod contains values taken from GOI statistics[1] for a year in 2010-2020 decade
```

1. https://m.rbi.org.in/Scripts/AnnualPublications.aspx


```
>>> prodf = pd.DataFrame(iprod)
```

```
>>> prodf
           Fruits   Pulses    Rice    Wheat
Andhra P.  7830.0   931.0    7452.4    NaN
Gujarat    11950.0  818.0    1930.0    2737.0
Kerala     113.1    1.7      2604.8    NaN
Punjab     7152.0   33.0     11586.2   16440.5
Tripura    44.1     23.2     814.6     0.5
Uttar P.   140169.2 2184.4   13754.0   30056.0
```

Let us now begin with our discussion on statistical functions.

NOTE

Please note, like earlier, we shall discuss only that much functionality (arguments etc.) of a function, which is required for covering basic functioning and working. For complete syntax and details of these functions, you may refer to the Pandas documentation.

2.4.1 Functions min( ) and max( )

The **min( )** and **max( )** functions find out the minimum or maximum value respectively from a given set of data (dataframe in our case).

Let us see how these work. The syntax for using **min( )** and **max( )** is :

```
<dataframe>.min(axis = None, skipna = None, numeric_only = None)
<dataframe>.max (axis = None, skipna = None, numeric_only = None)
```

To see min( ), max( ) functions in action

Scan QR Code

Parameters :

axis (*0 or 1*) by default, minimum or maximum is calculated along axis 0 (*i.e.*, {index (0), columns (1)}).

skipna (*True or False*) Exclude NA/null/NaN values when computing the result.

numeric_only (*True or False*) Include only *float, int, boolean* columns for calculating min or max. If *None*, will attempt to use everything, than use only numeric data.

You can skip **skipna** and **numeric_only** arguments if your dataframe mainly contains numeric data. Consider following examples :

```
>>> prodf.min()
Fruits    44.1
Pulses     1.7
Rice     814.6
Wheat      0.5
dtype: float64
```

By default, the calculation is done along the rows (index), *i.e.*, when (axis = 0 or not specified at all) and **for each column** the calculated result is displayed.

```
>>> prodf.max()
Fruits    140169.2
Pulses      2184.4
Rice       13754.0
Wheat      30056.0
dtype: float64
```

```
>>> prodf.min(axis = 1)
Andhra P.     931.0
Gujarat       818.0
Kerala          1.7
Punjab         33.0
Tripura         0.5
Uttar P.     2184.4
dtype: float64
```

When **axis = 1** argument is specified then calculation is done *along* the columns and **for each row**, the calculated result is displayed.

```
>>> prodf.max(axis = 1 )
Andhra P.     7830.0
Gujarat      11950.0
Punjab       16440.5
Tripura        814.6
Uttar P.    140169.2
dtype: float64
```

Since the above dataframe contains mostly numeric values, the arguments **skipna** and **numeric_only** won't make much of difference (*see* below and compare with earlier results)

```
>>> prodf.min(skipna = True, numeric_only = True)
Fruits  44.1
Pulses  1.7
Rice    814.6
Wheat   0.5
dtype: float64

>>> prodf.max(skipna = True, numeric_only = True)
Fruits  140169.2
Pulses  2184.4
Rice    13754.0
Wheat   30056.0
dtype: float64
```

Only non_NaN, numeric values are used for calculation because of the arguments : skipna = True, numeric_only = True

Given a DataFrame **dfmks** that stores highest marks in five subjects across four sections of a school :

	A	B	C	D
Acct	99	94.0	92	97.0
Eco	90	94.0	92	97.0
Eng	95	89.0	91	89.0
IP	94	NaN	99	95.0
Math	97	100.0	99	NaN

Marks dataframe dfmks
As a disciplinary measure, exams for two sections were called off and hence NaN values there.

Reference 2.1

EXAMPLE 11 *Consider the DataFrame (dfmks) given above (Reference 2.1). Write a program to print the maximum marks scored in each subject across all sections.*

SOLUTION

```
import pandas as pd
import numpy as np
: # given dfmks created or loaded
print("Maximum marks scored in each subject")
print(dfmks.max(axis = 1))
```

Output

```
Maximum marks scored in each subject
Acct    99.0
Eco     97.0
Eng     95.0
IP      99.0
Math    100.0
dtype: float64
```

EXAMPLE 12 *Which statement would you change in the above program so that it considers only non-Nan values for calculation purpose ?*

SOLUTION The code line

```
print(dfmks.max(axis = 1))
```

should be changed as follows :

```
print(dfmks.max(axis = 1, skipna = True))
```

to consider only non-NaN values for calculation purpose.

EXAMPLE 13 *Consider the marks' DataFrame (dfmks) given above (reference 3.1). Write a program to print the maximum marks scored in a section, across all subjects.*

SOLUTION

```
import pandas as pd
import numpy as np
: # given dfmks created or loaded
print("Maximum marks scored in a section")
print(dfmks.max())
```

Output

```
Maximum marks scored in a section
A     99.0
B    100.0
C     99.0
D     97.0
dtype: float64
```

Index of Maximum and Minimum Values

You can get the index of maximum and minimum value in columns using idxmax( ) and idxmin( ) functions, as shown below. These are used as per syntax :

```
<DF>.idxmax()
<DF> .idxmin()
```

Following screenshots show it for dataframe objects **dtf5** given below.

```
>>> dtf5.idxmin()
Amount     0
Average    1
people     0
dtype: int64
```

Similarly indexes for minimum and maximum values in each column shown here

```
>>> dtf5.idxmax()
Amount     2
Average    0
people     2
dtype: int64
```

2.4.2 Functions mode( ), mean( ), median( )

To see mode( ), mean( ), median( ) functions in action

Scan QR Code

Functions mode( ), mean( ) and median( ) are common statistics functions used as per the following :

- Function ***mode( )*** returns the mode value (*i.e.*, the value that appears most often in a given set of values) from a set of **values**. The syntax for using **mode( )** is :

```
<dataframe>.mode(axis = 0, numeric_only = False)
```

The **mode( )** gets the mode(s) of each element along the axis selected. It adds a row for each mode per label and fills in gaps with NaN. (*NaN in mode is explained in the following section 2.4.2A*)

- Function ***mean( )*** returns the computed mean (average) from a set of values. The syntax for using the function **mean( )** is :

```
<dataframe>.mean(axis = None, skipna = None, numeric_only = None)
```

The *mean( )* returns the mean of the values for the requested axis.

- Function ***median( )*** returns the middle number from a set of numbers. It returns the median value that separates the higher half from the lower half of a set of values. The syntax for using the function **median( )** is :

```
<dataframe>. median(axis = None, skipna = None, numeric_only = None)
```

The *median( )* returns the median of the values for the requested axis.

Above given three functions have common parameters as explained below :

Parameters :

axis {0 or 'index', 1 or 'columns'}, default 0
axis 0 or 'index' : get mode/mean/median of each column
axis 1 or 'columns' : get mode/mean/median of each row

skipna boolean (*True or False*), default *True* ; Exclude NA/null/NaN values when computing the result. Used only by mean( ) and median( ).

numeric_only boolean (*True or False*), default None ; Include only *float, int, boolean* columns for calculating mode/mean/median. If *None*, will attempt to use everything, than use only numeric data.

Consider following examples now (notice the arguments passed) :

mode( ) Returns the mode (the value appearing the most)

```
>>> prodf.mode()
   Fruits  Pulses  Rice    Wheat
0  44.1    1.7     814.6   0.5
1  113.1   23.2    1930.0  2737.0
2  7152.0  33.0    2604.8  16440.5
3  7830.0  818.0   7452.4  30056.0
4  11950.0 931.0   11586.2 NaN
5  140169.2 2184.4 13754.0 NaN
```

```
>>> prodf.mode(axis = 1)
            0       1       2       3
Andhra P.   931.0   7452.4  7830.0  NaN
Gujarat     818.0   1930.0  2737.0  11950.0
Kerala      1.7     113.1   2604.8  NaN
Punjab      33.0    7152.0  11586.2 16440.5
Tripura     0.5     23.2    44.1    814.6
Uttar P.    2184.4  13754.0 30056.0 140169.2
```

median( ) Returns the middle value

```
>>> prodf.median()
Fruits    7491.00
Pulses    425.50
Rice      5028.60
Wheat     9588.75
```

```
>>> prodf.median(axis = 1)
Andhra P.   7452.40
Gujarat     2333.50
Kerala      113.10
Punjab      9369.10
Tripura     33.65
Uttar P.    21905.00
```

mean( ) Returns the mean/average value

```
>>> prodf.mean()
Fruits    27876.400000
Pulses    665.216667
Rice      6357.000000
Wheat     12308.500000
```

```
>>> prodf.mean(axis = 1)
Andhra P.   5404.466667
Gujarat     4358.750000
Kerala      906.533333
Punjab      8802.925000
Tripura     220.600000
Uttar P.    46540.900000
```

2.4.2A NaN values in mode( )

There may be more than one modes for a DataFrame, *e.g.*, if 2 values are repeated exact number of times and they both are eligible to become mode, then Pandas will display 2 lines to show two modes and for the rest of the columns other than the mode's column, it will fill NaN.

Following example will make it clear.

Given a DataFrame **mks1** as shown below :

	A	B	C	D
Acct	99	94	92	99
Eco	94	94	92	97
Eng	95	89	91	89
IP	94	87	99	94
Math	97	100	99	99

Reference 2.2

Carefully observe column C. Which value is repeated most of the times ? Now read on.

EXAMPLE **14** *Consider the DataFrame **mks1** given above (**Reference 2.2**). Write a program to calculate mode (maximum repeated value) for each section.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks1 created or loaded
print("Mode value for each section")
print(mks1.mode())
```

Output

```
   A     B     C   D
0  94.0  94.0  92  99.0
1  NaN   NaN   99  NaN
```

Let us now analyse above output as it contains some NaN values.

The mode for DataFrame ***mks1***, has returned 2 values for section **C** (92 and 94) and 1 value each for sections **A**(94), **B**(94) and **D**(99).

Now carefully have a look at given DataFrame mks1 (*reference* 2.2) and look at section C. There are 2 values which are repeated maximum number of times (*twice*) out of 5 values (92 and 99) and hence both qualify to become mode for section C. But for sections A, B and D, there is exactly 1 value for mode.

So, Pandas, will display 2 values for section C in two rows and since other sections do not have 2nd mode value, it will fill them with NaN and hence the result (*see* above).

EXAMPLE **15** *Consider the DataFrame **mks1** given above (**Reference 2.2**). Write a program to calculate mode (maximum repeated value) for each subject.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks1 created or loaded
print("Mode value for each subject")
print(mks1.mode(axis = 1))
```

Output

```
        0
Acct    99
Eco     94
Eng     89
IP      94
Math    99
```

NOTE

In *mode()*, if there are multiple mode values, Pandas will display all of them in separate rows and for other columns, it will fill NaN values (refer Example 17 and its output explanation).

EXAMPLE **16** *Consider the DataFrame* ***mks1*** *given above (Reference 2.2). Write a program to calculate median value for each section.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks1 created or loaded
print("Median value for each section")
print(mks1.median())
```

Output

```
Median value for each section
A    95.0
B    94.0
C    92.0
D    97.0
dtype: float64
```

EXAMPLE **17** *Consider the DataFrame* ***mks1*** *given above (Reference 2.2). Write a program to calculate median value for each subject.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks1 created or loaded
print("Median value for each subject")
print(mks1.median (axis =1))
```

Output

```
Median value for each subject
Acct    96.5
Eco     94.0
Eng     90.0
IP      94.0
Math    99.0
dtype: float64
```

EXAMPLE **18** *Consider the DataFrame* ***mks1*** *given above (Reference 2.2). Write a program to calculate average value (mean) for each section.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks1 creates or loaded
print("Average (Mean) marks for each section")
print(mks1.mean())
```

Output

```
Average (Mean) marks for each section
A    95.8
B    92.8
C    94.6
D    95.6
dtype: float64
```

EXAMPLE **19** *Consider the DataFrame* ***mks1*** *given above (Reference 2.2). Write a program to calculate average value (mean) for each subject.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks1 created or loaded
print("Average (Mean) marks for each subject")
print(mks1.mean(axis = 1))
```

Output

```
Average (Mean) marks for each subject
Acct    96.00
Eco     94.25
Eng     91.00
IP      93.50
Math    98.75
dtype: float64
```


2.4.3 Functions count( ) and sum( )

The function **count( )** counts the non-NA entries for each row or column. The values *None, NaN* etc., are considered as NA in Pandas. The syntax for using **count( )** is :

```
<dataframe>.count(axis = 0, numeric_only = False)
```

Parameters :

axis	{0 or 'index', 1 or 'columns'}, default 0; If 0 or 'index', counts are generated for each column. If 1 or 'columns', counts are generated for each row.
numeric_only	*boolean* (*True* or *False*), default *False* ; Include only *float, int* or *boolean* data.

The function ***sum( )*** returns the sum of the values for the requested axis. The syntax for **sum( )** is :

```
<dataframe>.sum(axis = None, skipna = None, numeric_only = None, min_count = 0)
```

Parameters :

axis	{ index (0), columns (1) } default 0 ;
skipna	boolean, default True ; Exclude NA/null values when computing the result.
numeric_only	*boolean*, default *None*; Include only *float, int, boolean columns.* If *None*, will attempt to use everything, than use only numeric data.
min_count	*int*, default 0 ; the required number of valid values to perform the operation. If fewer than *min_count* non-NA values are present the result will be NA. (Added with the default being 1. This means the sum or product of an all-NA or empty series is NaN.)

Consider following examples (notice the arguments passed) :

count( ) Returns count of non NA values for each row/column

```
>>> prodf.count()
Fruits    6
Pulses    6
Rice      6
Wheat     4
```

```
>>> prodf.count(axis = 1)
Andhra P.    3
Gujarat      4
Kerala       3
Punjab       4
Tripura      4
Uttar P.     4
```

sum( ) Returns sum of values for given axis

```
>>> prodf.sum()
Fruits    167258.4
Pulses    3991.3
Rice      38142.0
Wheat     49234.0
```

```
>>> prodf.sum(axis = 1)
Andhra P.    16213.4
Gujarat      17435.0
Kerala       2719.6
Punjab       35211.7
Tripura      882.4
Uttar P.     186163.6
```

To see count( ), sum( ) functions in action

Scan QR Code

EXAMPLE 20 *Consider the DataFrame dfmks (Reference 2.1), calculate how many exams were conducted, for each section.*

SOLUTION

```
import pandas as pd
import numpy as np
:          # given dfmks created or loaded
print("No. of exams conducted for each section")
print(dfmks.count())
```

Output

```
No. of exams conducted for each section
A   5
B   4
C   5
D   4
```

EXAMPLE 21 *Consider the DataFrame dfmks (Reference 2.1). Assuming that in each section, only 1 student scored maximum marks in each subject, write a program to calculate the topper's total marks, in each section.*

SOLUTION

```
import pandas as pd
import numpy as np
:          # given dfmks created or loaded
print("Topper's total marks in each section")
print(dfmks.sum())
```

Output

```
Topper's total marks in each section
A   475.0
B   377.0
C   473.0
D   378.0
```

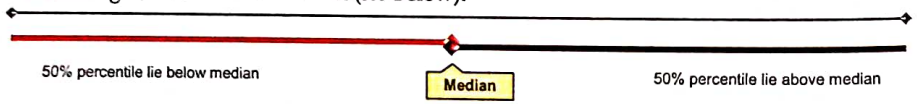
2.4.4 Functions quantile( ), std( ) and var( )

The quantile( ) function returns the values at the given quantiles over requested axis (axis 0 or 1).

Quantile is a statistical concept, which you must have read before. Please note that this is not a statistics book and thus we cannot explain **quantile** in details here, but still, we are covering quantile briefly in lines below. For detailed explanation, you may refer to any statistics book.

Understanding Quantile

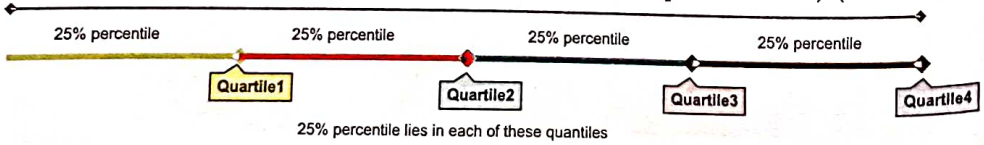
You know what a median is. In simple English, it is a mid-way point of some data. In other words, the **median** splits the data set in half, *i.e.*, 50% on the left/below of median and the 50% is on the right/above of the median (*see* below).



In other words, the median, or 50th percentile of a continuous distribution splits the distribution in half in terms of area. That means, if we divide a distribution in 2 exact halves (50-50), we call the dividing marker a **median**.

The divided parts are called **quantiles**, which means **median** divides a distribution in 2 ***quantiles*** (50% percentile each).

If we divide a distribution in 4 exact quarters (25 each), we call the dividing marker a **quartile**. That is, a **quartile** divides a distribution in ***4 quantiles*** (25% percentile each). (*see* below)



Thus, you can say that when a distribution is divided in some equal parts, the equally divided parts are called **quantiles** and the dividing marker has a specific name as per the division.

Following list will give you the names of such dividing markers.

S.No.	Dividing Measure Name	Division basis	Quantile share in percentile
1.	Median	2 quantiles	50%
2.	Terciles	3 quantiles	33.333333%
3.	**Quartiles**	**4 quantiles**	**25%**
4.	Quintiles	5 quantiles	20%
5.	Sextiles	6 quantiles	16.66666%
6.	Septiles	7 quantiles	14.285714285%
7.	Octiles	8 quantiles	12.25%
8.	Deciles	10 quantiles	10%
9.	Duodeciles	12 quantiles	8.3333333%
10.	Vigintiles	20 quantiles	5%
11.	Percentiles	100 quantiles	1%

2.4.4A The quantile( ) Function

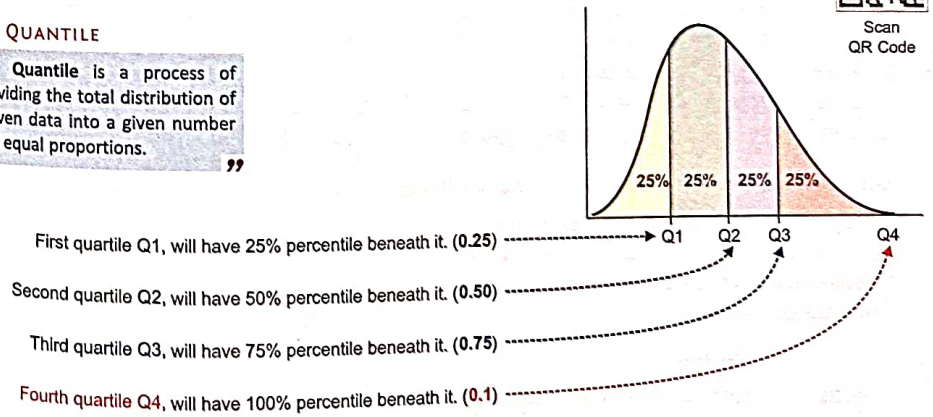
To see quantile( ) function in action

Scan QR Code

After above discussion, you can say that a distribution divided through **Quartiles** will have ***4 quantiles***, each having 25% percentile of distribution share (*see* below). Following figure shows a Quartile division with first intermediate quartile marked as Q1, second as Q2, and third as Q3.

QUANTILE

> **Quantile** is a process of dividing the total distribution of given data into a given number of equal proportions.



In order to calculate any type of quantiles in Pandas, you can use **quantile( )** function as per following syntax :

```
<dataframe>.quantile(q = 0.5, axis = 0, numeric_only = True)
```

Parameters :

q float or array-like, default 0.5 (50% quantile). 0 <= q <= 1, the quantile(s) to compute.

Note. Since as per syllabus, you have to produce **quartiles,** *i.e.*, a 4-quantile result, thus you should give this argument as : q = **[0.25, 0.5, 0.75, 1.0]**

axis [{0, 1, 'index', 'columns'} (default 0)] 0 or 'index' for row-wise,
1 or 'columns' for column-wise.

numeric_only If False, the quantile of *datetime* and *timedelta* data will be computed as well.

The quantile( ) returns quantiles (Series or DataFrame)

- If **q** is an array, a DataFrame will be returned where the index is **q**, the columns are the columns of self, and the values are the **quantiles**. (we shall get this as we shall compute quartiles)
- If **q** is a float, a Series will be returned where the index is the columns of self and the values are the quantiles.

NOTE

In quantile( ), the default **q** value is 0.5 (*i.e.*, for median). Parameter **q** specifies the percentile share for each quantile to be calculated.

See example below (notice the arguments passed) :

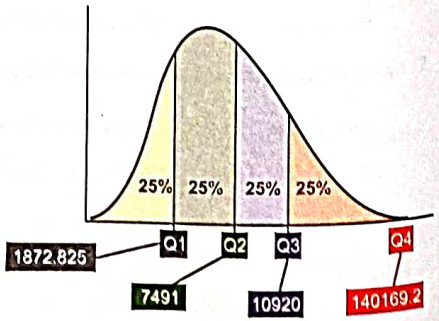
```
>>> prodf.quantile([0.25, 0.5, 0.75, 1.0])
        Fruits     Pulses     Rice       Wheat
0.25    1872.825   25.65      2098.70    2052.875
0.50    7491.000   425.50     5028.60    9588.750
0.75    10920.000  902.75     10552.75   19844.375
1.00    140169.200 2184.40    13754.00   30056.000
```

```
>>> prodf.quantile( [ 0.25, 0.5, 0.75, 1.0], axis = 1)
        Andhra P.  Gujarat    Kerala     Punjab      Tripura    Uttar P.
0.25    4191.7     1652.00    57.40      5372.250    17.525     10861.6
0.50    7452.4     2333.50    113.10     9369.100    33.650     21905.0
0.75    7641.2     5040.25    1358.95    12799.775   236.725    57584.3
1.00    7830.0     11950.00   2604.80    16440.500   814.600    40169.2
```

Let us understand the output of **prodf.quantile([0.25, 0.5, 0.75, 1.0])** (given above).

Let us consider **Fruits column's** 25 percentile (0.25 row), 50 percentile (0.50 row), 75 percentile (0.75 row), 100 percentile (01.00 row) values, which are :

	Fruits	
0.25	1872.825	Q1 marker at this point
0.50	7491.000	Q2 marker at this point
0.75	10920.000	Q3 marker at this point
1.00	140169.200	Q4 marker at this point



It means, if we plot a distribution graph for *Fruits*, 25% percentile's dividing marker (Q1) will be at 1872.825 on the graph, Q2 at point 7491 on the graph, Q3 at point 10920 and Q4 at 140169.2 point on the graph.

QUARTILE

"**Quartile** refers to the division of the total distribution of given data into a four equal proportions with each containing one fourth of the total population."

EXAMPLE **22** *Given a DataFrame* **mks2** *storing marks of 10 students in a class, in five subjects, as shown below :*

	1	2	3	4	5	6	7	8	9	10
Acct	99	94	81	70	88	90	41	61	42	68
Eco	94	94	72	67	82	81	36	54	42	67
Eng	95	89	71	69	82	79	51	60	45	66
IP	94	87	79	65	89	81	42	63	43	64
Math	97	100	65	69	86	84	40	55	40	60

Write a program to measure the subject wise performance of the students, so that the teacher gets to know the following details (for each subject) :

25% percentile students scored below ?
50% percentile students scored below ?
75% percentile students scored below ?

Also, separately list the performance for the subject 'IP'.

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks2 created or loaded
print("Performance of students, subject wise")
print(mks2.quantile([0.25, 0.5, 0.75, 1.0], axis = 1))
```

Output

```
Performance of students, subject wise
      Acct   Eco    Eng    IP     Math
0.25  62.75  57.25  61.50  63.25  56.25
0.50  75.50  69.50  70.00  72.00  67.00
0.75  89.50  81.75  81.25  85.50  85.50
1.00  99.00  94.00  95.00  94.00 100.00
```

Thus, in the subject IP :

25% percentile students scored below 63.25
50% percentile students scored below 72.0
75% percentile students scored below 85.5

2.4.4B The std( ) Function

The **std( )** function computes the standard deviation over requested axis.

The syntax for using the std( ) function is :

```
<dataframe>.std(axis = None, skipna = None, numeric_only = None,)
```

Parameters :

axis	{index (0), columns (1)} default 0
skipna	boolean, default True ; Exclude NA/null/NaN values. If an entire row/column is NA, the result will be NA
numeric_only	boolean, default None ; Include only float, int, boolean columns. If *None*, will attempt to use everything, than use only numeric data.

Consider the following examples (carefully notice the arguments passed)

```
>>> prodf.std()
Fruits      55288.911975
Pulses        854.872997
Rice         5434.253848
Wheat       13846.150566
dtype: float64
```

```
>>> prodf.std(axis = 1)
Andhra P.     3878.733486
Gujarat       5121.617152
Kerala        1471.796434
Punjab        6969.368921
Tripura        396.400859
Uttar P.     63457.389515
dtype: float64
```

EXAMPLE 23 *Consider the mks2 DataFrame used in example 25 above. Write a program to calculate the standard deviation for each subject.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks2 created or loaded
print("Standard deviation, subject wise")
print(mks2.std(axis = 1))
```

Output

```
Standard deviation, subject wise
Acct  20.699973
Eco   20.063510
Eng   16.021167
IP    18.397162
Math  21.680508
dtype: float64
```

2.4.4C The var( ) Function

The **var( )** function computes variance and returns the unbiased variance over the requested axis. The syntax for using the **var( )** function is :

```
<dataframe>.var(axis = None, skipna = None, numeric_only = None,)
```

axis	{index (0), columns (1)} default 0
skipna	boolean, default True; Exclude NA/null values. If an entire row/column is NA, the result will be NA
numeric_only	boolean, default *None*; Include only *float, int, boolean* columns. If *None*, will attempt to use everything, then use only numeric data.

Consider the following examples (notice the arguments passed) :

```
In [57]: sal_df.var()
Out[57]:
2016    8.022917e+07
2017    5.299000e+07
2018    1.875000e+07
2019             NaN
dtype: float64
```

```
In [58]: sal_df.var(axis = 1)
Out[58]:
Qtr1    1.336692e+08
Qtr2    2.450333e+07
Qtr3    3.333333e+07
Qtr4    3.175000e+07
dtype: float64
```

EXAMPLE 24 *Consider the mks2 DataFrame used in example 25 above. Write a program to calculate the variance for each subject.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given mks2 created or loaded
print("Variance, subject wise")
print(mks2.var(axis = 1))
```

Output

```
Variance, subject wise
Acct     428.488889
Eco      402.544444
Eng      256.677778
IP       338.455556
Math     470.044444
dtype: float64
```

2.4.5 The describe( ) Function

So far you have learnt about many descriptive statistics functions : count, mean, minimum, maximum, quartiles etc. These descriptive statistics details can be calculated through a single function, namely **describe( )**. The **describe( )** is used as :

```
<DF>.describe( )
```

The **describe( )** gives following information for a dataframe object having numeric columns :

- **Count.** Count of non-NA values in a column
- **Mean.** Computed mean of values in a column
- **Std.** Standard deviation of values in a column
- **Min.** Minimum value in a column
- **25%, 50%, 75%.** Percentiles of values in that column how these percentiles are calculated, we are explaining below.
- **Max.** Maximum value in a column

Consider the following examples (notice the arguments passed) :

```
>>> prodf.describe()
          Fruits      Pulses        Rice        Wheat
count      6.000000    6.000000    6.000000     4.000000
mean   27876.400000  665.216667  6357.000000  12308.500000
std    55208.911975  854.872997  5434.253848  13846.150566
min       44.100000    1.700000   814.600000      0.500000
25%     1872.825000   25.650000  2098.700000   2052.875000
50%     7491.000000  425.500000  5028.600000   9588.750000
75%    10920.000000  902.750000 10552.750000  19844.375000
max   140169.200000 2184.400000 13754.000000  30056.000000
```

The percentiles of values in a column are calculated using minimum and maximum values of column as per following formula :

```
<Min value> + (<Max value> – <Min value>) * <percentile to be calculated>
```

For example, look at the output produced by **res1.describe( )**. Let us manually calculate the percentile values of 25%, 50% and 75% of column '**A**' of dataframe object **res1** :

Min value of *column* '*A*' = 9 **Max value** of column '*A*' = 63

25% (percentile) as per above formula = 9 + (63 - 9) * 25/100 = 9 + 13.5 = **22.5**

50% (percentile) as per above formula = 9 + (63 – 9) * 50/100 = 9 + 27 = **36.0**

75% (percentile) as per above formula = 9 + (63 – 9) * 75/100 = 9 + 40.5 = **49.5**

Now compare the results with above output of **res1.describe( )**.

Function describe( ) for String type Columns

For a dataframe with string type column(s), the **describe( )** will give information as shown below. Say we have a dataframe object **res6** as shown on the right. If you give res6.describe( ), Pandas will return output as shown below :

```
In [11]: res6
Out[11]:
        0
0    This
1      is
2    good
3    This
4  is fun
```

```
In [12]: res6.describe()
Out[12]:
           0
count      5
unique     4
top     This
freq       2
```

As you can see that **describe( )** gives different output for string columns. The information returned by describe( ) for string columns includes :

- **Count** – the number of non-NA entries in the column
- **Unique** – number of unique entries in the column
- **Top** – the most common entry in the column, *i.e.*, the one with highest frequency. If, however, multiple values have the same highest count, then the **count** and **most common** (*i.e.*, **top**) **pair** will be arbitrarily chosen from among those with the highest count.
- **Freq** – it is the frequency of the most common element displayed as top above.

Now consider a dataframe object that has both types of columns, *numeric* as well as **string** type, as shown below (*on the left*) ; if you give **describe( )** with such a dataframe object then by default Pandas will show details about numeric columns only, see below (*on the right*).

```
In [20]: res5
Out[20]:
     A     B      C       D
0  9.0  18.0   27.0    This
1 36.0  45.0   54.0    This
2 63.0  72.0   81.0  string
3 90.0  99.0  108.0      is
4 91.0  99.0   18.0  string
5 91.0  99.0   18.0    This
```

```
In [21]: res5.describe()
Out[21]:
                A           B           C
count    6.000000    6.000000    6.00000
mean    63.333333   72.000000   51.00000
std     34.471244   34.152599   37.18064
min      9.000000   18.000000   18.00000
25%     42.750000   51.750000   20.25000
50%     76.500000   85.500000   40.50000
75%     90.750000   99.000000   74.25000
max     91.000000   99.000000  108.00000
```

See, even though res5 object contains string column, **describe()** by default, shows details of numeric columns only

The default behavior of **describe( )** is to only provide a summary for the numerical columns. You can give **include = 'all'** as argument to **describe( )** to list summary for all columns. (Notice 'all' is given as string *i.e.*, enclosed in quotes) (*see* below).

```
In [23]: res5.describe(include = 'all')
Out[23]:
                A           B           C    D
count    6.000000    6.000000    6.00000    6
unique        NaN         NaN         NaN    3
top           NaN         NaN         NaN  This
freq          NaN         NaN         NaN    3
mean    63.333333   72.000000   51.00000  NaN
std     34.471244   34.152599   37.18064  NaN
min      9.000000   18.000000   18.00000  NaN
25%     42.750000   51.750000   20.25000  NaN
50%     76.500000   85.500000   40.50000  NaN
75%     90.750000   99.000000   74.25000  NaN
max     91.000000   99.000000  108.00000  NaN
```

With **include = 'all'**, summary of all columns is listed

2.5 Some Other Essential Functions and Functionality

Let us now talk about some other essential functions that you can use on dataframes. Following lines briefly cover these. For understanding purposes, we are using above used dataframe objects *dtf4, df1, df2, df3* and *df4*. We stored the result of subtraction in objects *df1, df2, df3* and *df4* in the form of dataframe objects ***res1, res2, res3*** as shown below :

```
In [8]: res1 = df1.rsub(df2)
In [11]: res1
Out[11]:
   A  B  C
0  9 18 27
1 36 45 54
2 63 72 81
```

```
In [9]: res2 = df1.rsub(df3)
In [12]: res2
Out[12]:
      A     B     C
0  99.0 198.0 297.0
1 396.0 495.0 594.0
2   NaN   NaN   NaN
```

All columns have 2 non-null values and 1 null value

```
In [10]: res3 = df1.rsub(df4)
In [13]: res3
Out[13]:
     A     B   C
0  999  1998 NaN
1 3996  4995 NaN
2 6993  7992 NaN
```

Columns 'A' & 'B' have all non-Column 'C' has all null values

2.5.1 Inspection Function info( )

To inspect broadly or to get basic information about your dataframe object, you can use **info( )** function as follows :

```
<DF>.info()
```

The **info( )** gives following information for a dataframe object :

- Its **type**. Obviously, it is an instance of a DataFrame.
- **Index values.** As each row of a dataframe object has an index, this information shows the assigned indexes.
- **Number of rows** in the dataframe object.
- **Data columns and values in them.** It lists number of columns and count of only non-NA values in them.

- **Datatypes of each column.** The listed datatypes are not necessarily in the corresponding order to the listed columns. You can however use the **dtypes** attribute to get the datatype for each column.
- **Memory_usage.** Approximate amount of memory used to hold the DataFrame.

For instance, see below :

```
In [14]: res1.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
A    3 non-null int64
B    3 non-null int64
C    3 non-null int64
dtypes: int64(3)
memory usage: 112.0 bytes

In [15]: res2.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 2
Data columns (total 3 columns):
A    2 non-null float64
B    2 non-null float64
C    2 non-null float64
dtypes: float64(3)
memory usage: 96.0 bytes
```

```
In [16]: res3.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
A    3 non-null int64
B    3 non-null int64
C    0 non-null float64
dtypes: float64(1), int64(2)
memory usage: 112.0 bytes

In [17]: dtf4.info()
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, Delhi to Chennai
Data columns (total 3 columns):
Population          4 non-null int64
Avg. Income         4 non-null int64
Per Capita Income   4 non-null float64
dtypes: float64(1), int64(2)
memory usage: 112.0+ bytes
```

2.5.2 DataFrame's Top and Bottom Rows using head( ) and tail( )

You can use head( ) and tail( ) to retrieve top 5 or bottom 5 rows respectively of a dataframe object if you pass no argument to these.

These functions are to be used as :

```
<DF>.head([n = 5])
<DF>.tail([n = 5])
```

← *Argument n = <n> is optional and default value of n is taken as 5 and 5 rows are listed*

```
In [34]: res7
Out[34]:
    A     B     C          D
0   9.0   18.0  27.0       This
1   36.0  45.0  54.0       This
2   63.0  72.0  81.0       string
3   90.0  99.0  108.0      is
4   91.0  99.0  18.0       string
5   91.0  99.0  18.0       This
6   91.0  99.0  18.0       This is fun
7   91.0  99.0  18.0       string
8   91.0  99.0  18.0       hellow
9   91.0  99.0  18.0       hellow
10  91.0  99.0  18.0       hellow
11  91.0  99.0  18.0  string hellow
```

```
In [35]: res7.head()
Out[35]:
    A     B     C     D
0   9.0   18.0  27.0  This
1   36.0  45.0  54.0  This
2   63.0  72.0  81.0  string
3   90.0  99.0  108.0 is
4   91.0  99.0  18.0  string
```

```
In [36]: res7.tail()
Out[36]:
    A     B     C          D
7   91.0  99.0  18.0       string
8   91.0  99.0  18.0       hellow
9   91.0  99.0  18.0       hellow
10  91.0  99.0  18.0       hellow
11  91.0  99.0  18.0  string hellow
```


You can retrieve any number of top rows or bottom rows with **head( )** and **tail( )** respectively by specifying value of *n* as argument to head( ) or tail( ) *e.g.,* as <DF>.head(n = 12) or <DF>.tail(n = 24). You can give any value of **n** as per your need (as many rows you want to list)

```
In [37]: res7.head(n = 3)
Out[37]:
    A     B     C     D
0   9.0   18.0  27.0  This
1   36.0  45.0  54.0  This
2   63.0  72.0  81.0  string
```

3 top rows listed (n=3) given

4 bottom rows listed (n=4) given

```
In [38]: res7.tail(n=4)
Out[38]:
    A     B     C          D
8   91.0  99.0  18.0       hellow
9   91.0  99.0  18.0       hellow
10  91.0  99.0  18.0       hellow
11  91.0  99.0  18.0  string hellow
```

You can also write **res7.head(n = 3)** as **res7.head(3)** and **res7.tail(n = 4)** as **res7.tail(4)**, *i.e.,* you can simply pass a number signifying number of rows to be retrieved through **head( )** or **tail( )** functions.

EXAMPLE **25** *Given a DataFrame dtf6.*

	Hospitals	Schools
Delhi	189.0	7916.0
Mumbai	208.0	8508.0
Kolkata	149.0	7226.0
Chennai	157.0	7617.0

Write a program to display top two rows' values of 'Schools' column and last 3 values of 'Hospitals' column.

SOLUTION.
```
import pandas as pd
:   # Series object dtf6 created or loaded
print(dtf6.Schools.head(2))
print(dtf6.Hospitals.tail(3))
```

2.5.3 Cumulative Calculations Functions

Cumulative calculation means after applying a function to one row, its output is also used in next row's calculation, *e.g.,*

1	1
2	+2
3	+3
Normal sum	6

1	1	
2	3	← 1+2 =3
3	6	← 3+3 =6

Cumulative sum

1. cumsum( ) Function

cumsum( ) calculates cumulative sum *i.e.,* in the output of this function, the value of each row is replaced by sum of all prior rows including this row. String value rows use concatenation. It is used as :

```
<DF>.cumsum([axis = None])
```

← *axis argument is optional*

```
>>> res1
    A     B     C
0   9     18    27
1   36    45    54
2   63    72    81
```

For example, for **res1** object (shown on the right), cumsum( ) will give output as shown on the next page.

```
In [40]: res1.cumsum()
Out[40]:
     A    B    C
0    9   18   27
1   45   63   81
2  108  135  162
```

```
In [41]: res1.cumsum(axis='rows')
Out[41]:
     A    B    C
0    9   18   27
1   45   63   81
2  108  135  162
```

Default behaviour is *axis = 'rows'*

```
In [42]: res1.cumsum(axis='columns')
Out[42]:
    A    B    C
0   9   27   54
1  36   81  135
2  63  135  216
```

axis = 'columns' will yield cumulative sum calculated and displayed column-wise

By default, it calculates the sum of columns' value and displays cumulative sum in each successive row (default behaviour is **axis = 'rows'**) but you can give argument as (**axis = 'columns'**) to calculate the cumulative sum which is calculated and displayed column-wise.

2. cumprod( ), cummax( ), cummin( ) Functions

To see cumprod( ), cummax( ), cummin( ) functions in action

Scan QR Code

There are other cumulative functions too. As the name suggests :

- **cumprod( )** gives cumulative product of values
- **cummax( )** gives cumulative maximum of values
- **cummin( )** gives cumulative minimum of values

In the same manner as you used **cumsum( )**, you can use **cumprod( )** to get cumulative product, **cummax( )** to get cumulative maximum and **cummin( )** to get cumulative minimum value from a dataframe object.

Consider some examples given below which are based on two 2-player teams' points DataFrames :

Team 1's points df1

	p1	p2
1	700	490
2	975	460
3	970	570
4	900	590

Player1(p1) and player(p2) points stored

Team 2's points df2

	p1	p2
1	1100	1400
2	1275	1260
3	1270	1500
4	1400	1190

EXAMPLE 26 *Consider above given team points DataFrame df1 and df2, write a program to list maximum points till that round (cumulatively).*

SOLUTION

```
import pandas as pd
:   #df1, df2 created or loaded
print("Cumulative max. points till each round")
print("Team1")
print(df1.cummax( ))
print("Team2")
print(df2.cummax( ))
```

Output

```
Cumulative max. points till each round
Team1
    p1    p2
1  700   490
2  975   490
3  975   570
4  975   590
Team2
     p1    p2
1  1100  1400
2  1275  1400
3  1275  1500
4  1400  1500
```


EXAMPLE 27 *Considering DataFrame of team2's points, write a program to print the total points of each player till each round.*

SOLUTION

```
import pandas as pd
:     # df2 created or loaded
print("Team 2 points")
print(df2)
print("Cumulative total points till each round")
print(df2.cumsum())
```

Output

```
Team 2 points
     p1    p2
1  1100  1400
2  1275  1260
3  1270  1500
4  1400  1190
Cumulative total points till each round
     p1    p2
1  1100  1400
2  2375  2660
3  3645  4160
4  5045  5350
```

2.5.4 Applying Functions on a Subset of Dataframe

Sometimes, you need to apply a function on a selective column or a row or a subset of the data frame. You already know how to access a single row or column or a subset from a data frame. You can work with all the functions in the similar way as discussed here.

Applying Functions on a Column of a DataFrame

To apply a function on a column, you need to use the following in place of dataframe name :

```
<dataframe>[<column name>]
```

We are using the DataFrame **dfmks** (*Reference 2.1*) for below given examples.

```
>>> dfmks['D'].min()
89.0
```

Applying functions on individual column of a dataframe

```
>>> dfmks['B'].count()
4
```

And then apply the function on it (*see* examples below).

EXAMPLE 28 *Consider the DataFrame prodf (given in the beginning of section 2.4) that stores some agriculture statistics of some Indian states. Write a program to find out how many states (only as per prodf's data) produce Wheat crop ?*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given prodf created or loaded
print("Wheat crop is produced by", prodf['Wheat'].count(), "states.")
```

Output

```
Wheat crop is produced by 4 states.
```

Applying Functions on Multiple Columns of a DataFrame

To apply a function on multiple columns, you need to use following in place of dataframe name :

```
<dataframe>[ [<column name>, <columnname>, ...] ]
```

group of column names given in a list within [ ] of dataframe. Notice double [[ ]]

And then apply the function on it (*see* examples below).

EXAMPLE 29 *Consider the DataFrame* ***prodf*** *(given in the beginning of section 2.4) that stores some agriculture statistics of some Indian states. Write a program to find out how many states (only as per* ***prodf****'s data) produce the* ***Wheat*** *crop and how many states produce the* ***Rice*** *crop ?*

SOLUTION

```
import pandas as pd
import numpy as np
:        # given prodf created or loaded
print("Wheat and Rice crops are produced by these many states")
print( prodf[['Wheat', 'Rice']].count() )
```

Output

```
Wheat and Rice crops are produced by these many states
Wheat    4
Rice     6
dtype: int64
```

Applying Functions on a Row of a DataFrame

To apply a function on a row, you need to use following in place of dataframe name :

```
<dataframe>.loc[<row index>, :]
```

And then apply the function on it (*see* examples below).

EXAMPLE 30 *Consider the DataFrame* ***prodf*** *(given in the beginning of section 2.4) that stores some agriculture statistics of some Indian states. Write a program to find out how many crops are produced by the state Andhra Pradesh ?*

SOLUTION

```
import pandas as pd
import numpy as np
:        # given prodf created or loaded
print("Crops produced by Andhra Pradesh are")
                                    print(prodf.loc['Andhra P.', : ].count())
```

Output

```
Crops produced by Andhra Pradesh are
3
```


Applying Functions on a Range of Rows of a DataFrame

To apply a function on multiple rows, you need to use the following :

With row index labels

```
<dataframe>.loc[ <start row label>: <end row label>, : ]
```

With row index numbers

```
<dataframe>.iloc[ <start row>: <end row>, : ]
```

And then apply the function on it (*see* examples below).

```
>>> prodf.loc[:, : ].count(axis = 1)
Andhra P.    3
Gujarat      4
Kerala       3
Punjab       4
Tripura      4
Uttar P.     4
dtype: int64
```

Using loc

```
>>> prodf.loc[:, : ].count( )
Fruits       6
Pulses       6
Rice         6
Wheat        4
dtype: int64
```

```
>>> prodf.iloc[ :, 2: ].count( )
Rice         6
Wheat        4
dtype: int64
```

Using iloc

```
>>> prodf.iloc[ 2:, : ].count(axis = 1)
Kerala       3
Punjab       4
Tripura      4
Uttar P.     4
dtype: int64
```

EXAMPLE 31 *Consider the DataFrame* ***dfmks*** *given in Reference 2.1. Write a program to count how many exams were conducted for each section ?*

SOLUTION

```
import pandas as pd
import numpy as np
:        # given dfmks created or loaded
print("Exams conducted for each section are :")
print(dfmks.iloc[ : , :].count())
```

Output

```
Exams conducted for each section are :
A    5
B    4
C    5
D    4
dtype: int64
```

EXAMPLE 32 *Consider the DataFrame* ***dfmks*** *given in Reference 2.1. Write a program to list subject wise about how many sections took exam of that subject ?*

SOLUTION

```
import pandas as pd
import numpy as np
:        # given dfmks created or loaded
print("Exams conducted for each section are :")
print(dfmks.iloc[ : , :].count(axis = 1))
```

Output

```
Exams conducted for each section are :
Acct     4
Eco      4
Eng      4
IP       3
Math     3
dtype: int64
```

Applying functions to a subset of the DataFrames

To apply a function on a subset of a dataframe, you need to use following :

With row index labels

```
<dataframe>.loc[ <start row> : <end row>, : <start column> : <end column>]
```

With row index numbers

```
<dataframe>.iloc[ <start row>: <end row>, <start column>: <end column>]
```

And then apply the function on it (*see* examples below).

EXAMPLE 33 *Consider DataFrame* **prodf** *given earlier. There are six states listed. Write a program to print the maximum produce of the middle 2 states listed (Kerala and Punjab).*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given prodf created or loaded
print("Maximum produce by Kerala and Punjab:")
print(prodf.iloc[2:4, : ].max(axis = 1))
```

Output

```
Maximum produce by Kerala and Punjab:
Kerala     2604.8
Punjab    16440.5
dtype: float64
```

2.6 Advanced Operations on DataFrame

After talking about descriptive statistics and other essential functions, in this section, we are going to talk about *three* advanced operations on dataframes : *pivoting*, *sorting* and *aggregation*. Let us begin with pivoting.

2.6.1 Pivoting

Before we discuss the pivoting functions of Python Pandas, you should know what pivoting is and why this is used. Firstly have a look at the following two views of the same data. Following table lists the answer choices by each of the students for an exam containing 5 MCQ questions.

Respondent	Question	Answer
Yash	Question 1	1
Yash	Question 2	4
Yash	Question 3	4
Yash	Question 4	5
Yash	Question 5	2
Khushi	Question 1	5
Khushi	Question 2	4
Khushi	Question 3	3
Khushi	Question 4	3
Khushi	Question 5	5
Misha	Question 1	3
Misha	Question 2	2
Misha	Question 3	5
Misha	Question 4	1
Misha	Question 5	2
⋮		

Now have a look at another table which also represents the same data :

Respondent	Question 1	Question 2	Question 3	Question 4	Question 5
Yash	1	4	4	5	2
Khushi	5	4	3	3	5
Misha	3	2	5	1	2
Japneet	5	2	2	1	4
Sana	1	5	5	3	3

Both these views list the same data ***in two different perspectives***. Carefully notice these views and you will find that :

(*i*) The second view is a summarised representation of the first view of the data.

(*ii*) The second view has converted row data to columns (Questions 1, Question2, ... are rows in the first view but columns in the second view).

In data handling, sometimes, you need to view your data in different perspectives just like you saw above. And to accomplish this, the technique that is used is called **pivoting**.

Pivoting is actually a summary technique that works on tabular data (*i.e.*, data in rows and columns). Pivoting technique rearranges the data from rows and columns, by possibly rotating rows and columns or by aggregating data from multiple sources, in a report form.

> PIVOTING
>
> "Pivoting technique rearranges the data from rows and columns, by possibly rotating rows and columns or by aggregating data from multiple sources, in a report form."

Got it ? No ? Do not worry. Come back to above paragraph after going through the following lines. Then the above paragraph will make full sense to you.

In simplest terms, the ***pivoting*** means summarising the data in a way to make understanding of descriptive data easier. *For example*, consider the following data.

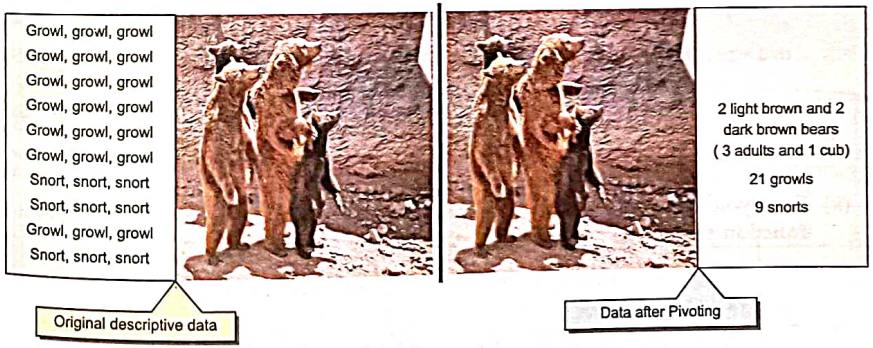


Figure 2.2 Impact of Pivoting : (*a*) Original, descriptive dataset (*b*) Summarised data by pivoting.

Consider above figure where you can see the representation of same data (consisting of an image and animal sounds) in two forms. While the Fig. 2.2(*a*) represents the original data in descriptive form, Fig. 2.2(*b*) represents the same descriptive data in a summarised form.

This is what pivoting does to extensive data sets. What Pivoting does to a real dataset can be depicted through following *two* points :

- it summarises extensive data.
- it rotates or pivots data by transforming rows into columns.

DATA PIVOTING

> "Data pivoting is a summarising technique to rearrange the columns and rows in a report so as to view data from different perspectives."

Now that you have an idea of what pivoting of data is, let us understand pivoting with the help of some real life data.

Say, an online tutoring company maintains its data about tutors and online classes in the table.

Tutor	Classes	Country
Tahira	28	USA
Gurjyot	36	UK
Anusha	41	Japan
Jacob	32	USA
Venkat	40	Brazil

From the adjacent table, it is easier to make out how many classes were taken by a tutor and how many classes were held for a country, as the above table is not a large table (only 5 rows). But, if you want to know how many classes were held for a **country** from a table containing 5000 entries or 50000 entries, would it still be easier ? No ?

So, for such a situation, there should be a way that transforms a specific column(s) into rows and presents information accordingly. Well, there is one – just do **pivoting**. (See below) :

2.6.1A Using pivot( ) Function

Let us now learn how you can perform pivoting using Pandas pivot( ) function.

(*i*) First of all, represent data in a Dataframe datastructure of Pandas :

```
import pandas as pd
d1 = {   'Tutor' : ['Tahira', 'Gurjyot', 'Anusha', 'Jacob', 'Venkat'],
         'Classes': [28, 36, 41, 32, 40],
         'Country' :['USA', 'UK', 'Japan', 'USA', 'Brazil']
     }
dfd = pd.DataFrame(d1)
```

```
In [13]: dfd
Out[13]:
   Classes Country   Tutor
0       28     USA  Tahira
1       36      UK Gurjyot
2       41   Japan  Anusha
3       32     USA   Jacob
4       40  Brazil  Venkat
```

To see pivot( ) function in action

Scan QR Code

(*ii*) Once you have represented your data in the form of a dataframe, you can pivot it using **function pivot( )** as per following syntax :

```
<dataframe>.pivot(index = <columnname>, columns = <columnname>,
values = <columnname>)
pivot(index = , columns = , values = )
```

Specify here the column which is to be treated as index (i.e., as rows)

Specify here the column, whose values will become columns

*Specify here the column, whose values are to be spread across the dataframe created as per specified **index** and **columns**.*

e.g.,

```
dfd.pivot(index = 'country', columns = 'Tutor', values = 'classes')
```


Thus, the following code will produce the result as shown below :

```
dfd.pivot(index = 'Country', columns = 'Tutor', values = 'Classes')
```

```
In [14]: dfd.pivot(index='Country', columns='Tutor', values='Classes')
Out[14]:
Tutor    Anusha  Gurjyot  Jacob  Tahira  Venkat
Country
Brazil      NaN      NaN    NaN     NaN    40.0
Japan      41.0      NaN    NaN     NaN     NaN
UK          NaN     36.0    NaN     NaN     NaN
USA         NaN      NaN   32.0    28.0     NaN
```

See, the *Tutor column*'s values have become columns (**columns = 'Tutor'**)

The Values spread across the table are from the column Classes (**values = 'Classes'**)

The index , *i.e.*, the rows are taken from the column (**Index = 'Country'**)

Notice , for USA , there are two entries as in original data, there were 2 tutors for USA

The result of *pivot( )* function has the *index-rows* as per the **index** argument, *columns* as per the values of **columns** argument and the *values* created from the **values** argument (*see* above). Cells in the pivoted table which do not have a matching entry in the original one are set with **NaN**.

The **pivot( )** function returns the result in the form of a newly created dataframe, *i.e.*, you may store the result in a dataframe (*see* below).

```
In [17]: df = dfd.pivot(index = 'Country', columns='Tutor', values='Classes')

In [18]: df
Out[18]:
Tutor    Anusha  Gurjyot  Jacob  Tahira  Venkat
Country
Brazil      NaN      NaN    NaN     NaN    40.0
Japan      41.0      NaN    NaN     NaN     NaN
UK          NaN     36.0    NaN     NaN     NaN
USA         NaN      NaN   32.0    28.0     NaN

In [19]: type(df)
Out[19]: pandas.core.frame.DataFrame
```

*result of pivot( ) stored in **df** label*

See the pivot() function result is returned in the form of a data frame

Tutor	Classes	Country
Tahira	28	USA
Gurjyot	36	UK
Anusha	41	Japan
Jacob	32	USA
Venkat	40	Brazil

Tutor / Country	Anusha	Gurjyot	Jacob	Tahira	Venkat
Brazil	NaN	NaN	NaN	NaN	40.0
Japan	41.0	NaN	NaN	NaN	NaN
UK	NaN	36.0	NaN	NaN	NaN
USA	NaN	NaN	32.0	28.0	NaN

```
dfd.pivot(index = 'Country', columns='Tutor', values='Classes')
```

Figure 2.3 Functioning of *pivot( )* function

So far so good. Now, let us try something new. Let us reapply the pivot( ), but this time let us interchange the rows and columns, *i.e.*, the **index** and **columns** arguments (*see* below) :

```
In [20]: dfd.pivot(index = 'Tutor', columns='Country', values='Classes')
Out[20]:
Country  Brazil  Japan    UK   USA
Tutor
Anusha      NaN   41.0   NaN   NaN
Gurjyot     NaN    NaN  36.0   NaN
Jacob       NaN    NaN   NaN  32.0
Tahira      NaN    NaN   NaN  28.0
Venkat     46.0    NaN   NaN   NaN
```

NOTE

The pivot( ) method creates a dataframe storing the pivoted table, which is a simplified version of the original data and only contains information as specified through the parameters to the ***pivot( )*** method.

You can skip the **values** argument, and if you skip the values argument, it will consider the rest of the columns (not mentioned in **index** and **columns** arguments) for values automatically :

```
In [21]: dfd.pivot(index = 'Tutor', columns='Country')
Out[21]:
        Classes
Country  Brazil Japan    UK   USA
Tutor
Anusha      NaN  41.0   NaN   NaN
Gurjyot     NaN   NaN  36.0   NaN
Jacob       NaN   NaN   NaN  32.0
Tahira      NaN   NaN   NaN  28.0
Venkat     46.0   NaN   NaN   NaN
```

With **pivot( )** function, remember that the arguments work as follows :

index	stores the column name about which the information is to be summarised (will become **rows** in the result).
columns	stores the column name whose data will become a column each in the summary information (will become **columns** in the result).
values	stores the column name whose data will be displayed for the index, column combination (will become **cells** in the result).

EXAMPLE **34** *Consider a DataFrame ndf as shown below :*

	Name	Sex	Position	City	Age	Projects
0	Rabia	F	Manager	Bangalore	30	13
1	Evan	M	Programmer	New Delhi	27	17
2	Jia	F	Manager	Chennai	32	16
3	Lalit	M	Manager	Mumbai	40	20
4	Jaspreet	M	Programmer	Chennai	28	21
5	Suji	F	Programmer	Bangalore	32	14

*Write a program to summarise how many **projects** are being handled by each **position** for each **city**?*

SOLUTION This problem requires pivoting. Let us try to identify the arguments of pivot( ) :

Information to be summarised *position* wise	position (index)
How many *projects* ?	projects (values)
For each *City*	city (columns) [City is a column in the DataFrame, whose each value will now become a column]

Now our program will look like :

```
import pandas as pd
import numpy as np
:       # given ndf created or loaded
print(ndf.pivot(index = 'Position', columns = 'City', values = 'Projects'))
```

Output

```
City        Bangalore  Chennai  Mumbai  New Delhi
Position
Manager          13.0     16.0    20.0        NaN
Programmer       14.0     21.0     NaN       17.0
```

Filling NA values

The above example listed NaN values where the projects value is missing. You can fill this NaN value with a different value, say 0, if you call fillna( ) function on pivot( )'s result *i.e.*, as :

```
<Df>.pivot( <all arguments of pivot>).fillna(<value>)
```

Following example illustrates this.

EXAMPLE **35** *Consider the ndf of previous example, write a program to summarise information Position wise, about how many projects are being handled by them for each city? Print zero (0) in case the data is missing for some column/row.*

SOLUTION

Function fillna( ) called on pivot( ) with value 0. Now 0 will be filled at the place of every NaN value

```
import pandas as pd
import numpy as np
:       # given ndf created or loaded
print(ndf.pivot(index = 'Position', columns = 'City', values = 'Projects').fillna(0))
```

Output

```
City        Bangalore  Chennai  Mumbai  New Delhi
Position
Manager          13.0     16.0    20.0        NaN
Programmer       14.02     1.0     0.0       17.0
```

Compare with previous example's output

2.6.1B Using pivot-table( ) Function

The above given tutors' data is for one quarter only. The online tutoring company has data for the entire year as shown below :

Quarter1

S.No.	Tutor	Classes	Quarter	Country
0.	Tahira	28	1	USA
1.	Gurjyot	36	1	UK
2.	Anusha	41	1	Japan
3.	Jacob	32	1	USA
4.	Venkat	40	1	Brazil

Quarter2

S.No.	Tutor	Classes	Quarter	Country
5.	Tahira	36	2	USA
6.	Gurjyot	40	2	USA
7.	Anusha	36	2	Japan
8.	Jacob	40	2	Brazil
9.	Venkat	46	2	USA

Quarter3

S.No.	Tutor	Classes	Quarter	Country
10.	Tahira	24	3	Brazil
11.	Gurjyot	30	3	USA
12.	Anusha	44	3	UK
13.	Jacob	40	3	Brazil
14.	Venkat	32	3	USA

Quarter4

S.No.	Tutor	Classes	Quarter	Country
15.	Tahira	36	4	Japan
16.	Gurjyot	32	4	Japan
17.	Anusha	36	4	Brazil
18.	Jacob	42	4	UK
19.	Venkat	38	4	USA

Let us represent the above data in a dataframe :

```
ontutD = { 'Tutor' : [ 'Tahira', 'Gurjyot', 'Anusha', 'Jacob', 'Venkat',
                       'Tahira', 'Gurjyot', 'Anusha', 'Jacob', 'Venkat',
                       'Tahira', 'Gurjyot', 'Anusha', 'Jacob', 'Venkat',
                       'Tahira', 'Gurjyot', 'Anusha', 'Jacob', 'Venkat' ],

'Classes':[28, 36, 41, 32, 40, 36, 40, 36, 40, 46, 24, 30, 44, 40, 32, 36, 32, 36, 42, 38 ],

'Quarter': [1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4 ],

'Country': [ 'USA', 'UK', 'Japan', 'USA', 'Brazil', 'USA', 'USA', 'Japan',
             'Brazil', 'USA', 'Brazil', 'USA', 'UK', 'Brazil', 'USA', 'Japan',
             'Japan', 'Brazil ', 'UK', 'USA' ]
}

df1 = pd.DataFrame(ontutD)
```

Data frame created from above dictionary that contains the online tutoring company data

```
In [6]: df1 = pd.DataFrame(ontutD)

In [7]: df1
Out[7]:
    Classes  Country  Quarter    Tutor
0        28      USA        1   Tahira
1        36       UK        1  Gurjyot
2        41    Japan        1   Anusha
3        32      USA        1    Jacob
4        40   Brazil        1   Venkat
5        36      USA        2   Tahira
6        40      USA        2  Gurjyot
7        36    Japan        2   Anusha
8        40   Brazil        2    Jacob
9        46      USA        2   Venkat
10       24   Brazil        3   Tahira
11       30      USA        3  Gurjyot
12       44       UK        3   Anusha
13       40   Brazil        3    Jacob
14       32      USA        3   Venkat
15       36    Japan        4   Tahira
16       32    Japan        4  Gurjyot
17       36   Brazil        4   Anusha
18       42       UK        4    Jacob
19       38      USA        4   Venkat
```


Let us try to use **pivot( )** for the above data frame :

```
In [22]: df1.pivot(index = 'Tutor', columns='Country')
Traceback (most recent call last):
  ...
  File "C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\reshape\reshape.py", line 154, in _make_selectors
    raise ValueError('Index contains duplicate entries,
ValueError: Index contains duplicate entries, cannot reshape
```

See, when we tried pivoting with the whole year data, it resulted in error, but it worked for single quarter data. The reason behind this error would be known to you if you carefully look at the data. Read on.

The index *i.e.*, the rows are specified as *'Tutor'* and the columns as *'Country'*. There are multiple entries of tutors which are having multiple countries.

Let us consider one Tutor say **Tahira's** entries.

Classes	Country	Quarter	Tutor
28	USA	1	Tahira
36	USA	2	Tahira
24	Brazil	3	Tahira
36	Japan	4	Tahira

If you try to create a row for the tutor Tahira from above data with columns as *Country*, how would you do? Something like :

	USA	Brazil	Japan
Tahira	28 or 36?	24	36

Multiple entries for a column for a single row CAUSES ERROR in pivot() function

NOTE

With pivot( ), if there are multiple entries for a column's value for the same values for index(row), it leads to error. Hence, before you use pivot( ), you should ensure that the data does not have rows with duplicate values for the specified columns.

As you can see there are multiple entries for USA for the same row *Tahira*. This leads to error in pivot( ) function.

But what if there is data with multiple entries for columns ? No pivoting possible ?

Well, no *pivot( )*ing possible but *pivot_table()*ing possible. Confused? Don't be. ☺ Just read on.

For data having multiple values for same row and column combination, you can use another pivoting function–the **pivot-table( )** function.

The **pivot_table( )** is also a pivoting function, which like *pivot()*, also produces a pivoted table, BUT it is different from the *pivot( )* function in following ways :

(*i*) It does not raise errors for multiple entries of a row, column combination.

(*ii*) It aggregates the multiple entries present for a row-column combination; you need to specify what type of aggregation you want (sum, mean etc.).

The syntax of simpler form of **pivot_table( ) function** is :

```
pandas.pivot_table(<dataframe>, values=None, index=None, columns=None, aggfunc='mean')
```

or

```
(<dataframe>.pivot_table( values = None, index = None, columns = None, aggfunc = 'mean')
```

where

the **index**	argument contains the column name for rows.
the **columns**	argument contains the column name for columns.
the **values**	argument contains the column names for data of the pivoted table.
the **aggfunc**	argument contains the function as per which data is to be aggregated, if skipped, it, **by default will compute the mean** of the multiple entries for the same row-column combination.

Consider the following example that now creates the pivoted table for the same, full-year's data of online tutoring company.

```
pivT = df1.pivot_table(index = 'Tutor', columns = 'Country', values = 'Classes')
```

Since no **aggfunc** is specified, it will **by default** compute the **mean** or the average of the multiple values having the same row and column combination.

And now you can display the new created dataframe **pivT** :

To see pivot_table function in action

Scan QR Code

```
In [8]: pivT
Out[8]:
Country  Brazil  Brazil   Japan    UK        USA
Tutor
Anusha      NaN    36.0    38.5  44.0        NaN
Gurjyot     NaN     NaN    32.0  36.0  35.000000
Jacob      40.0     NaN     NaN  42.0  32.000000
Tahira     24.0     NaN    36.0   NaN  32.000000
Venkat     40.0     NaN     NaN   NaN  38.666667
```

*Notice, for index **Tahira** and column **USA**, the mean of 2 values (28, 36) has been given here*

You can use any aggregate function for **aggfunc** argument (the aggregate functions you covered in previous section, *i.e., min, max, mode, median, mean, count* etc.).

Consider the following examples.

EXAMPLE 36 *Considering the tutoring company data, compute total classes per tutor.*

SOLUTION

```
In [60]: df1.pivot_table(index = 'Tutor', values = 'Classes', aggfunc = 'sum')
Out[60]:
         Classes
Tutor
Anusha       157
Gurjyot      138
Jacob        154
Tahira       124
Venkat       156
```


EXAMPLE 37 *Considering the tutoring company data, compute number of countries (count) per tutor.*

```
In [61]: df1.pivot_table(index = 'Tutor', values = 'Country', aggfunc = 'count')
Out[61]:
         Country
Tutor
Anusha         4
Gurjyot        4
Jacob          4
Tahira         4
Venkat         4
```

EXAMPLE 38 *Considering the tutoring company data, compute total classes by country.*

```
In [62]: df1.pivot_table(index = 'Country', values = 'Classes', aggfunc = 'sum')
Out[62]:
         Classes
Country
Brazil       144
Brazil        36
Japan        145
UK           122
USA          282
```

You may aggregate the results by more than one field (unlike the previous two scenarios where we aggregated the results based on a single field). For example, if you want to compute total classes, **tutor** and **country** wise, then you need to set the **index** argument to include both the field for which the sum id to be calculated, *e.g.,* see below :

EXAMPLE 39 *Considering the tutoring company data, compute total classes on two fields, tutor and country wise.*

```
In [64]: df1.pivot_table(index=['Tutor','Country'], values=['Classes'], aggfunc='sum')
Out[64]:
                Classes
Tutor   Country
Anusha  Brazil       36
        Japan        77
        UK           44
Gurjyot Japan        32
        UK           36
        USA          70
Jacob   Brazil       80
        UK           42
        USA          32
Tahira  Brazil       24
        Japan        36
        USA          64
Venkat  Brazil       40
        USA         116
```

EXAMPLE 40 *Considering the tutoring company data, compute average classes on two fields.*

```
In [65]: df1.pivot_table(index=['Tutor','Country'], values=['Classes'], aggfunc='mean')
Out[65]:
                 Classes
Tutor   Country
Anusha  Brazil  36.000000
        Japan   38.500000
        UK      44.000000
Gurjyot Japan   32.000000
        UK      36.000000
        USA     35.000000
Jacob   Brazil  40.000000
        UK      42.000000
        USA     32.000000
Tahira  Brazil  24.000000
        Japan   36.000000
        USA     32.000000
Venkat  Brazil  40.000000
        USA     38.666667
```

NOTE

The pivot() is used for pivoting without aggregation. Therefore, it can't deal with duplicate values for one index/column pair. For pivoting data with duplicate values, some kind of aggregation is required and which is done by pivot_table( ).

You can use any of these aggregate functions for **aggfunc** argument : *count, sum, mean, median, min, max, mode, abs, std, var, and other aggregate functions.*

2.6.2 Sorting Dataframe Values

Sorting refers to arranging values in a particular order. The values can be sorted on the basis of a specific column or columns and can be in ascending or descending order.

Pandas makes available **sort_values( )** function for this purpose, which can be used as per following syntax :

```
<dataframe>.sort_values(by, axis = 0, ascending = True, inplace = False,
                        kind = 'quicksort', na_position = 'last')
```

Parameters :

by	str or list of str ; Name or list of names to sort by. if axis is 0 or 'index' then by may contain index levels and/or column labels if axis is 1 or 'columns' then by may contain column levels and/or index labels
axis	{0 or 'index', 1 or 'columns'}, default 0 ; Axis to be sorted
ascending	bool or list of bool, default *True* ; Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the **by** argument.
inplace	bool, default *False* ; if *True*, perform operation in-place
na_position	{'first', 'last'}, default 'last' ; ***first*** puts NaNs at the beginning, ***last*** puts NaNs at the end.

The ***sort_values( )*** *sorts the values of a data frame by the values along either axis.*

Consider the following examples :

```
In [66]: df1.sort_values('Country')
Out[66]:
    Classes  Country  Quarter   Tutor
4        40   Brazil        1   Venkat
8        40   Brazil        2    Jacob
10       24   Brazil        3   Tahira
13       40   Brazil        3    Jacob
17       36   Brazil        4   Anusha
2        41    Japan        1   Anusha
16       32    Japan        4  Gurjyot
15       36    Japan        4   Tahira
7        36    Japan        2   Anusha
1        36       UK        1  Gurjyot
18       42       UK        4    Jacob
12       44       UK        3   Anusha
0        28      USA        1   Tahira
14       32      USA        3   Venkat
9        46      USA        2   Venkat
```

```
In [71]: df1.sort_values('Tutor')
Out[71]:
    Classes  Country  Quarter   Tutor
2        41    Japan        1   Anusha
17       36   Brazil        4   Anusha
7        36    Japan        2   Anusha
12       44       UK        3   Anusha
1        36       UK        1  Gurjyot
6        40      USA        2  Gurjyot
11       30      USA        3  Gurjyot
16       32    Japan        4  Gurjyot
3        32      USA        1    Jacob
8        40   Brazil        2    Jacob
18       42       UK        4    Jacob
13       40   Brazil        3    Jacob
0        28      USA        1   Tahira
5        36      USA        2   Tahira
10       24   Brazil        3   Tahira
```

```
In [67]: df1.sort_values(['Country', 'Tutor'])
Out[67]:
    Classes  Country  Quarter   Tutor
8        40   Brazil        2    Jacob
13       40   Brazil        3    Jacob
10       24   Brazil        3   Tahira
4        40   Brazil        1   Venkat
17       36   Brazil        4   Anusha
2        41    Japan        1   Anusha
7        36    Japan        2   Anusha
16       32    Japan        4  Gurjyot
15       36    Japan        4   Tahira
12       44       UK        3   Anusha
1        36       UK        1  Gurjyot
18       42       UK        4    Jacob
6        40      USA        2  Gurjyot
11       30      USA        3  Gurjyot
```

Values sorted Country wise and within Country, Tutor-wise

```
In [68]: df1.sort_values(by =['Tutor', 'Country'])
Out[68]:
    Classes  Country  Quarter   Tutor
17       36   Brazil        4   Anusha
2        41    Japan        1   Anusha
7        36    Japan        2   Anusha
12       44       UK        3   Anusha
16       32    Japan        4  Gurjyot
1        36       UK        1  Gurjyot
6        40      USA        2  Gurjyot
11       30      USA        3  Gurjyot
8        40   Brazil        2    Jacob
13       40   Brazil        3    Jacob
18       42       UK        4    Jacob
3        32      USA        1    Jacob
10       24   Brazil        3   Tahira
15       36    Japan        4   Tahira
```

Values sorted Tutor wise and within Tutor, country wise

```
In [72]: df1.sort_values(by =['Tutor', 'Country'], ascending = False)
Out[72]:
    Classes  Country  Quarter   Tutor
9        46      USA        2   Venkat
14       32      USA        3   Venkat
19       38      USA        4   Venkat
4        40   Brazil        1   Venkat
0        28      USA        1   Tahira
5        36      USA        2   Tahira
15       36    Japan        4   Tahira
10       24   Brazil        3   Tahira
3        32      USA        1    Jacob
18       42       UK        4    Jacob
8        40   Brazil        2    Jacob
13       40   Brazil        3    Jacob
6        40      USA        2  Gurjyot
11       30      USA        3  Gurjyot
```

Values sorted in descending order

NOTE

Please note that we are not giving the complete full syntax of the functions in this chapter. We are only giving the syntax for basic and minimal functioning. Discussing full syntax of these functions is beyond the scope and nature of the book (being a textbook, not a manual)

- You can use parameter **inplace**, if you want to store the sorted data in the dataframe itself (in place).
- Use **na_position** parameter to specify the position of NaN values, which is by default **last** but can be selected as **first**.

EXAMPLE 41 *Consider the prodf DataFrame (given in the beginning of section 2.4) storing crop production statistics of various states. Write a program to sort the values of prodf in the order of Wheat production by them.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given prodf created or loaded
print(prodf.sort_values(by = ['Wheat']))
```

Output

	Fruits	Pulses	Rice	Wheat
Tripura	44.1	23.2	814.6	0.5
Gujrat	11950.0	818.0	1930.0	2737.0
Punjab	7152.0	33.0	11586.2	16440.5
Uttar P.	140169.2	2184.4	13754.0	30056.0
Andhra P.	7830.0	931.0	7452.4	NaN
Kerala	113.1	1.7	2604.8	NaN

EXAMPLE 42 *Consider the prodf DataFrame (given in the beginning of section 2.4) storing crop production statistics of various states. Write a program to sort the values of prodf in the order of Fruits production by them, in descending order.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given prodf created or loaded
print(prodf.sort_values(by = ['Fruits'], ascending = False))
```

Output

	Fruits	Pulses	Rice	Wheat
Uttar P.	140169.2	2184.4	13754.0	30056.0
Gujrat	11950.0	818.0	1930.0	2737.0
Andhra P.	7830.0	931.0	7452.4	NaN
Punjab	7152.0	33.0	11586.2	16440.5
Kerala	113.1	1.7	2604.8	NaN
Tripura	44.1	23.2	814.6	0.5

Sorting on the Basis of Index

We can use **sort_index( )** function to sort a dataframe to sort by row index or names as per following sysntax :

```
DataFrame.sort_index(ascending=True, inplace=False)
```

By specifying the value as **ascending = False** (True by default), it will sort the DataFrame in descending order of indexes.

Consider following example, which is sorting by index.

EXAMPLE 43 *Consider the prodf DataFrame storing crop production statistics of various states. Write a program to sort the values of prodf in the order of the indexes, in descending order.*

SOLUTION

```
import pandas as pd
import numpy as np
:       # given prodf created or loaded
print(prodf.sort_index(ascending = False))
```

Output

	Fruits	Pulses	Rice	Wheat
Uttar P.	140169.2	2184.4	13754.0	30056.0
Tripura	44.1	23.2	814.6	0.5
Punjab	7152.0	33.0	11586.2	16440.5
Kerala	113.1	1.7	2604.8	NaN
Gujrat	11950.0	818.0	1930.0	2737.0
Andhra P.	7830.0	931.0	7452.4	NaN

2.6.3 Aggregation

With large amounts of data, most often we need to aggregate data so as to analyse it effectively. Pandas offers many aggregate functions, using which you can aggregate data and get summary statistics of the data.

Most commonly used summary statistics are : the *mean* and *standard deviation*. Other than these other aggregate functions are also used. The following table summarizes some other built-in Pandas aggregations. There may be other functions as well but we shall limit our discussion only to these :

S.No.	Aggregation	Description
1.	count( )	Total number of items
2.	sum( )	Sum of all items
3.	mean( ), median( )	Mean and median
4.	min( ), max( )	Minimum and maximum
5.	std( ), var( )	Standard deviation and variance
6.	mad( )	Mean absolute deviation

You have already worked with mostly all of the mentioned functions except for **mad( )** and thus, we are covering only **mad( )** below.

The mad( ) function

The **mad( )** function is used to calculate the **mean absolute deviation** of the values for the requested axis. The *Mean Absolute Deviation (MAD)* of a set of data is the average distance between each data value and the mean. You can use **mad( )** as per following syntax :

```
<dataframe>.mad(axis = None, skipna = None)
```

Parameters :

axis {index (0), columns (1)} default 0

skipna boolean, default True; Exclude NA/null values. If an entire row/column is NA, the result will be NA

Consider the **prodf** DataFrame storing crop production statistics of various states. Now consider the following examples that calculate MAD for DataFrame **prodf** on both the axes.

```
>>> prodf.mad()
Fruits   37430.933333
Pulses     645.916667
Rice      4573.866667
Wheat    10939.750000
dtype: float64
```

MAD calculated along **axis 0**, *i.e.*, for all the columns considering data across rows

MAD calculated along **axis 1** i.e., or all the rows considering data across the columns

```
>>> prodf.mad( axis = 1)
Andhra P.    2982.311111
Gujrat       3795.625000
Kerala       1132.177778
Punjab       5210.425000
Tripura       297.000000
Uttar P.    46814.150000
dtype: float64
```

2.7 Handling Missing Data

Pandas library is designed to deal with huge amounts of data or big data. In such volumes of data, there may be some values which have *NA values* such as *NULL* or *NaN* or *None* values. These are the values that cannot participate in computation constructively. These values are known as **missing values**. For instance, ranking of colleges is based on many parameters such as *infrastructure, faculty, research,* and *placements* data etc. for past 10 years etc. Now for newer colleges, placement data may not be available. Hence in such a dataset, the placement data will be missing for newer colleges and this will be signified by some empty value – this is *missing value*. In some datasets, missing data may appear as *?* or *blank* or *0* or *–1* also.

MISSING VALUES

> "Missing values are the values that cannot contribute to any computation or we can say that missing values are the values that carry no computational significance."

Some other real life examples where you may find missing data in a dataset may be situations like : a 2 bedroom house wouldn't include the size of the third bedroom, it will be a missing value in its dataset ; in a survey, some respondent may choose not to share their income, then this will also be a missing value in that data set.

You must be thinking – **Why handle missing data ?** The simple answer is that data science is all about inferring accurate results/predictions from the given data and if the missing values are not handled properly, then it may affect the overall result and the researcher may end up drawing an inaccurate inference about the data. The performance the predictive model based on data having means values can hugely be impacted if the missing values are high in number and are not appropriately handled.

Say, for example, in a dataset where some employees' number-of-project-participation-data is shown as –1 which means data not yet received. Now say if you as researcher create a predictive model that predicts the salary based on number-of-projects-participation, then these values will seriously hamper the accuracy of your model. Thus, it is very important to know about the missing values in your data set and also how to handle these missing values as per your requirements.

You can handle missing data in many ways, most common ones are :

(*i*) Dropping missing data

(*ii*) Filling missing data (Imputation)

Let us talk about these BUT before that you should know how to detect or filter the missing data. Also, you can apply the way to detect /filter missing data and methods to handle missing data on both Series as well as Dataframe objects. For detecting and handling missing data, we shall use the following sample objects having some missing data : a series object namely **dtSer** and a dataframe **patientDF** (see Fig. 2.4)

```
In [58]: patientDF
Out[58]:
  First_name Last_name   Age  Sex  ValBefore  ValAfter
0      Jivin     Arora  42.0    m        4.0      25.0
1        NaN       NaN   NaN  NaN        NaN       NaN
2       Ruhi      Hyat  36.0    f        NaN       NaN
3     Raunaq     Singh  24.0    m        2.0      62.0
4       Mike      John  73.0    f        3.0      70.0
```

This dataframe has a row with NaN values in all its columns and some other rows also have some NaN values

This series has some NaN values in it

```
In [60]: dtSer
Out[60]:
0    11.0
1     NaN
2    30.5
3     NaN
4    17.0
dtype: float64
```

Figure 2.4 Some Pandas objects will missing data

2.7.1 Detecting/Filtering Missing Data

You can use **isnull()** to detect missing values in a Pandas object; it returns ***True*** or ***False*** for each values in a Pandas object if it is a missing value or not.

It can be used as :

```
<PandaObject>.isnull()
```

You can place Series or dataframe object at <PandaObject>

<PandaObject> means it is applicable to both Series as well as *Dataframe* objects.

Consider the following outputs :

```
In [61]: dtSer.isnull()
Out[61]:
0    False
1     True
2    False
3     True
4    False
dtype: bool
```

The **isnull()** returns *True* or *False* for each value in pandas object if it is a missing value or not

```
In [62]: patientDF.isnull()
Out[62]:
   First_name  Last_name    Age    Sex  ValBefore  ValAfter
0       False      False  False  False      False     False
1        True       True   True   True       True      True
2       False      False  False  False       True      True
3       False      False  False  False      False     False
4       False      False  False  False      False     False
```

If you want to filter data which is not a missing value *i.e.*, non-null data, then you can use following for Series object :

```
<Series>[filter condition>]
```

For dataframe objects, it is not that simple, the above method will not work for dataframes. For a dataframe object, you need to use methods to handle missing values like dropna( ) etc., which we are covering next.

For example

```
In [63]: dtSer[dtSer.notnull()]
Out[63]:
0    11.0
2    30.5
4    17.0
dtype: float64
```

See, it has listed only non-null data

2.7.2 Handling Missing Data – Dropping Missing Values

As mentioned earlier you can either drop missing values or fill missing values. To drop missing values you can use **dropna( )** in following *three* ways :

(*a*) <PandaObject>.dropna( ). This will drop all the rows that have NaN values in them, even row with a single *NaN* value in it, *e.g.*, see below :

```
In [66]: ser1 = dtSer.dropna()

In [67]: ser1
Out[67]:
0    11.0
2    30.5
4    17.0
dtype: float64
```

See the NaN values of both objects have been removed

```
In [71]: pdf1 = patientDF.dropna()

In [72]: pdf1
Out[72]:
  First_name Last_name   Age Sex  ValBefore  ValAfter
0      Jivin     Arora  42.0   m        4.0      25.0
3    Raunaq     Singh  24.0   m        2.0      62.0
4       Mike      John  73.0   f        3.0      70.0
```

(*b*) <DF>.dropna(how = 'all'). With argument **how = 'all'** (see 'all' is in quotes), it will drop only those rows that have all NaN values, *i.e.*, no value is non-null in those rows, *e.g.*, see below :

```
In [75]: pdf2 = patientDF.dropna(how ='all')

In [76]: pdf2
Out[76]:
  First_name Last_name   Age Sex  ValBefore  ValAfter
0      Jivin     Arora  42.0   m        4.0      25.0
2       Ruhi      Hyat  36.0   f        NaN       NaN
3    Raunaq     Singh  24.0   m        2.0      62.0
4       Mike      John  73.0   f        3.0      70.0
```

Notice this time row with index 2 is not dropped because it has some non-null values in it BUT row with index1 has been deleted as it has all values as null or NaN values

(*c*) <DF>.dropna(axis = 1). With argument **axis = 1**, will drop columns that have any *NaN* values in them, *e.g.*, see below :

```
In [77]: pdf3 = patientDF.dropna(axis = 1)

In [78]: pdf3
Out[78]:
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4]
```

See it has removed all the columns because each column had at least one NaN value, hence it removed all

Using argument **how = 'all'** along with argument **axis = 1** will drop columns with all NaN values.

2.7.3 Handling Missing Data – Filling Missing Values

Though **dropna( )** removes the null values, but you also lose other non-null data with it too. To avoid this, you may want to fill the missing data with some appropriate value of your choice. For this purpose you can use **fillna( )** in following ways :

(*a*) <PandaObject>.fillna(<n>). This will fill all NaN values in a Pandas object with the given <n> value, *e.g.*, see below :


```
In [79]: ser4 = dtSer.fillna(0)

In [80]: ser4
Out[80]:
0    11.0
1     0.0
2    30.5
3     0.0
4    17.0
dtype: float64
```

See all NaN values have been filled with given value 0

```
In [81]: pdf4 = patientDF.fillna(0)

In [82]: pdf4
Out[82]:
  First_name Last_name   Age Sex  ValBefore  ValAfter
0      Jivin     Arora  42.0   m        4.0      25.0
1          0         0   0.0   0        0.0       0.0
2       Ruhi      Hyat  36.0   f        0.0       0.0
3    Raunaq     Singh  24.0   m        2.0      62.0
4       Mike      John  73.0   f        3.0      70.0
```

(*b*) **Using dictionary with fillna( ) to specify fill values for each column separately.** You can create a dictionary that defines fill values for each of the columns. And then you can pass this dictionary as an argument to fillna( ), Pandas will fill the specified value for each column defined in the dictionary. It will leave those columns untouched or unchanged that are not in the dictionary. The syntax to use this format of fillna( ) is :

```
<DF>.fillna( <dictionary having fill values for columns> )
```

For example, see below :

```
In [87]: fillValues = {
    ...: 'First_name' : 'x',
    ...: 'Last_name' : 'x',
    ...: 'Age' : 0,
    ...: 'Sex' : 'x',
    ...: 'ValBefore' : -999,
    ...: 'ValAfter' : -999
    ...: }
```

Dictionary with fill values for columns defined

Missing values filled as per passed dictionary

```
In [88]: pdf5 = patientDF.fillna(fillValues)

In [89]: pdf5
Out[89]:
  First_name Last_name   Age Sex  ValBefore  ValAfter
0      Jivin     Arora  42.0   m        4.0      25.0
1          x         x   0.0   x     -999.0    -999.0
2       Ruhi      Hyat  36.0   f     -999.0    -999.0
3    Raunaq     Singh  24.0   m        2.0      62.0
4       Mike      John  73.0   f        3.0      70.0
```

You can directly specify fill values in dictionary from inside fillna( ) and if the dictionary does not have all the column names then only the columns specified will be filled as per given values, other columns will be untouched :

```
In [90]: pdf6 = patientDF.fillna( {'Age':0, 'Sex': 'x'} )

In [91]: pdf6
Out[91]:
  First_name Last_name   Age Sex  ValBefore  ValAfter
0      Jivin     Arora  42.0   m        4.0      25.0
1        NaN       NaN   0.0   x        NaN       NaN
2       Ruhi      Hyat  36.0   f        NaN       NaN
3    Raunaq     Singh  24.0   m        2.0      62.0
4       Mike      John  73.0   f        3.0      70.0
```

Dictionary with fill values can be defined inside fillna() itself

See, only the columns for which fill values are given, have been filled, rest are left unchanged.

NOTE

Please note that the functions covered in this chapter have lots of other options and functionality. We are not covering all the options of these functions here as this chapter is not a manual. Only that much functionality is covered which is as per the syllabus defined or something that is needed to explain that functionality.

EXAMPLE **44** *Consider the DataFrame dfmks having some missing values (shown as NaN) as shown below :*

	A	B	C	D
Acct	99	94.0	92	97.0
Eco	90	94.0	92	97.0
Eng	95	89.0	91	89.0
IP	94	NaN	99	95.0
Math	97	100.0	99	NaN

Write a program to fill the missing values with value 0.

SOLUTION

```
import pandas as pd
import numpy as np
:        # given dfmks created or loaded
dfmks = dfmks.fillna(0)
print("Dataframe after filling missing values")
print(dfmks)
```

Output

```
Dataframe after filling missing values
      A     B    C     D
Acct  99   94.0   92  97.0
Eco   90   94.0   92  97.0
Eng   95   89.0   91  89.0
IP    94    0.0   99  95.0
Math  97  100.0   99   0.0
```

Filling missing values from another DataFrame

You can also fill the missing values from a similar DataFrame, *i.e.*, from a DataFrame having the same index and columns. If you pass another DataFrame's name in **fillna( )**, then it will take corresponding cell's values for missing values.

That is,

```
Df1.fillna(Df2)
```

Will fill missing values of DataFrame df1 from the corresponding cells of Df2. But the condition here is that both the dataframes should have similar structure, only their values differ.

Refer to solved problem 27, which is doing the same.

There are even some other advanced, and of course better ways of handling missing data but covering those here is beyond the scope of this book.

Check Point 2.1

1. Name some descriptive statistic functions used with dataframes.
2. To calculate statistical values for each row, the **axis** argument should be ____ ?
3. What are quantile and quartiles ?
4. What does quantile( ) do ?
5. Name pivoting functions available for dataframes.
6. What is the use of **aggfunc** argument in pivot_table( ) ?
7. What does hist( ) do ?
8. What are missing values in a dataset ?
9. Write two methods generally used for handling missing values ?

2.8 Combining DataFrames

While working, we may encounter situations where we have access to two or more similar dataframes containing more or less similar data. To make a decision based on those multiple dataframes having similar data, you would want a way that combines the two dataframes in some efficient way so that missing data in one dataframe is picked from another similar dataframe or dissimilar entries from two dataframes are combined to form one dataframe. In this section, we are going to talk about some ways *concat( ), join( )* and *merge( )* to combine multiple dataframes.

2.8.1 Combining Dataframes using concat( )

Let us understand the working of **concat( )** to combine dataframes with the help of following *three* dataframes (**sd1, sd2** and **sd3**) :

```
>>> sd1
   Subject_id First_Name Last_Name
0           1      Sarah     Kapur
1           2       Reet       Kaur
2           3     Robert    Douglas
3           4    Shahida       Ali
4           5        Jia    Gopalan
```

```
>>> sd2
   Subject_id First_Name Last_Name
0           4       Aani      Sinha
1           5       Asma     Ansari
2           6      Billy       John
3           7       Rudy      Monga
4           8  Prabhnoor      Singh
```

```
>>> sd3
   Subject_id  Test_id
0           1       51
1           2       15
2           3       15
3           4       61
4           5       16
5           7       14
6           8       15
7           9        1
8          10       61
9          11       16
```

The **concat( )** can concatenate two dataframes along the rows or along the columns. This method is useful if the two dataframes have similar structures. It is used as per following syntax :

```
pd.concat([<df1>, <df2>])
pd.concat([<df1>, <df2>], ignore_index = True)
pd.concat([<df1>, <df2>], axis = 1 )
```

The dataframes to be added are given as a sequence, e.g., a list

If you skip the **axis = 1** argument, it will join the two dataframes along the rows, *i.e.*, the result will be the union of rows from both the dataframes, *e.g.*, carefully look at the command given below and the result it produces :

To see concat( ) in action

Scan QR Code

```
>>> sd4 = pd.concat ( [ sd1, sd2) ]
>>> sd4
   Subject_id First_Name Last_Name
0           1      Sarah     Kapur
1           2       Reet       Kaur
2           3     Robert    Douglas
3           4    Shahida       Ali
4           5        Jia    Gopalan
0           4       Aani      Sinha
1           5       Asma     Ansari
2           6      Billy       John
3           7       Rudy      Monga
4           8  Prabhnoor      Singh
```

indexes of sd1

See, row indexes are also concatenated

indexes of sd2

See this concat( ) function has simply concatenated all the rows from both the dataframes (**sd1** and **sd2**), including the row indexes. It has also concatenated the row indexes too.

As you can see above that by default **concat( )** has concatenated all the rows from both the dataframes (*sd1* and *sd2*), including the row indexes; it has also concatenated the row indexes too.

But if you do not want this mechanism for row indexes and want to have new row indexes generated from 0 to $n-1$, then you can give argument **ignore_index = True** which the following example is doing.

EXAMPLE 45 *Write a program to concat above used dataframes sd1 and sd2 along the rows but ensure that the new indexes are generated in the resultant dataframe.*

SOLUTION

```
import pandas as pd
  :   # sd1 and sd2 created or loaded
sd3 = pd.concat(sd1, sd2, ignore_index = True)
print(sd3)
```

The output produced will be as follows. (*Notice new indexes are generated in the resultant dataframe*)

Now new indexes are generated for the resultant dataframe

```
  Subject_id First_Name Last_Name
0          1      Sarah     Kapur
1          2       Reet       Kaur
2          3     Robert   Douglas
3          4    Shahida       Ali
4          5        Jia   Gopalan
5          4       Aani     Sinha
6          5       Asma    Ansari
7          6      Billy      John
8          7       Rudy     Monga
9          8  Prabhnoor     Singh
```

By default, concat( ) concatenates along the row. To concatenate along the column, you can give argument **axis = 1** in the **concat( )** function as shown below :

```
>>> sd5 = pd.concat( [sd1, sd2] , axis = 1)

>>> sd5

  Subject_id First_Name Last_Name  Subject_id First_Name Last_Name
0          1      Sarah     Kapur           4       Aani     Sinha
1          2       Reet       Kaur          5       Asma    Ansari
2          3     Robert   Douglas           6      Billy      John
3          4    Shahida       Ali           7       Rudy     Monga
4          5        Jia   Gopalan           8  Prabhnoor     Singh
```

See, with axis =1 argument, it has now concatenated along the columns

2.8.2 Combining Dataframes using join( )

The **join( )** method used by DataFrames basically creates a new dataframe from two dataframes by **joining** their rows. Here, it is important for you to understand what **joining** means.

Joining of two dataframes means creating a third dataframe from the two dataframes in one of the following ways. For understanding purposes, we shall join the following two dataframes Df1 and Df2.

Df1

	Name
1	Harini
2	Dave
3	Simrat
4	Saqib

In these dataframes Df1 and Df2, only two indexes are common – index *1* and index *3*

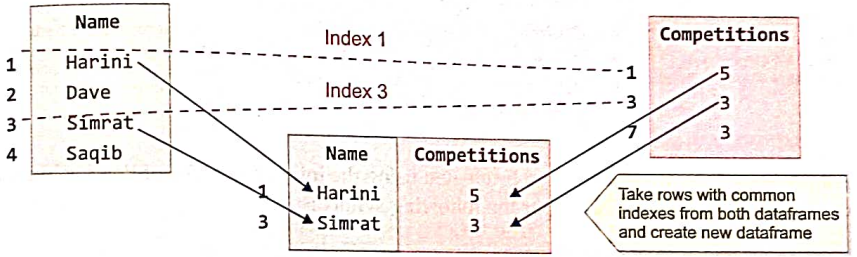
Df2

	Competitions
1	5
3	3
7	3

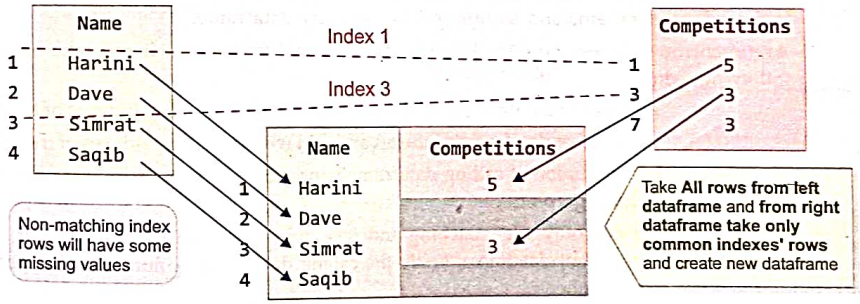
To see join( ) in action

Scan QR Code

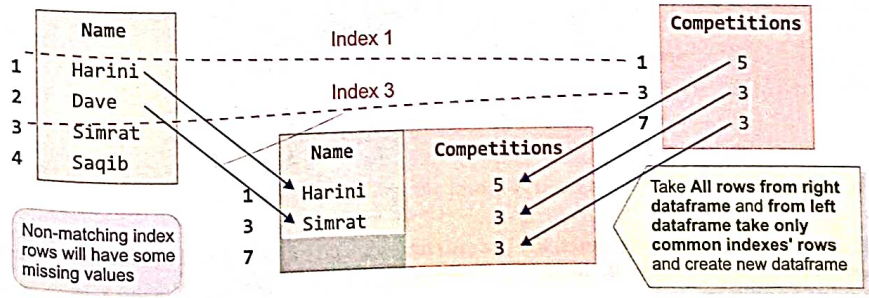
(*i*) Take the **rows having common indexes** from both the dataframes and create a new dataframe from it. This type of join is called **Inner join**, *e.g.*,



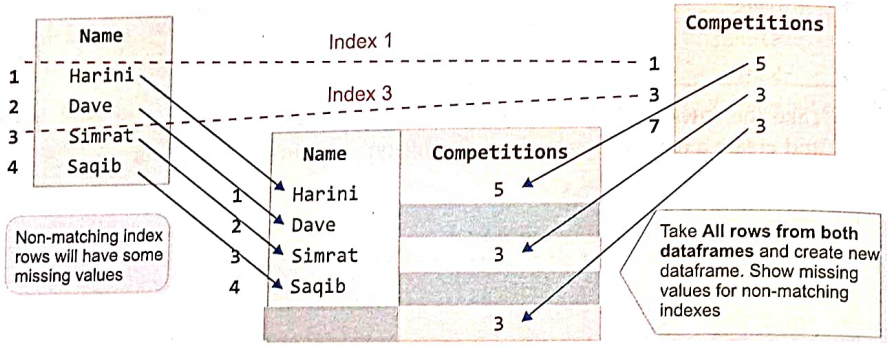
(*ii*) Take **all the rows from the left (first) dataframe and join with it only those rows from the second dataframe that have common indexes** as dataframe1 and create a new dataframe from it. This type of join is called **Left join**, *e.g.*,



(*iii*) Take **all the rows from the right (second) dataframe and join with it only those rows from the first dataframe that have common indexes** as dataframe2 and create a new dataframe from it. This type of join is called **Right join**, *e.g.*,



(*iv*) **Take all rows from both the dataframes and join them.** There will be values for all the columns only in the matching indexes' rows and non-matching index rows will have some missing values. This type of join is called **Outer join**, *e.g.*,



All the above mentioned joins can be created using the **join( )** functions of Pandas library. You can use the **join( )** function as per the following syntax :

```
<DataFrame1>.join(<DataFrame2>, [how = 'left'])[1]
```

Where

- <DataFrame1> and <DataFrame2> are the two dataframes being joined. The dataframe that calls the join( ) is the first dataframe (the **calling dataframe**). Thus, in this case *DataFrame1* is the calling dataframe and *Dataframe2* is the **other dataframe.**
- Argument how defines how the joined dataframe will be created. It can take one of the following values :

  `Left` : use calling dataframe's indexes and join with matching indexes of the other.

  `Right` : use other dataframe's indexes and join with matching indexes of the calling.

  `Outer` : form union of calling dataframe's indexes with other's indexes, and sort it lexicographically.

  `Inner` : have only the matching indexes from both the dataframes but in the resultant dataframe, firstly the calling dataframe's columns will appear and then other's

  If skipped, the how argument considers **'left'** as the default value.

Let us now use it practically by considering the same dataframes with which we have explained the join operation earlier, *i.e.*

```
>>> df1
     Name
1    Harini
2    Dave
3    Simrat
4    Saqib
```

```
>>>df2
   Competition
1        5
3        3
7        3
```

1. *Joins are created by default on indexes. But you can create joins on columns as well by specifying an additional argument on = <column name> in the join() function.*

Please note that in the joined dataframe :

- for non-matching indexes, missing values will be represented as NaN.
- the **join( )** function expects the column names to be different in both the dataframes. Common column names will lead to **ValueError.**

Now, consider the following examples of **join( ).**

(*i*) By default **join( )** will create the LEFT JOIN, *i.e.*, when no **how** argument is given or argument **how = 'left'** is given.

```
>>> df1.join(df2)
     Name     Competitions
1    Harini      5.0
2    Dave        NaN
3    Simrat      3.0
4    Saqib       NaN
```

Left join
All rows from left dataframe and only common indexes' rows taken from right dataframe

```
>>> df1.join(df2, how = 'left')
     Name     Competitions
1    Harini      5.0
2    Dave        NaN
3    Simrat      3.0
4    Saqib       NaN
```

Non matching indexes get NaN values in the joined dataframe

(*ii*) Specify **how = 'inner'** to get the INNER JOIN.

```
>>> df1.join(df2, how = 'inner')
     Name     Competitions
1    Harini      5
3    Simrat      3
```

(*iii*) Specify **how = 'outer'** to get the OUTER JOIN.

```
>>> df1.join(df2, how = 'outer')
     Name     Competitions
1    Harini      5.0
2    Dave        NaN
3    Simrat      3.0
4    Saqib       NaN
7    NaN         3.0
```

Non matching indexes get NaN values in the joined dataframe

In the joined dataframe, the order of columns is as :
Firstly the columns of first dataframe followed by the columns of the second datafranme.
The dataframe calling the join() is treated as the first dataframe.

(*iv*) Specify **how = 'right'** to get the RIGHT JOIN .

```
>>> df1.join(df2, how = 'right')
     Name     Competitions
1    Harini      5
3    Simrat      3
7    NaN         3
```

Non matching indexes get NaN values in the joined dataframe

Please note that the dataframe calling the **join( )** is treated as the first or the left dataframe and accordingly the result of LEFT or RIGHT joins is obtained. That means, the *right join* with *dataframe2* as the calling dataframe will be similar to the *left join* of *dataframe1* as the calling dataframe BUT the **order of the columns will be changed.**

For example, look at the following two **join( )** statements and their results.

```
>>> df1.join(df2, how = 'left')
```

	Name	Competitions
1	Harini	5.0
2	Dave	NaN
3	Simrat	3.0
4	Saqib	NaN

Notice that even though the values in both the results are the same, the order of the columns is different.

```
>>> df2.join(df1, how = 'right')
```

	Competitions	Name
1	5.0	Harini
2	NaN	Dave
3	3.0	Simrat
4	NaN	Saqib

Non matching indexes get NaN values in the joined dataframe

Consider below given two dataframes namely *saleDf1* and *saleDf2* to be used in the coming *two* examples.

```
>>> saleDf1
```

	Target	Sales
zoneA	56000	58000
zoneB	70000	68000
zoneC	75000	78000
zoneD	60000	61000

```
>>> saleDf2
```

	Name	Product
zoneA	Purv	Oven
zoneB	Paschim	AC
zoneC	Kendriya	AC
zoneD	Dakshin	Oven
zoneE	Uttar	TV
zoneF	Rural	Tubewell

EXAMPLE **46** *Write a program to create a dataframe namely* ***newDf*** *that stores all the rows of* ***saleDf2*** *and only the matching indexes' rows from* ***saleDf1****.*

SOLUTION

```
import pandas as pd
:          # dataframes saleDf1 and saleDf2 created or loaded here
newDf = saleDf2.join(saleDf1)        # or newDf = saleDf2.join(saleDf1, how ='left')
print(newDf)
```

The output produced by the above code is :

Output

	Name	Product	Target	Sales
zoneA	Purv	Oven	56000.0	58000.0
zoneB	Paschim	AC	70000.0	68000.0
zoneC	Kendriya	AC	75000.0	78000.0
zoneD	Dakshin	Oven	60000.0	61000.0
zoneE	Uttar	TV	NaN	NaN
zoneF	Rural	Tubewell	NaN	NaN

Non matching indexes get NaN values in the joined dataframe

EXAMPLE **47** *Write a program to create a dataframe namely* ***nDf*** *that stores only the matching indexes' rows from both* ***saleDf2*** *and* ***saleDf1****.*

SOLUTION

```
import pandas as pd
:          # dataframes saleDf1 and saleDf2 created or loaded here
nDf = saleDf2.join(saleDf1, how = 'inner')
print(nDf)
```

The output produced by the above code is :

Output

	Name	Product	Target	Sales
zoneA	Purv	Oven	56000	58000
zoneB	Paschim	AC	70000	68000
zoneC	Kendriya	AC	75000	78000
zoneD	Dakshin	Oven	60000	61000

Joining on a column

The **join( )** function provides an **on** ***argument*** in which you can provide a column name of the first dataframe, *e.g.,*

```
Df1.join(Df2, on = <column name of Df1> )
```

When you specify the **on** ***argument,***

- the left dataframe's mentioned column's values will be matched with indexes of the second dataframe

Recall that by default indexes of two dataframes are matched but with **on** ***argument,*** the left dataframe's column's values are matched with the indexes of the right dataframe. Only the left dataframe's column is considered for matching ; the indexes of the right dataframe are matched. For example,

```
>>> Df1
```

	Cust_id	Product
0	1	Oven
1	2	AC
2	3	AC
3	4	Speaker
4	5	Tablet
5	6	Smartphone

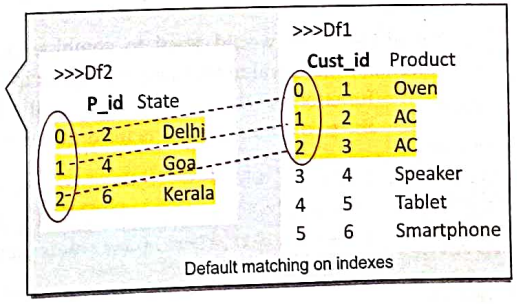
```
>>> Df2
```

	P_id	State
0	2	Delhi
1	4	Goa
2	6	Kerala

If you give the statement as :

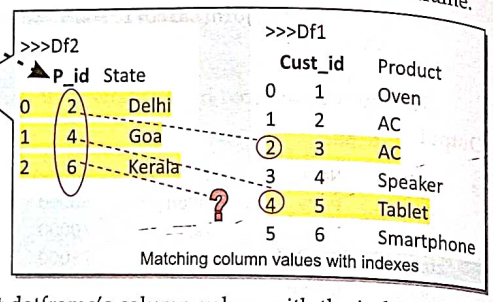
```
>>> Df2.join(Df1)
```

	P_id	State	Cust_id	Product
0	2	Delhi	1	Oven
1	4	Goa	2	AC
2	6	Kerala	3	AC



The above output is the result of default joining behaviour (*left join* on the basis of index matching). But when you specify a column of the first dataframe (the dataframe calling the **join( )**) in **on *argument***, then that column's values are matched with the indexes of the second dataframe.

```
>>> Df2.join(Df1, on = "P_id")
   P_id   State   Cust_id  Product
0   2     Delhi    3.0      AC
1   4     Goa      5.0      Tablet
2   6     Kerala   NaN      NaN
```



But the **join( )** function can join only the left datframe's column values with the indexes of the right dataframe. It cannot join the values of columns in both the dataframes. For that **merge( ) function** is used, which we shall talk about in the coming section.

When using the **on** argument in **join( )**, you must provide **lsuffix** and **rsuffix** values if the two dataframes contain a column with same name :

```
<DF1>.join(<DF2>, on =<column_name>, lsuffix = <lsuffix string>, rsuffix= <rsuffix string>)
```

so that it can create unique names for columns by adding suffixes , otherwise it will return an error. If the dataframe Df1's P_id is renamed as ***Cust_id*** and you issue the following command

```
>>> Df1.join(Df2, on = 'Cust_id')
:
ValueError: columns overlap but no suffix specified: Index(['Cust_id'], dtype='object')
```

Both dataframes have the column namely Cust_id hence the error

In such a situation, if you specify suffixes with the **on** argument as

```
>>>Df1.join(Df2, on = 'Cust_id', lsuffix = '_CDf', rsuffix = '_SDf')
```

Then in the joined dataframe, the left dataframe's column will be represented as Cust_id_CDf (because **lsuffix** is *_Cdf*) and, the right dataframe's column will be represented as Cust_id_Sdf (because **rsuffix** is *_Sdf*). You can try it yourself.

NOTE

With **on** argument in **join( )**, the specified column's values from the left dataframe are joined with the indexes of the right dataframe.

2.8.3 Combining Dataframes using merge( )

Sometimes, you would want to combine two dataframes such that two rows with some common column value are merged together in the final result. For this, Pandas provides **merge( )** function in which you can specify the field on the basis of which you want to combine the two dataframes. It is used as per syntax :

To see merge( ) in action

Scan QR Code

```
pd.merge(<DF1>, <DF2>)
pd.merge(<DF1>, <DF2>, on = <field_name>)
pd.merge(<DF1>, <DF2>, [on = <field_name>],<how = 'left'|'right'|'inner'|'outer'>)
```


Where,

- The **on** argument takes the column name found in both the dataframes.
- The **how** argument specifies the type of merging which is actually a type of join (you have read earlier about joins). Default is **'inner'** join for merge( ).

If you skip the argument **on = <field_name>**, then it will take any merge on common fields of the two dataframes, but you can explicitly specify the field on the basis of which you want to merge the two dataframes.

Consider the following dataframes, which have a common column name, **Cust_id :**

```
>>> Df1
   Cust_id   Product
0    1       Oven
1    2       AC
2    3       AC
3    4       Speaker
4    5       Tablet
5    6       Smartphone
```

```
>>> Df2
   Cust_id   State
0    2       Delhi
1    4       Goa
2    6       Kerala
```

Now if you give simple **merge( )** function by just giving dataframes' names, it will produce the result as follows :

```
>>> pd.merge(Df1, Df2)
   Cust_id   Product     State
0    2       AC          Delhi
1    4       Speaker     Goa
2    6       Smartphone  Kerala
```

```
>>> pd.merge(Df1, Df2, on = 'Cust_id')
   Cust_id   Product     State
0    2       AC          Delhi
1    4       Speaker     Goa
2    6       Smartphone  Kerala
```

The result with on argument is just the same as by default merging takes place on common column and with on argument also, we specify the common column name.

You can also specify the merge-type which is actually a join type (discussed earlier in joins) by specifying the **how** argument, *e.g.*,

```
>>> pd.merge(Df1, Df2, on = 'Cust_id', how = 'left')
   Cust_id   Product     State
0    1       Oven        NaN
1    2       AC          Delhi
2    3       AC          NaN
3    4       Speaker     Goa
4    5       Tablet      NaN
5    6       Smartphone  Kerala
```

Like joins, in merge also, missing values for non-matching column values are represented through NaN values

Following three examples are based on the following *two* dataframes as shown below :

```
>>> cust
   cust_id first_name last_name   email
0        1     Danyal      Shah   dshah@kbc.com
1        2      Nisha      Jain   jainn@uru.com
2        3     Thomas    Madison  madisont@imp.com
3        4     Shubhi     Pillai  pillais@abc.com
4        5    Ishpreet     Singh   ips@xya.biz
```

```
>>> order
   order_id  amount  cust_id
0         1  23234.56       1
1         2  62378.50       3
2         3  32124.00       2
3         4   8365.50       3
4         5  12325.50      10
5         6  12614.40       9
```

EXAMPLE **48** *Write a program to create a merged dataframe using the* **cust** *and* **order** *dataframes (given before example 48), which is merged on common field* **cust_id** *and contains only the rows having matching cust_id values.*

SOLUTION

```
import pandas as pd
:          # dataframes cust and order created or loaded
res0 = pd.merge(order, cust, on = 'cust_id')
print(res0)
```

The output produced by above program is :

Output

```
   order_id  amount  cust_id first_name last_name  email
0         1  23234.56       1     Danyal     Shah  dshah@kbc.com
1         2  62378.50       3     Thomas   Madison  madisont@imp.com
2         4   8365.50       3     Thomas   Madison  madisont@imp.com
3         3  32124.00       2      Nisha      Jain  jainn@uru.com
```

EXAMPLE **49** *Write a program to create a merged dataframe using the* **cust** *and* **order** *dataframes (given before example 48), which is merged on the common field* **cust_id** *and contains all the rows from* **order** *dataframe and from* **cust** *dataframe, only the rows having matching cust_id.*

SOLUTION

```
import pandas as pd
:        # dataframes cust and order created or loaded
res = pd.merge(order, cust, on = 'cust_id', how = 'left')
print(res)
```

Output

```
   order_id  amount  cust_id first_name last_name  email
0         1  23234.56       1     Danyal     Shah  dshah@kbc.com
1         2  62378.50       3     Thomas   Madison  madisont@imp.com
2         3  32124.00       2      Nisha      Jain  jainn@uru.com
3         4   8365.50       3     Thomas   Madison  madisont@imp.com
4         5  12325.50      10        NaN        NaN  NaN
5         6  12614.40       9        NaN        NaN  NaN
```


EXAMPLE **50** *Write a program to create a merged dataframe using the* **cust** *and* **order** *dataframes (given before example 48), which is merged on common field* **cust_id** *and contains all the rows both from* **order** *and* **cust** *dataframes.*

SOLUTION

```
import pandas as pd
:          # dataframes cust and order created or loaded
res2 = pd.merge(order, cust, on = 'cust_id', how = 'outer')
print(res2)
```

Output

```
   order_id  amount  cust_id first_name last_name  email
0       1.0  23234.56       1     Danyal     Shah  dshah@kbc.com
1       2.0  62378.50       3     Thomas   Madison  madisont@imp.com
3       4.0   8365.50       3     Thomas   Madison  madisont@imp.com
2       3.0  32124.00       2      Nisha      Jain  jainn@uru.com
4       5.0  12325.50      10        NaN        NaN  NaN
5       6.0  12614.40       9        NaN        NaN  NaN
6       NaN       NaN       4     Shubhi     Pillai  pillais@abc.com
7       NaN       NaN       5    Ishpreet     Singh  ips@xya.biz
```

2.9 Function groupby( )

Within a dataframe, based on a field's values, you may want to group the data. Consider following dataframe **df1** storing yearly data of online tutoring company :

```
>>> df1
    Classes  Country  Quarter    Tutor
0        28      USA        1   Tahira
1        36       UK        1  Gurjyot
2        41    Japan        1   Anusha
3        32      USA        1    Jacob
4        40   Brazil        1   Venkat
5        36      USA        2   Tahira
6        40      USA        2  Gurjyot
7        36    Japan        2   Anusha
8        40   Brazil        2    Jacob
9        46      USA        2   Venkat
10       24   Brazil        3   Tahira
11       30      USA        3  Gurjyot
12       44       UK        3   Anusha
13       40   Brazil        3    Jacob
14       32      USA        3   Venkat
15       36    Japan        4   Tahira
16       32    Japan        4  Gurjyot
17       36   Brazil        4   Anusha
18       42       UK        4    Jacob
19       38      USA        4   Venkat
```

Here, you may want to create groups :

- Quarter wise
- Tutor wise
- Country wise
- and so on

In other words, the duplicate values in the same field are grouped together to form groups, *e.g.*, for creating *Tutor* wise groups :

- All the rows having **Tutor** as Tahira will be clubbed to form *Tahira group*
- All the rows having **Tutor** as Anusha will be clubbed to form *Anusha group*
- All the rows having **Tutor** as Gurjyot will be clubbed to form *Gurjyot group*

and so on.

Similarly, other types of groups get created.

To create such groups, Pandas provide **groupby( )** function. The syntax of minimalistic form of **groupby( )** is :

```
<dataframe>.groupby(by=None, axis=0)
```

by	label, or list of labels to be used for grouping
axis	{0 or 'index', 1 or 'columns'}, default 0 ; Split along rows (0) or columns (1).

The **groupby( )** creates the groups internally and does not display the grouped data by default, *e.g.,*

```
>>> df1.groupby('Tutor')
<pandas.core.groupby.DataFrameGroupBy object at 0x085A4490>
```

Python created groups based on Tutor column's values but did not display grouped data

The result of **groupby( )** is also an object, the *DataFrameGroupBy object*.

You can store the GroupBy object in a variable name and then use following attributes and functions to get information about groups or to display groups :

`<GroupByObject>.groups`	lists the groups created
`<GroupByObject>.get_group(<value>)`	lists the group created for the passed value
`<GroupByObject>.size()`	lists the size of the groups created
`<GroupByObject>.count()`	lists the count of non-NA values for each column in the groups created
`<GroupByObject>.[<columnname>].head()`	lists the specified column from the grouped object created
`<GroupByObject>.[<columnname>].sum()`	lists the sum of values of the specified column from the grouped object created
`<GroupByObject>.[<columnname>].mean()`	lists the average of values of the specified column from the grouped object created

Following example will make it clear as they are self-explanatory.

```
>>> gdf = df1.groupby('Tutor')
>>> gdf.groups
{'Anusha': Int64Index([2, 7, 12, 17], dtype='int64'),
 'Gurjyot': Int64Index([1, 6, 11, 16], dtype='int64'),
 'Jacob': Int64Index([3, 8, 13, 18], dtype='int64'),
 'Tahira': Int64Index([0, 5, 10, 15], dtype='int64'),
 'Venkat': Int64Index([4, 9, 14, 19], dtype='int64')}
>>> gdf.get_group('Venkat')
   Classes Country  Quarter  Tutor
4       40  Brazil        1  Venkat
9       46     USA        2  Venkat
14      32     USA        3  Venkat
19      38     USA        4  Venkat
>>> gdf.get_group('Gurjyot')
   Classes Country  Quarter  Tutor
1       36      UK        1  Gurjyot
6       40     USA        2  Gurjyot
11      30     USA        3  Gurjyot
16      32   Japan        4  Gurjyot
```

```
>>> gdf.size()
Tutor
Anusha     4
Gurjyot    4
Jacob      4
Tahira     4
Venkat     4
dtype: int64
>>> gdf.count()
         Classes  Country  Quarter
Tutor
Anusha         4        4        4
Gurjyot        4        4        4
Jacob          4        4        4
Tahira         4        4        4
Venkat         4        4        4
```

```
>>> gdf2['Classes'].head()
0    28
1    36
2    41
3    32
4    40
5    36
6    40
7    36
8    40
9    46
10   24
11   30
12   44
13   40
14   32
15   36
16   32
17   36
18   42
19   38
Name: Classes, dtype: int64
```

First of all, we created the groupby object based on field 'Tutor' and stored it in object namely **gdf**.

All other attributes and functions are then applied to this object **gdf**

EXAMPLE **51** *Create a Data Frame QtrSales where each row contains the item category, item name, and expenditure. Group the rows by the category, and print the total expenditure per category.*

SOLUTION

```
import pandas as pd
QtrSales = pd.DataFrame({ 'Item Category':['A','B','A','A','B','C', 'B', 'C'],
   'Item Name':['iPad', 'LCD', 'iPhone', 'iWatch', 'Projector', 'Hard disk', 'Smartboard', 'Pen drive'],
   'Expenditure':[288000, 356000, 497000, 315000, 413000, 45000, 211000, 21000] })
print(QtrSales)
print("Total expenditure category wise :")
print(QtrSales.groupby("Item Category")["Expenditure"].sum())
```

Output

```
  Item Category    Item Name   Expenditure
0             A         iPad        288000
1             B          LCD        356000
2             A       iPhone        497000
3             A       iWatch        315000
4             B    Projector        413000
5             C    Hard disk         45000
6             B   Smartboard        211000
7             C     Pen drive         21000
Total expenditure category wise :
Item Category
A             1100000
B              980000
C               66000
Name: Expenditure, dtype: int64
```

EXAMPLE **52** *Consider the same dataframe QtrSales of the previous example and Group the rows by the category, and print the average expenditure per category.*

SOLUTION

```
import pandas as pd
:       # dataframe QtrSales created or loaded
print("Average expenditure category wise :")
print(QtrSales.groupby("Item Category")["Expenditure"].mean())
```

Output

```
Average expenditure category wise :
Item Category
A             366666.666667
B             326666.666667
C              33000.000000
Name: Expenditure, dtype: float64
```

Check Point 2.2

1. What does join( ) do ?
2. How are concat( ) and merge( ) similar or different ?
3. What is the difference between dropna( ) and fillna( ) ?

Please note that each function covered in the chapter has much more functionality. Covering each and every argument and all the functionality of each function is not possible as this book's aim is not to be a manual rather explain concepts as per given syllabus.

With this we have come to the end of this chapter. Let us quickly revise what we have learnt in this chapter.

LET US REVISE

- *You can iterate over a dataframe row-wise using **iterrows( )** and column-wise using **iteritems( )**.*
- *You can perform binary arithmetic operations on dataframes using **add( ), radd( ), sub( ), rsub( ), mul( ), div( )** and **rdiv( )** functions.*
- *Python integer types cannot store NaN values. To store a NaN value in a column, the datatype of a column is changed to non-integer suitable type.*
- *Common dataframe attributes include: **index, columns, axes, dtype, size, shape, values, empty, ndim, T.***
- *Transpose of a dataframe is available through <DF>.T*
- *When you assign something to a column of dataframe, then for existing column, it will change the data values and for non-existing column, it will add a new column.*
- *Quantiles are points in a distribution of data that relate to a share of values in that distribution. The quantile of a value is the fraction of observations less than or equal to the value.*
- *The quantile of the median is 0.5, by definition.*
- *You can apply functions to whole dataframe or subsets of dataframes.*
- *Data pivoting is a summarising technique to rearrange the columns and rows in a report so as to view data from different perspectives.*
- *Pandas library makes available two functions for pivoting – the **pivot( )** and **pivot_table( )** functions.*
- *With pivot( ), if there are multiple entries for a column's value for the same values for index(row), it leads to error.*
- *The pivot_table( ) pivots the data by aggregating it, thus it can work with duplicate entries. Commonly used aggregate functions are **count( ), sum( ), mean( ), median( ), min( ), max( ), std( ), var( ), mad( )**.*
- *A histogram is a plot that lets you discover, and show, the underlying frequency distribution (shape) of a set of continuous data.*
- *Pandas' **hist( )** function lets you create histograms.*
- *Function **isnull( )** checking if a DataFrame has missing values.*
- *Function **fillna( )** fills the missing values.*
- *To join two dataframes, one can use functions – concat( ), join( ) and merge( ).*

Objective Type Questions — OTQs

Multiple Choice Questions

1. To iterate over horizontal subsets of dataframe, _____ function may be used.
   (a) iterate( ) (b) iterrows( ) (c) itercols( ) (d) iteritems( )
2. To iterate over vertical subsets of a dataframe, _____ function may be used.
   (a) iterate( ) (b) iterrows( ) (c) itercols( ) (d) iteritems( )

3. To add two dataframes' values, _____ function may be used.
   (a) plus (b) rplus (c) add (d) radd
4. To subtract the values of two dataframes, _____ function may be used.
   (a) sub (b) difference (c) minus (d) rsub
5. To divide the values of two dataframes, _____ function may be used.
   (a) divide (b) div (c) rdiv (d) division
6. To skip NaN values in a calculation, you can specify _____ attribute.
   (a) NaN (b) NA (c) skipna (d) all of these
7. Which of the following is not a valid function that can be used with dataframes ?
   (a) count( ) (b) sum( ) (c) length( ) (d) mad( )
8. A technique, which when performed on a dataframe, rearranges the data from rows and columns in a report form, is called _____ .
   (a) summarising (b) reporting (c) grouping (d) pivoting
9. The technique that divides total distribution of data into a given number of equal proportions is called a _____ .
   (a) quartile (b) tercile (c) median (d) quantile
10. _____ divides the total distribution in four equal parts.
    (a) quartile (b) tercile (c) median (d) quantile
11. To divide total distribution of given data in *two* equal parts, _____ function is used.
    (a) median( ) (b) quartile( ) (c) quantile( ) (d) all of these
12. To divide total distribution of given data in *four* equal parts, _____ function is used.
    (a) median( ) (b) quartile( ) (c) quantile( ) (d) all of these
13. To divide total distribution of given data in *eight* equal parts, _____ function is used.
    (a) median( ) (b) quartile( ) (c) quantile( ) (d) all of these
14. The technique to summarise given data by transferring rows to columns is called _____ .
    (a) transfer (b) transpose
    (c) pivoting (d) swapping
15. The function to perform pivoting with dataframes having unique values is _____ .
    (a) pivot(unique = True) (b) pivot( )
    (c) pivot_table(unique = True) (d) pivot_table( )
16. The function to perform pivoting with dataframes having duplicate values is _____ .
    (a) pivot(unique = True) (b) pivot( )
    (c) pivot_table(unique = True) (d) pivot_table( )
17. The function to create histograms for all numeric columns of a dataframe is _____ .
    (a) histogram( ) (b) hist(numeric_only = True)
    (c) hist( ) (d) all of these
18. Which of the following is not a descriptive statistics function ?
    (a) count( ) (b) add( ) (c) sum( ) (d) max( )
19. Which function calculates descriptive statistical details for a dataframe ?
    (a) info( ) (b) describe( ) (c) show( ) (d) list( )

20. To calculative cumulative sum of a column of a dataframe, you may use _____ function.
(a) sun( ) (b) sum(cumulative = True)
(c) cumsum( ) (d) none of these

21. The function to get the index of maximum value in a column of dataframe is _____ .
(a) max( ) (b) index( ) (c) idxmax( ) (d) maxidx( )

22. To get top 5 rows of a dataframe, you may use _____ function.
(a) head( ) (b) head(5) (c) top( ) (d) top(5)

23. To get bottom 3 rows of a dataframe, you may use _____ function.
(a) tail( ) (b) tail(3) (c) bottom( ) (d) bottom(3)

24. Function _____ can be used to drop missing values.
(a) fillna( ) (b) isnull( ) (c) dropna( ) (d) delna( )

25. Which of the following methods of combining two dataframes is a patching method ?
(a) concat( ) (b) merge( ) (c) join( ) (d) none of these

Fill in the Blanks

1. _____ and _____ functions help you to iterate over a dataframe.
2. To add two dataframes, you may use functions _____ or _____ .
3. To concatenate two string columns of a dataframe, _____ operator is used.
4. The _____ function returns the maximum repeating value.
5. The _____ function returns the average of given data.
6. The _____ function returns the halfway point in a given data.
7. Using _____ function you can calculate terciles.
8. Using _____ function you can calculate quartiles.
9. Using _____ function you can calculate octiles.
10. Using _____ method, you can check if there are any missing values in dataframe.
11. To concatenate two dataframes, you can use _____ method.
12. The _____ function combines two dataframes such that two rows with some common value are merged together in the final result.
13. _____ function sorts on the basis of the values of a dataframe.
14. _____ function sorts on the basis of index of a dataframe.

True/False Questions

1. The iteritems( ) iterates over the rows of a dataframe.
2. The iteritems( ) iterates over the columns of a dataframe.
3. The result produced by the functions sub( ) and rsub( ) is the same.
4. The result produced by the functions add( ) and radd( ) is the same.
5. The result produced by the functions div( ) and rdiv( ) is the same.
6. The info( ) and describe( ) list the same information about a dataframe.

7. Function **add( )** and operator **+** give the same result.
8. Function **rsub( )** and operator **–** give the same result.
9. The minus – operator's result is same as sub( ) and rsub( ).
10. Python integer datatype can store NaN values.
11. Functions sum( ) and cumsum( ) produce the same result.
12. Functions **pivot( )** and **pivot_table( )** are identical functions.
13. Function pivot( ) can work with unique data only.
14. Function pivot_table( ) can work with unique data.
15. Function pivot_table( ) can work with duplicate data.
16. By default, hist( ) will create histogram for all numeric columns of a dataframe.
17. The fillna( ) can also fill individual missing values for difficult columns.
18. To drop missing values from a dataframe, the function used is delna( ).
19. The merge( ) and concat( ) work identically.
20. With merge( ), you can specify the merging field.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *Name the functions you can use to iterate over dataframes.*
Solution. *iterrows( )* and *iteritems( )*

2. *What is the basic difference between iterrows() and iteritems() ?*
Solution.
<DF>.iteritems( ) iterates over vertical subsets in the form of (*col-index, Series*) pairs, and
<DF>.iterrows( ) iterates over horizontal subsets in the form of (*row-index, Series*) pairs.

3. *In Binary addition, if a column in a data frame contains a NaN and the corresponding column in other DataFrame is a numeric value, then what would be returned as a result? Why?*
Solution. The result of addition of a NaN value and a number is a NaN always. This is because, NaN means not-a-number and there is functionality defined only for adding two numbers and not with a NaN.

4. *Given two dataframe df3 and df4 as shown below :*

```
In [127]: df3
Out[127]:
     A    B    C
0  100  200  300
1  400  500  600
```

```
In [128]: df4
Out[128]:
     A     B
0  1000  2000
1  4000  5000
2  7000  8000
```

```
In [129]: df3 + df4
Out[129]:
        A       B   C
0  1100.0  2200.0 NaN
1  4400.0  5500.0 NaN
2     NaN     NaN NaN
```

Both these dataframes store integer values but when they are added as df3 + df4, the values in the resultant object automatically change to floating point (as shown on above right) contrary to the fact the two integers when added will result into integer only. Can you specify the reason ?

Solution. The reason behind the conversion to floating point type is that the two dataframes have different indexes and columns. For the non-matching row indexes and columns, Python will add NaN values to corresponding value from another dataframe.

Python stores NaN values in a non-integer suitable data type. Thus, the moment NaN is added or present in any column, the datatype of the entire column is changed. Thus, all the values are represented as floating point value because of the presence of NaN values in their column.

5. *Given two Dataframes One and Two as shown here :*

```
>>> One
   name  value
0    p    1.0
1    q    2.0
2    r    NaN
```

```
>>> Two
   name  value
0    p    1.0
1    q    NaN
2    r    3.0
3    s    4.0
```

What will be the result of the following ?

(*a*) One.radd(Two) (*b*) One + Two

(*c*) One.rsub(Two)

Solution.

For both (*a*) and (*b*) the output will be like :

```
>>> One + Two
   name  value
0   pp    2.0
1   qq    NaN
2   rr    NaN
3  NaN    NaN
```

For (*c*), Python will raise an error as string values cannot be subtracted.

6. *Write equivalent function for the following operations on two DataFrames A and B*

(*i*) A+B (*ii*) B+A (*iii*) A-B (*iv*) B-A (*v*) B*A (*vi*) B/A (*v*) A/B

Solution.

(*i*) A.add(B) (*ii*) B.add(A) (*iii*) A.sub(B)

(*iv*) B.sub(A) or A.rsub(B) (*v*) B.mul(A)

(*vi*) A.rdiv(B) or B.div(A) (*vii*) B.rdiv(A) or A.div(B)

7. *Given a dataframe namely **wdf** as shown below :*

```
    minTemp  maxTemp  Rainfall  Evaporation
0         1      8.0      24.3          0.0
1         2     14.0      26.9          3.6
2         3     13.7      23.4          3.6
3         4     13.3      15.5         39.8
4         5      7.6      16.1          2.8
5         6      6.2      16.9          0.0
6         7      6.1      18.2          0.2
7         8      8.3      17.0          0.0
8         9      8.8      19.5          0.0
9        10      8.4      22.8         16.2
10       11      9.1      25.2          0.0
11       12      8.5      27.3          0.2
12       13     10.1      27.9          0.0
13       14     12.1      30.9          0.0
14       15     10.1      31.2          0.0
15       16     12.4      32.1          0.0
16       17     13.8      31.2          0.0
17       18     11.7      30.0          1.2
18       19     12.4      32.3          0.6
19       20     15.6      33.4          0.0
20       21     15.3      33.4          0.0
```

(*i*) *Write command to compute sum of every column of the dataframe.*

(*ii*) *Write command to compute mean of column **Rainfall**.*

(*iii*) *Write command to compute sum of every row of the dataframe.*

(*iv*) *Write command to compute average of all the columns for last 10 rows only.*

(*v*) *Write command to compute average **maxTemp, Rainfall** for first 10 rows.*

Solution.

(*i*) `wdf.sum()`

(*ii*) `wdf['Rainfall'].mean ()`

(*iii*) `wdf.sum(axis = 1)`

(*iv*) `wdf.loc[11:, ].mean()`

(*v*) `wdf.loc[:11,'maxtemp':'Rainfall'].mean()`

8. *What is pivoting ? Which functions of Pandas support pivoting ?*

Solution. Data pivoting is a summarising technique to rearrange the columns and rows in a report so as to view data from different perspectives.

Pandas library makes available two functions for pivoting – the pivot( ) and pivot_table( ) functions.

9. *What is a quartile ? How is it related to quantile ? How do you generate it in Pandas ?*

Solution. ***Quartiles*** Q1, Q2 and Q3 are *three* points that divide a distribution into 4 equal parts containing 25% percentile each of the entire distribution. The 4-quantiles are called ***quartiles***. A quantile refers to an equal share in an equally divided distribution *e.g.*, median quantile divides a distribution into 2 equal parts and each equal share is 50% quantile.

Quartile on the other hand refers to when a distribution is divided into *four* quantiles each containing 25% percentile.

In Pandas, we generate these with function **quantile()**.

10. *What is the difference between pivot() and pivot_table() functions?*

Solution. The pivot( ) and pivot_table( ) both perform data pivoting a data set. But with pivot( ), if there are multiple entries for a column's value for the same values for index(row), it leads to an error.

The pivot_table( ) pivots the data by aggregating it, thus it can work with duplicate entries.

11. *A dataframe **fdf** stores data about passengers, flights and years. First few rows of the dataframe are shown below.*

	Year	Month	Passengers
0	2009	January	112
1	2009	February	118
2	2009	March	132
3	2009	April	129
4	2009	May	121

Using above dataframe, write commands for the following :

(*a*) *Compute total passengers per year.* (*b*) *Compute average passengers per month.*

Solution.

(*a*) `fdf.pivot_table(index = 'year', values = 'passengers', aggfunc = 'sum')`

(*b*) `fdf.pivot_table(index = 'month', values = 'passengers', aggfunc = 'mean')`

12. *What is descriptive statistics ? Name the functions commonly used for calculating this.*

Solution. A descriptive statistic is a summary statistic that quantitatively describes or summarizes features of a collection of information.

Commonly used functions for descriptive statistics are : *count(), sum(), mean(), max(), min(), std(), quartiles* etc.

13. *What is missing data ? Why is it considered a problem ?*

Solution. Missing Data means when no information is provided for one or more items or for a whole unit. Missing Data can also refer to as NA (Not Available) values in Pandas. Pandas puts *NaN* in place of missing data in dataframes.

Missing Data is a very big problem in real life scenario. This is because, the presence of *NaN* hampers calculations because Nan cannot be used in calculations and in fact, it makes the whole calculation result as NaN.

14. *Write command to print cumulative sum of columns Rainfall and Evaporation in the dataframe wdf used above.*

Solution. `wdf[['Rainfall', 'Evaporation']].apply(np.cumsum)`

```
In [134]: wdf[['Rainfall', 'Evaporation']].apply(np.cumsum)
Out[134]:
    Rainfall  Evaporation
0       24.3          0.0
1       51.2          3.6
2       74.6          7.2
3       98.1         47.0
4      106.2         49.8
5      123.1         49.8
6      141.3         50.0
7      158.3         50.0
8      177.8         50.0
9      200.6         66.2
10     225.8         66.2
11     253.1         66.4
12     281.0         66.4
13     311.9         66.4
14     343.1         66.4
15     375.2         66.4
16     406.4         66.4
17     436.4         67.6
18     468.7         68.2
19     502.1         68.2
20     535.5         68.2
```

15. *Is there any one function that performs descriptive statistics on a dataframe ?*

Solution. Yes, Pandas provides **describe( )** function that calculates most descriptive statistics information for a DataFrame along with 25%, 50% and 75% percentile values (quartiles), *e.g.,*

```
>>> df.describe()
            Age        Projects      Budget
count  6.000000     6.000000    6.000000
mean   31.500000   16.833333   23.500000
std     4.636809    3.188521   14.237275
min    27.000000   13.000000   10.000000
25%    28.500000   14.500000   14.000000
50%    31.000000   16.500000   19.000000
75%    32.000000   19.250000   29.250000
max    40.000000   21.000000   48.000000
```

16. *What functions does Pandas provide to handle missing data ?*

Solution. Most common functions provided by Pandas to handle missing data are :

`isnull(), dropna(), fillna()`

17. *Consider the same dataframes One and Two shown in previous solved problem. Write a statement so that the missing value of One is filled or combined from the corresponding value of dataframe Two. Also show the result after combining the values.*

Solution.

`One.combine_first(Two)`

The result after combining will be :

```
In [82]: One.combine_first(Two)
Out[82]:
  name  value
0    p    1.0
1    q    2.0
2    r    3.0
3    s    4.0
```


18. *Consider the dataframes One and Two given in solved problem 5. What will be the output of following statements :*

(a) `pd.concat( [One, Two] )`

(b) `pd.concat( [One, Two], axis = 1)`

(c) `pd.merge(One, Two, on = 'name' )`

Solution. (a)

	name	value
0	p	1.0
1	q	2.0
2	r	NaN
0	p	1.0
1	q	NaN
2	r	3.0
3	s	4.0

(b)

	name	value	name	value
0	p	1.0	p	1.0
1	q	2.0	q	NaN
2	r	NaN	r	3.0
3	NaN	NaN	s	4.0

(c)

	name	value_x	value_y
0	p	1.0	1.0
1	q	2.0	NaN
2	r	NaN	3.0

Practical Questions

19. *Write a program to iterate over a dataframe containing names and marks, which then calculates grades as per marks (as per guidelines below) and adds them to the grade column.*

Marks >= 90 grade A+ ; Marks 50-60 grade C ;
Marks 70-90 grade A ; Marks 40-50 grade D ;
Marks 60-70 grade B ; Marks <40 grade F

Solution.

```
import pandas as pd
import numpy as np
names = pd.Series(['Rohan', 'Misha', 'Mike', 'Simran'])
marks = pd.Series([76.0, 56.0, 91.0, 67.0])
Stud = { 'Name' : names, 'Marks' : marks }
df1 = pd.DataFrame(Stud, columns = ['Name', 'Marks'])
df1['Grade']= np.NaN          <-- This will add NaN values to complete column Grade
print ("Initial values in dataframe")
print(df1)

for (col, colSeries) in df1.iteritems() :
    length = len(colSeries)                  # number of entries in colSeries
    if col == 'Marks':
        lstMrks = [ ]                        # initialize empty list
        for row in range(length) :
            mrks = colSeries[row]
            if mrks >= 90 :
                lstMrks.append('A+')         #grade appended to list lstMrks
            elif mrks >= 70 :
                lstMrks.append('A')          #grade appended to list lstMrks
            elif mrks >= 60 :
                lstMrks.append('B')          #grade appended to list lstMrks
```

```
    elif mrks >= 50 :
        lstMrks.append('C')          #grade appended to list lstMrks
    elif mrks >= 40 :
        lstMrks.append('D')          #grade appended to list lstMrks
    else :
        lstMrks.append('F') )        #grade appended to list lstMrks
df1['Grade'] = lstMrks
print("\n Dataframe after calculating grades")
print(df1 )
```

This will put values of list lstMrks that contains computed grades to the column Grade

The output produced by the above program is as shown below :

```
Initial values in dataframe
      Name   Marks  Grade
0    Rohan    76.0    NaN
1    Misha    56.0    NaN
2    Mike     91.0    NaN
3    Simran   67.0    NaN

Dataframe after calculating grades
      Name   Marks  Grade
0    Rohan    76.0    A
1    Misha    56.0    C
2    Mike     91.0    A+
3    Simran   67.0    B
```

20. *Given a DataFrame df*

	Name	Sex	Position	City	Age	Projects	Budget
0	Rabia	F	Manager	Bangalore	30	13	48
1	Evan	M	Programmer	New Delhi	27	17	13
2	Jia	F	Manager	Chennai	32	16	32
3	Lalit	M	Manager	Mumbai	40	20	21
4	Jaspreet	M	Programmer	Chennai	28	21	17
5	Suji	F	Programmer	Bangalore	32	14	10

Write a program to print only the Name, Age and Position for all rows.

Solution.

```
import pandas as pd
import numpy as np
:       # given df created or loaded
for i, row in df.iterrows():
   print(row['Name'], '\t', row['Age'], '\t', row['Position'])
```

Output

```
Rabia    30    Manager
Evan     27    Programmer
Jia      32    Manager
Lalit    40    Manager
Jaspreet 28    Programmer
Suji     32    Programmer
```


21. *Consider the crop statistics DataFrame prodf used in the chapter. Write a program to print the minimum crop production for each crop.*

Solution.

```
import pandas as pd
import numpy as np
:       # prodf created or loaded
print("Minimum crop production")
print(prodf.min() )
```

Output

```
Minimum crop production
Fruits     44.1
Pulses      1.7
Rice      814.6
Wheat       0.5
dtype: float64
```

22. *Consider the crop statistics DataFrame prodf used in the chapter. Write a program to print the average crop production for each crop.*

Solution.

```
import pandas as pd
import numpy as np
:       # prodf created or loaded
print("Average crop production")
print(prodf.mean() )
```

Output

```
Average crop production
Fruits    27876.400000
Pulses      665.216667
Rice       6357.000000
Wheat     12308.500000
dtype: float64
```

23. *Consider the DataFrame ndf used in the chapter :*

	Name	Sex	Position	City	Age	Projects	Budget
0	Rabia	F	Manager	Bangalore	30	13	48
1	Evan	M	Programmer	New Delhi	27	17	13
2	Jia	F	Manager	Chennai	32	16	32
3	Lalit	M	Manager	Mumbai	40	20	21
4	Jaspreet	M	Programmer	Chennai	28	21	17
5	Suji	F	Programmer	Bangalore	32	14	10

Write a program to summarise how much budget is allocated to different Positions, for each City ?

Solution. Now our program will look like :

```
import pandas as pd
import numpy as np
:       # given ndf created or loaded
print(ndf.pivot(index = 'Position', columns = 'City', values = 'Budget'))
```

Output

```
City        Bangalore  Chennai  Mumbai  New Delhi
Position
Manager          48.0     32.0    21.0        NaN
Programmer       10.0     17.0     NaN       13.0
```

24. *Consider the original DataFrame* **mk** *of previous question. Write a program to calculate average marks for each section.*

Solution.

```
import pandas as pd
import numpy as np
:          # given mk created or loaded
print("Average marks section wise")
print( mk.mean())
```

Output

```
Average marks section wise
A    94.00
B    94.25
C    95.25
D    94.50
dtype: float64
```

25. *Given a dataframe Sales that store sales records of a company of past few years. But the dataframe contains some missing data.*

	Item Type	Sales Channel	Order Date	Order ID	Total Revenue	Total Cost	Total Profit
0	Cosmetics	Online	5/22/2017	8985.0	793518.00	477943.95	315574.05
1	Cereal	Online	5/20/2017	5559.0	1780539.20	1013704.16	766835.04
2	Personal Care	Offline	NaN	NaN	NaN	NaN	NaN
3	Personal Care	Online	3/11/2017	6992.0	246415.95	170860.05	75555.90
4	Snacks	Online	2/25/2017	7562.0	1117953.66	713942.88	404010.78
5	Household	Offline	2/8/2017	52284.0	5997054.98	4509793.96	1487261.02
6	Meat	Online	NaN	NaN	NaN	NaN	NaN
7	Clothes	Online	1/13/2017	1873.0	902980.64	296145.92	606834.72
8	Cosmetics	Online	12/31/2016	331438481.0	3876652.40	2334947.11	1541705.29
9	Office Supplies	Offline	NaN	NaN	NaN	NaN	NaN
10	Cosmetics	Offline	11/19/2016	419123971.0	3039414.40	1830670.16	1208744.24
11	Cosmetics	Online	11/15/2016	286959302.0	2836990.80	1708748.37	1128242.43
12	Beverages	Offline	NaN	NaN	NaN	NaN	NaN
13	Clothes	Offline	7/25/2016	807025039.0	600821.44	197048.32	403773.12
14	Snacks	Online	6/30/2016	795490682.0	339490.50	216804.00	122686.50

Write a script that does the following :

(i) Lists the presence of missing data in whole dataframe.

(ii) Fills the missing values with 999.

(iii) Print the dataframe after filling missing value.

Solution.

```
import pandas as pd
import numpy as np

:          # creation or loading of dataframe Sales

# presence of missing data element wise
print("Missing data element wise")
print( ( Sales.isnull() ) )
Sales = Sales.fillna(999)
print("After filling missing values with 999, the dataframe is like:")
print(Sales)
```


The output produced by above script is like :

	Item Type	Sales Channel	Order Date	Order ID	Total Revenue	Total Cost	Total Profit
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	True	True	True	True	True
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False
6	False	False	True	True	True	True	True
7	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False
9	False	False	True	True	True	True	True
10	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False
12	False	False	True	True	True	True	True
13	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False

After filling missing values with 999, the dataframe is like:

	Item Type	Sales Channel	Order Date	Order ID	Total Revenue	Total Cost	Total Profit
0	Cosmetics	Online	5/22/2017	8985.0	793518.00	477943.95	315574.05
1	Cereal	Online	5/20/2017	5559.0	1780539.20	1013704.16	766835.04
2	Personal Care	Offline	999	999.0	999.00	999.00	999.00
3	Personal Care	Online	3/11/2017	6992.0	246415.95	170860.05	75555.90
4	Snacks	Online	2/25/2017	7562.0	1117953.66	713942.88	404010.78
5	Household	Offline	2/8/2017	52284.0	5997054.98	4509793.96	1487261.02
6	Meat	Online	999	999.0	999.00	999.00	999.00
7	Clothes	Online	1/13/2017	1873.0	902980.64	296145.92	606834.72
8	Cosmetics	Online	12/31/2016	331438481.0	3876652.40	2334947.11	1541705.29
9	Office Supplies	Offline	999	999.0	999.00	999.00	999.00
10	Cosmetics	Offline	11/19/2016	419123971.0	3039414.40	1830670.16	1208744.24
11	Cosmetics	Online	11/15/2016	286959302.0	2836990.80	1708748.37	1128242.43
12	Beverages	Offline	999	999.0	999.00	999.00	999.00
13	Clothes	Offline	7/25/2016	807025039.0	600821.44	197048.32	403773.12
14	Snacks	Online	6/30/2016	795490682.0	339490.50	216804.00	122686.50

26. *Consider DataFrame* **mk** *as shown below :*

	A	B	C	D
Acct	NaN	94.0	92.0	97.0
Eco	90.0	94.0	NaN	97.0
Eng	95.0	89.0	91.0	89.0
IP	94.0	NaN	99.0	95.0
Math	97.0	100.0	99.0	NaN

Write a program to fill missing values section wise as given below and name the new DataFrame as **nmk** *:*

- For section 'A' fill 20,
- For section 'B' fill 10,
- For section 'C' fill 20,
- For section 'D' fill 0

Solution.

```
import pandas as pd
import numpy as np
:          # given mk created or loaded
nmk = mk.fillna( {'A':20, 'B':10, 'C':20, 'D':0})
print("Dataframe after filling missing values section wise")
print(nmk)
```

Output

```
Dataframe after filling missing values section wise
        A     B     C     D
Acct  20.0  94.0  92.0  97.0
Eco   90.0  94.0  20.0  97.0
Eng   95.0  89.0  91.0  89.0
IP    94.0  10.0  99.0  95.0
Math  97.0 100.0  99.0   0.0
```

27. *Consider the DataFrame* **nmk** *after filling the missing values section wise (refer to the solution of Q. 26), write a program to calculate average marks for each section after filling up of missing values.*

Solution.

```
import pandas as pd
import numpy as np
:        # given mk created or loaded
nmk = mk.fillna( {'A':20, 'B':10, 'C':20, 'D':0})
print("Average marks section wise after filling missing values")
print(nmk.mean())
print("Average marks section wise after filling missing values")
print(nmk.mean())
```

Output

```
Average marks section wise after filling missing values
A    79.2
B    77.4
C    80.2
D    75.6
dtype: float64
```

28. *Consider DataFrames* **pdf** *and* **pdf2** *as shown below :*

pdf

	Sean	Shadab	Sia	Simran
L1	295.0	600.0	505	NaN
L2	397.0	NaN	600	397.0
L3	NaN	800.0	700	500.0

pdf2

	Sean	Shadab	Sia	Simran
L1	295	600.0	400	200.0
L2	397	200.0	600	397.0
L3	400	NaN	500	NaN

Write a program to fill the missing values in **pdf** *with corresponding values.*

Solution.

```
import pandas as pd
import numpy as np
:        # given pdf, pdf2 created or loaded
print("Original pdf")
print(pdf)
print("Dataframe pdf after filling values from datafrme pdf2")
print(pdf.fillna(pdf2))
```


Output

```
Original pdf
      Sean  Shadab  Sia  Simran
L1  295.0   600.0  505     NaN
L2  397.0     NaN  600   397.0
L3    NaN   800.0  700   500.0
Dataframe pdf after filling values from datafrme pdf2
      Sean  Shadab  Sia  Simran
L1  295.0   600.0  505   200.0
L2  397.0   200.0  600   397.0
L3  400.0   800.0  700   500.0
```

29. *Consider the crop statistics DataFrame* **prodf**. *Write a program to calculate mean absolute deviation for all the crops.*

Solution.

```
import pandas as pd
import numpy as np
:        # given prodf created or loaded
print("Mean Absolute Deviation Crop wise")
print(prodf.mad())
```

```
Mean Absolute Deviation Crop wise
Fruits    37430.933333
Pulses      645.916667
Rice       4573.866667
Wheat     10939.750000
dtype: float64
```

30. *Consider the crop statistics DataFrame* **prodf**. *Write a program to calculate mean absolute deviation for all the states.*

Solution.

```
import pandas as pd
import numpy as np
:        # given prodf created or loaded
print("Mean Absolute Deviation State wise")
print(prodf.mad(axis = 1))
```

Output

```
Mean Absolute Deviation State wise
Andhra P.     2982.311111
Gujrat        3795.625000
Kerala        1132.177778
Punjab        5210.425000
Tripura        297.000000
Uttar P.     46814.150000
dtype: float64
```

31. *Consider the following dataframe, and answer the questions given below :* [CBSE Sample Paper 2019-20]

```
import pandas as pd
df = pd.DataFrame( { "Quarter1":[2000, 4000, 5000, 4400, 10000],
                     "Quarter2":[5800, 2500, 5400, 3000, 2900],
                     "Quarter3":[20000, 16000, 7000, 3600, 8200],
                     "Quarter4":[1400, 3700, 1700, 2000, 6000]})
```

(*i*) *Write the code to find mean value from above dataframe df over the index and column axis.*

(*ii*) *Use sum( ) function to find the sum of all the values over the index axis.*

(*iii*) *Find the median of the dataframe df.*

Solution.

(*i*)
```
print(df.mean(axis = 1))
print(df.mean(axis = 0))
```

(*ii*)
```
print(df.sum(axis = 1))
```

(*iii*)
```
print(df.median())
```

32. *Given a data frame df1 as shown below :* [CBSE Sample Paper 2019-20]

City	MaxTemp	MinTemp	RainFall
Delhi	40	32	24.1
Bengaluru	31	25	36.2
Chennai	35	27	40.8
Mumbai	29	21	35.2
Kolkata	39	23	41.8

(i) Write command to compute sum of every column of the data frame.

(ii) Write command to compute mean of column Rainfall.

(iii) Write command to compute Median of the MaxTemp Column.

Solution.

(i) `df1.sum( )` (ii) `df1['RainFall'].mean( )`

(iii) `df1.loc[:, 'MaxTemp'].median( )`

33. *Consider the two dataframes namely emp and dept given below :*

```
>>> emp
   emp_id  emp_name   hire_date  dept_id
0       1      Rina  2001-05-01      4.0
1       2      Mina  2002-07-15      1.0
2       3      Tina  2005-10-18      5.0
3       4      Dina  2007-01-03      3.0
4       5      Nina  2008-06-24      NaN
```

```
>>> dept
   dept_id        dept_name
0        1   Administration
1        2  Customer Service
2        3          Finance
3        4  Human Resources
4        5            Sales
```

Write a program to join these two dataframes on their indexes, but keep in mind that both dataframes have a common column name which , if not handled, leads to error.

Solution. To resolve common column name in **join( )**, we can use **lsuffix** and **rsuffix** arguments

```
import pandas as pd
:          # dataframes emp and dept created or loaded
ndf = emp.join(dept, lsuffix = '_emp', rsuffix= '_dept')
print("Joined dataframe is :")
print(ndf)
```

See, the column of left dataframe (emp here) is suffixed with lsuffix and the column of right dataframe (dept here) is suffixed with rsuffix

Output

```
Joined dataframe is :
   emp_id  emp_name   hire_date  dept_id_emp  dept_id_dept        dept_name
0       1      Rina  2001-05-01          4.0             1   Administration
1       2      Mina  2002-07-15          1.0             2  Customer Service
2       3      Tina  2005-10-18          5.0             3          Finance
3       4      Dina  2007-01-03          3.0             4  Human Resources
4       5      Nina  2008-06-24          NaN             5            Sales
```

34. *Consider the dataframes emp and dept of previous solved problem. Write a program to create a dataframe that stores the merged dataframes emp and dept , joined on the basis of dept_id and conatin all the rows of both the dataframes.*

Solution. Since outer join contains all the rows from both the dataframes, we shall merge using outer join.

```
import pandas as pd
:          # dataframes emp and dept created or loaded
mdf = pd.merge(emp, dept, on = 'dept_id', how = 'outer')
print("Merged dataframe is:")
print(mdf)
```

Output

```
Merged dataframe is:
   emp_id  emp_name   hire_date  dept_id        dept_name
0     1.0      Rina  2001-05-01      4.0  Human Resources
1     2.0      Mina  2002-07-15      1.0   Administration
2     3.0      Tina  2005-10-18      5.0            Sales
3     4.0      Dina  2007-01-03      3.0          Finance
4     5.0      Nina  2008-06-24      NaN              NaN
5     NaN       NaN         NaN      2.0  Customer Service
```

35. *Create a Data Frame **QtrSales** where each row contains the item category, item name, and expenditure. Locate the 3 largest values of Expenditure in this data frame.*

Solution.

```
import pandas as pd
QtrSales = pd.DataFrame({
   'Item Category':['A','B','A','A','B','C', 'B', 'C'],
   'Item Name':['iPad', 'LCD', 'iPhone', 'iWatch', 'Projector', 'Hard disk', 'Smartboard', 'Pen drive'],
   'Expenditure':[288000, 356000, 497000, 315000, 413000, 45000, 211000, 21000]
})
print("Dataframe QtrSales is :")
print(QtrSales)
print(" 3 largest expenditure values in given dataframe are :" )
print(QtrSales.sort_values("Expenditure", ascending = False).head(3) )
```

```
Dataframe QtrSales is :
Item     Category  Item Name    Expenditure
0              A       iPad         288000
1              B        LCD         356000
2              A     iPhone         497000
3              A     iWatch         315000
4              B  Projector         413000
5              C  Hard disk          45000
6              B  Smartboard         211000
7              C  Pen drive          21000
3 largest expenditure values in given dataframe are :
Item     Category  Item Name    Expenditure
2              A     iPhone         497000
4              B  Projector         413000
1              B        LCD         356000
```

GLOSSARY

DataFrame	Pandas data structure storing the two-dimensional labelled array.
Pivoting	Data summarising technique to rearrange the columns and rows in a report so as to view data from different perspectives.
Quantile	Points in a distribution of data that relate to a share of values in that distribution.

Assignment

For selective assignment solutions

Scan QR Code

Type A : Very Short Answer Questions

1. Name the function to iterate over a DataFrame horizontally.
2. Name the function to iterate over a DataFrame vertically.
3. Write equivalent expressions for the given functions :
   (*i*) A.add(B) (*ii*) B.add(A) (*iii*) A.sub(B) (*iv*) B.sub(A)
   (*v*) A.rsub(B) (*vi*) B.mul(A) (*vii*) A.rdiv(B) (*viii*) B.div(A)
   (*ix*) B.rdiv(A) (*x*) A.div(B)
4. Is the result of sub( ) and rsub( ) the same? Why/why not ?
5. Write appropriate functions to perform the following on a DataFrame ?
   (*i*) Calculate the sum (*ii*) Count the values
   (*iii*) Calculate the average (*iv*) Calculate the most repeated value
   (*v*) Calculate the median (*vi*) Calculate the standard deviation
   (*vi*) Calculate the variance (*viii*) Calculate the maximum value
   (*ix*) Calculate the standard deviation (*x*) Calculate the variance
6. What does info( ) and describe( ) do ?
7. Are sum( ) and add( ) functions the same ?
8. Name some functions that perform descriptive statistics on a DataFrame.
9. To consider only the numeric values for calculation, what argument do you pass to statistics functions of Pandas ?
10. Is there one function that calculates much of descriptive statistics values ? Name it.
11. What happens if mode( ) returns multiple values for a column but other columns have a single mode ?
12. What is quantile and quartile ?
13. Name the function that lets you calculate different types of quantiles.
14. Name the functions that give you maximum and minimum values in a DataFrame.
15. Name the functions that give you the index of maximum and minimum values in a DataFrame.
16. What is pivoting ? Name the two functions that you can use for pivoting.
17. What is the basic difference between pivot( ) and pivot_table( ) ?
18. What is missing data ?
19. Why is missing data filled in DataFrame with some value ?

20. Name the functions you can use for filling missing data.
21. Name the function to create histogram from a DataFrame.
22. Name some functions used to handle missing data in dataframes.
23. Name some functions used to join or combine dataframes.
24. Name two functions that can produce result similar to SQL joins.

Type B : Short Answer Questions/Conceptual Questions

1. How do you iterate over a DataFrame?
2. What are binary operations ? Name the functions that let you perform binary operations on a DataFrame.
3. What is descriptive statistics ? Name Pandas descriptive statistics functions.
4. The info( ) and describe( ) are said to be inspection functions. What do they do ?
5. What is the difference between sum and cumulative sum? How do you perform the two on DataFrame ?
6. Name and explain some cumulative functions provided by Pandas.
7. The head( ) and tail( ) extract rows or columns from a dataframe. Explain.
8. Why does Python change the datatype of a column as soon as it stores an empty value (NaN) even though it has all other values stored as integer ?
9. What do quantile and var( ) functions do ?
10. What is a quartile ? How is it different from quantile ?
11. How do you create quantiles and quartiles in Python Pandas ?
12. What is Pivoting ? How is it useful ?
13. How is pivot_table( ) different from pivot( ) when both perform pivoting ?
14. Assume that required libraries (Pandas and Numpy) are imported and dataframe **ndf** has been created as shown in solved problem 24.
    Predict the output produced by following code fragment :

```
print (ndf[ndf["age"] > 30])
print(ndf.head(2))
print(ndf.tail(3))
```

15. Given the two dataframes as :

```
In [126]: dfc1
Out[126]:
   0  1
0  2  a
1  3  b
2  4  c
```

```
In [130]: dfc2
Out[130]:
   0  1  2
0  2  3  4
2  p  q  r
```

    Why are following statements giving errors ?
    (*a*) `print(dfc1 + dfc2)`
    (*b*) `print(dfc1.sub(dfc2))`
    (*c*) `print(dfc1 * dfc2 )`
16. Consider the following code that creates two dataframes :

```
ore1 = pd.DataFrame(np.array([[20,35,25,20], [11,28,32,29]]),
        columns = ['iron', 'magnesium', 'copper','silver'])

ore2 = pd.DataFrame(np.array([[14,34,26,26], [33,19,25,23]]),
        columns = ['iron', 'magnesium', 'gold','silver'])
```

What will be the output produced by the following code fragments :

(a)
```
print(ore1 + ore2)
ore3 = ore1.radd(ore2)
print(ore3)
```

(b)
```
print(ore1 - ore2)
ore3 = ore1.rsub(ore2)
print(ore3)
```

(c)
```
print(ore1 * ore2)
ore3 = ore1.mul(ore2)
print(ore3)
```

(d)
```
print(ore1 / ore2)
ore3 = ore1.rdiv(ore2)
print(ore3)
```

17. Consider the dataframe wdf as shown below.

	minTemp	maxTemp	Rainfall	Evaporation
0	2.9	8.0	24.3	0.0
1	3.1	14.0	26.9	3.6
2	6.2	13.7	23.4	3.6
3	5.3	13.3	15.5	39.8
4	6.3	17.6	16.1	2.8
5	5.4	18.2	16.9	0.0
6	5.5	21.1	18.2	0.2
7	4.8	18.3	17.0	0.0
8	3.6	20.8	19.5	0.0
9	7.7	19.4	22.8	16.2
10	9.9	24.1	25.2	0.0
11	11.8	28.5	27.3	0.2
12	13.2	29.1	27.9	0.0
13	16.8	24.1	30.9	0.0
14	19.4	28.1	31.2	0.0
15	21.6	34.4	32.1	0.0
16	20.4	33.8	31.2	0.0
17	18.5	26.7	30.0	1.2
18	18.8	32.4	32.3	0.6
19	17.6	28.6	33.4	0.0
20	19.7	30.3	33.4	0.0

(a) Write statement(s) to calculate minimum value for each of the columns.

(b) Write statement(s) to calculate maximum value for each of the rows.

(c) Write statement(s) to calculate variance for column Rainfall.

(d) Write statement(s) to compute mean , mode median for last 10 rows.

18. Explain what the following statement is doing ?
```
df.pivot_table(df, index = 'rows', columns = 'cols', values = 'values', aggfunc = np.mean).
```

19. Write code snippet to create histogram only for column "Agg" from DataFrame mdf.

20. Consider following dataframes, namely Sprices1 and Sprices2 storing prices closing 1 year apart.

Sprices1

	closingPrice
05-01	531.35
05-02	527.93
05-03	NaN
05-04	NaN
05-05	527.81
05-06	515.14
05-07	509.96

Sprices2

	closingPrice
05-01	631.35
05-02	427.93
05-03	805
05-04	NaN
05-05	267.81
05-06	455.14
05-07	NaN

What will be the output produced by the following code fragments ?

(a)
```
print(Sprices1.isnull( ))
print(Sprices1.notnull( ))
```

(b)
```
pd.merge(Sprices1, Sprices2)
pd.merge(Sprices2, Sprices1)
```

(c)
```
pandas.concat(Sprices1, Sprices2)
```

21. Consider the dataframes of previous question and figure out why the following statement is producing an error ?
```
Sprices2.join(Sprices1)
```

22. How are merge( ) and join( ) similar or different ? Support your answer with example.

Type C : Long Answer Questions

1. Write a program to print a DataFrame one column at a time and print only first three columns.
2. Write a program to print a DataFrame one row at a time and print only first five rows.
3. Write a program that performs count, sum, max, and min functions :
   - On rows
   - On columns
4. Take a DataFrame of your choice. Write a program to calculate count of values only in a selective column.
5. Give two identical dataframes **Sales16** and **Sales17**. But **Sales17** has some values missing. Write code so that **Sales17** fills its missing values from the corresponding entries of **Sales16**.
6. Write code that just produces single True/False as a result for the presence of missing values in whole dataframe namely **df**.
7. Write code to get summary details about a dataframe object for all its columns including numeric and string columns.
8. Four Series objects *Temp1, Temp2, Temp3* and *Temp4* store the temperatures of week1, week2, week3 and week4 respectively. Create a dataframe from these four series objects where the indexes should be 'Sunday', 'Monday', ... , 'Saturday', and columns should be 'Week1', 'Week2', 'Week3' and 'Week4'.

9. From the dataframe object created in previous question, write a script to calculate :
   (*a*) average temperatures for each day of the week *i.e.*, average temperature for Mondays, Tuesdays and so on.
   (*b*) average temperature per week.
   (*c*) average temperature of whole month.

10. Given three dataframes namely Sales18, Sales19 and Sales20 storing quarterwise sales of items 'Beverages', 'Milk', 'Snack items', 'Icecreams' and 'Bakery items'.
   Write a script that prints the following :
   (*i*) Maximum yearly sales item-wise
   (*ii*) Maximum Quarterly sale for an item across three years (*e.g.*, 'Snack items' had maximum sales in Quarter 3 of year 2019).
   (*iii*) Average sales Quarterwise for all items year wise.
   (*iv*) Average sales Quarter wise for all items for all years.

11. Given a dataframe that stores the details of past 25 years' monthly sales. Some old data is however missing. Write a script to calculate average :
   - Monthly sales across years
   - Yearly sales

   Make sure that missing values do not hamper the overall result.

12. Dataframe **Det1** stores details about a state such as *TaxCollection, Dabts, Surplus, Int_rate* for its districts D1, D2, D4, D7, Dataframe **Det2** also stores details of districts D1, D2, D5, D6, D7 details as *PerCapita, AvgIncome, Population, UnEmployment*.
   Write a script that merges the details from two dataframes district-wise and then calculate the average working Income as AvgIncome/(Population-UnEmployment). Make sure to handle missing data efficiently.

13. Given two identical dataframes **Sales16** and **Sales17**. But **Sales17** has some values missing. Write code so that **Sales17** fills its missing values from corresponding entries of **Sales16**.

14. Write code that just produces single True/False as a result for the presence of missing values in whole dataframe namely **df**.

15. A dataframe **Stu** stores details like 'Name', 'Class', 'Subject_Id' for 10 students and another dataframe **Marks** stores details like 'Subject_Id' and 'Avgmarks'. Write code so that two dataframes merge data on the basis of common Subject id.

16. A dataframe **Result** stores the details as (Rollno., marks). Write a program to join it with a dataframe **Stu** storing details as (Name, Class, Subject). Make sure that all the rows from both the dataframes are part of the resultant dataframe.

3 Plotting with PyPlot

Outline

3.1 What is Data Visualization ?
3.2 Using PyPlot of Matplotlib Library
3.3 Creating Line Charts and Scatter Charts
3.4 Creating Bar Charts and Pie Charts
3.5 Customizing the Plot
3.6 Creating Histograms with PyPlot
3.7 Creating Frequency Polygons
3.8 Creating Box Plots
3.9 Plotting Data from a DataFrame

3.1 What is Data Visualization ?

Modern age has become very fast and competitive with the pace of the development in all fields of life. The reach of all businesses and facilities have gone global because of the modern technologies. This has made things easier and competitive but at the same time data have increased multifold. In fact, data have grown so big that a specific term has been coined, *'big data'*.

As you all are aware that the role of data is to empower decision makers to make decisions based on *facts, trends* and *statistical numbers* available in the form of data. But since data is so huge these days that the decision makers must be able to sift through the unnecessary, unwanted data and get the right information presented in compact and apt way, so that they can make the best decisions. For this purpose, *data visualization techniques* have gained popularity. Confused ? What is this *data visualization* ? Well, read on.

Data visualization basically refers to the graphical or visual representation of information and data using visual elements like *charts, graphs,* and *maps* etc. Data visualization is immensely useful in decision making. Data visualization unveils *patterns, trends, outliers, correlations* etc. in the data, and thereby helps decision makers understand the meaning of data to drive business decisions.

For instance, a company has to decide about, *'which advertising solution should it invest in to promote its new product?' Data visualization* is here to help – just pull out data of company's previous campaigns, and plot a bar chart comparing the performance of different platforms and bingo! the right decision is right in front.

DATA VISUALIZATION

"Data visualization basically refers to the graphical or visual representation of information and data using visual elements like *charts, graphs, maps* and so forth."

This chapter is dedicated to get you started with data visualization using Python's useful module **PyPlot** where you shall learn to represent data visually in various forms such as *line, bar* and *pie charts* etc. So, let's get started.

3.2 Using Pyplot of Matplotlib Library

For data visualization in Python, the Matplotlib library's Pyplot interface is used.

The **matplotlib** is a Python library that provides many interfaces and functionality for 2D-graphics similar to MATLAB's[1] in various forms. In short, you can call *matplotlib* as a high quality plotting library of Python. It provides both a very quick way to visualize data from Python and publication-quality figures in many formats. The **matplotlib** library offers many different named collections of methods; **PyPlot** is one such interface.

PyPlot is a collection of methods within **matplotlib** which allows user to construct 2D plots easily and interactively. **PyPlot** essentially reproduces plotting functions and behavior of MATLAB.

NOTE

PyPlot is a collection of methods within **matplotlib** library (of Python) which allows user to construct 2D plots easily and interactively.

3.2.1 Installing and Importing *matplotlib*

You shall learn to plot data using **PyPlot** but before that make sure that **matplotlib** library is installed on your computer.

- If you have installed Python using Anaconda, then **matplotlib** library is already installed on your computers. You can check it yourself by starting **Anaconda Navigator**. From Navigator window, click **Environments** and then scroll down in the alphabetically sorted list of installed packages on your computer. You will find **matplotlib** there. (Figure on next page)

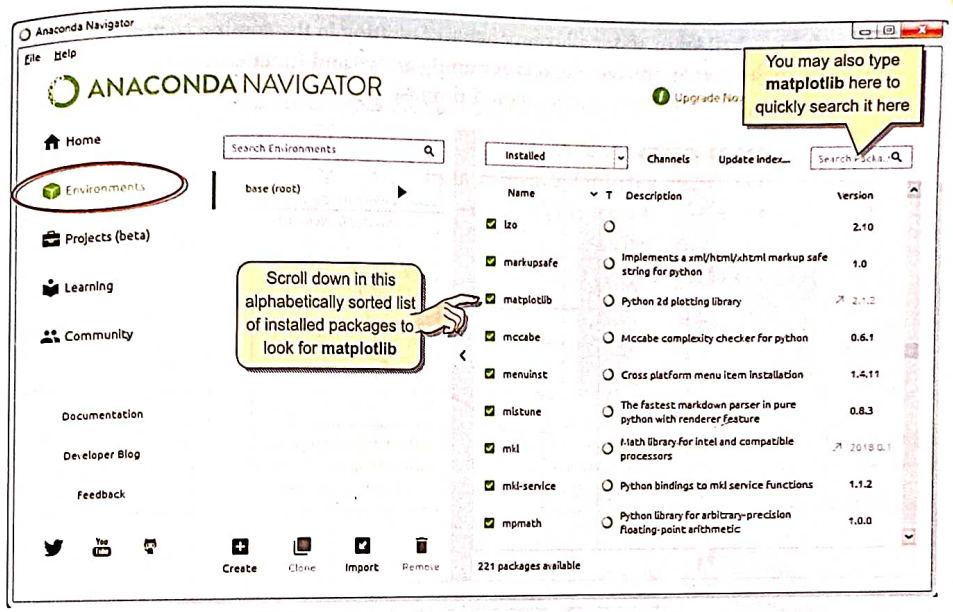
NOTE

The **matplotlib** library is preinstalled with **Anaconda distribution**; you need not install it separately if you have installed Python using Anaconda distribution.

- If you have installed Python using standard official distribution, you may need to install **matplotlib** separately as explained here.
  - First you will need to download wheel package of **matplotlib** as per Python's version installed and the platform (*MacOs or Windows or Linux*) on which it is installed. For this go to the link **https://pypi.org/project/matplotlib/#files** and download required wheel package as per your platform.
  - Next you need to install it by giving following commands on the command prompt.

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

1. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.



Importing PyPlot

In order to use **pyplot** on your computers for data visualization, you need to first import it in your Python environment by issuing one of the following commands :

```
import matplotlib.pyplot
```
This would require you to refer to every command of pyplot as matplotlib.pyplot.<command>

```
import matplotlib.pyplot as pl
```
With this, you can refer to every command of pyplot as pl.<command> as you have given an alias name to matplotlib.pyplot as pl

With the *first command* above, you will need to issue every **pyplot** command as per following syntax :

```
matplotlib.pyplot.<command>
```

But with the *second command* above, you have provided **pl** as the shorthand for **matplotlib.pyplot** and thus now you can invoke PyPlot's methods as this :

```
pl.plot(X,Y)
```

You can choose any legal identifier in place of **pl** above.

3.2.2 Working with PyPlot Methods

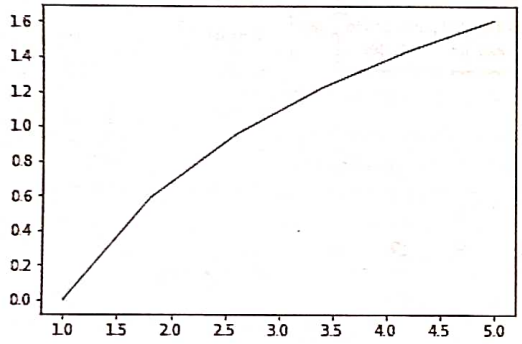
The PyPlot interface provides many methods for 2D plotting of data. The *matplotlib's* Pyplot interface lets one plot data in multiple ways such as *line chart, bar chart, pie chart, scatter chart* etc. Let us talk about how you can plot sequence or array of data using PyPlot methods.

You can easily plot the data available in the form of NumPy arrays (ndarrays) or dataframes etc. Have a look at below example where we are plotting a simple chart using an ndarray.

Although you shall learn about all components of plotting in the coming section, yet following code lines make you appreciate the use of numpy arrays and functions. (Don't worry if things are not clear right now, they will be, soon. I promise ☺)

```
>>> import numpy as np
>>> import matplotlib.pyplot as pl
>>> x = np.linspace(1, 5, 6)
>>> y = np.log(x)
>>> pl.plot(x, y)
```

Generate an array in range 1..5, having 6 elements



3.2.2A Basics of Simple Plotting

Data visualization essentially means graphical representation of compiled data. Thus, graphs and charts are very effective tools for data visualization. You can create many different types of graphs and charts using PyPlot. Some commonly used chart types are :

- Line Chart. A line chart or line graph is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments. With PyPlot, a line chart is created using **plot( )** function.
- Bar Chart. A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally. With PyPlot, a bar chart is created using **bar( )** and **barh( )** functions.
- Scatter Plot. The scatter plot is similar to a line chart, the major difference is that while line graph connects the data points with a line, scatter chart simply plots the data points to show the trend in the data. With PyPlot, a scatter chart is created using **scatter( )** function.
- Pie Chart. A pie chart (or a circle chart) is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. With PyPlot, a pie chart is created using **pie( )** function. But pie chart can plot only one data sequence unlike other chart types.
- Histogram Plot. A histogram is a type of graph that provides a visual interpretation of numerical data by indicating the number of data points that lie within a range of values. With PyPlot, a histogram is created using **hist( )** function.
- BoxPlot Chart. A Box Plot is the visual representation of the statistical five number summary of a given data set. With PyPlot, a boxplot is created using **boxplot( )** function.

Since scatter plots and line charts have similarities, we shall also cover line charts with scatter plots as both can be plotted using similar functionality, only *linestyle* is different.

Some commonly used chart types are shown in Fig. 3.1 below.

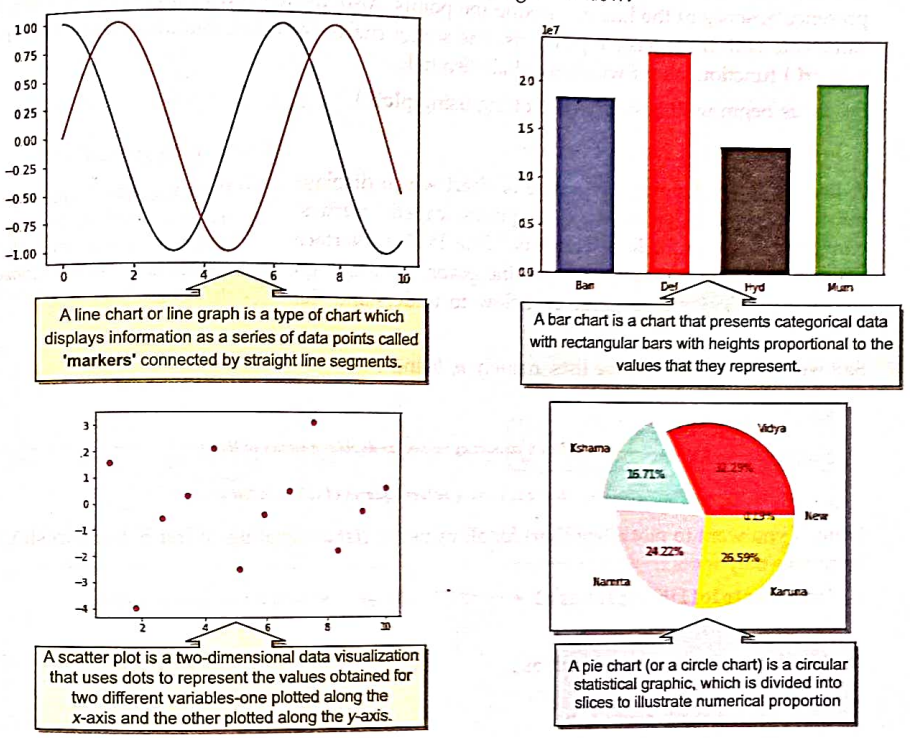


Figure 3.1 Some commonly used chart types.

Following section talks about how you can create various chart types using pyplot methods.

3.3 Creating Line Charts and Scatter Charts

Now that you know about basic plot types and styles, let us start creating them using **matplotlib.pyplot** library of Python.

To see Line, Scatter charts in action

Scan QR Code

Important

Before you plot/create any chart or graph type, make sure to import the matplotlib.pyplot library by giving command :

```
import matplotlib.pyplot
```

or

```
import matplotlib.pyplot as pl
```

preferred command as now you can issue pyplot commands as ***pl.<command>***

Along with pyplot, if you are using any NumPy or Pandas functionality, make sure to import them too using **import** commands.

In this section, we are going to talk about both line chart and scatter chart together. Why have we clubbed line chart with scatter chart, you must be thinking, right ? Well, the reason is that, for simple charts, the line charts and the scatter charts are similar. The only difference is the presence/absence of the line connecting the points. Also, using **plot( )** function of pyplot, you can create both these basic type charts. The scatter charts, however, can also be created using **scatter( )** function, about which we shall also talk.

So, let us begin with basic charts plotting using **plot( )**.

3.3.1 Line Chart using plot( ) Function

A line chart or line graph is a type of chart which displays information as a series of data points called *'markers'* connected by straight line segments. The PyPlot interface offers **plot( )** function for creating a line graph. Carefully go through the example codes given below to understand the working of **plot( )**.

> **LINE CHART**
>
> **"** A line chart or line graph is a type of chart which displays information as a series of data points called *'markers'* connected by straight line segments. **"**

Say we have following three lists, namely *a*, *b* and *c* :

```
a = [1, 2, 3, 4]
b = [2, 4, 6, 8]
c = [1, 4, 9, 16]
```

List b containing values as double of values in list a

List c containing values squares of values in list a

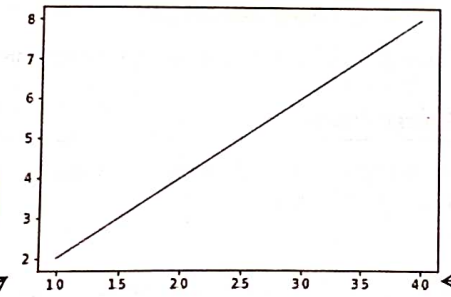
Now, if you want to plot a line chart for all values of ***list a*** vs values of ***list b***, then, in simplest form you may write :

```
import matplotlib.pyplot as pl
pl.plot(a, b)
```

The import statement is to be given just once

And Python will show you result as :

```
>>> pl.plot(a, b)
```



The values of ***list b*** are plotted on vertical axis

The values of ***list a*** are plotted on horizontal axis

> **NOTE**
>
> Data points are called ***markers***.

But are you satisfied with the result? Shouldn't the axes show the labels of the axes ? Let us say we want to label the **x-axis**, the **horizontal axis**, as **'Some values'** and the *y*-axis, the vertical axis as **'Doubled values'**. You can set *x*-axis' and *y*-axis' labels using functions **xlabel( )** and **ylabel( )** respectively, *i.e.*, :

```
<matplotlib.pyplot or its alias>.xlabel(<str>)
```

and

```
<matplotlib.pyplot or its alias>.ylabel(<str>)
```

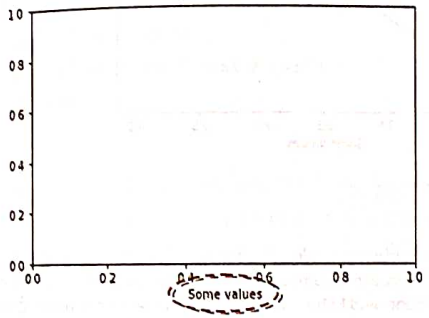
> **NOTE**
>
> Before using any **PyPlot function** or **numpy function**, make sure that these libraries are imported beforehand.

So you may write (since we used identifier **pl** as alias for **matplotlib.pyplot** in **import** statement) :

```
pl.xlabel("Some Values")
pl.ylabel("Doubled Values")
pl.plot(a, b)
```

Let us see, what Python does.

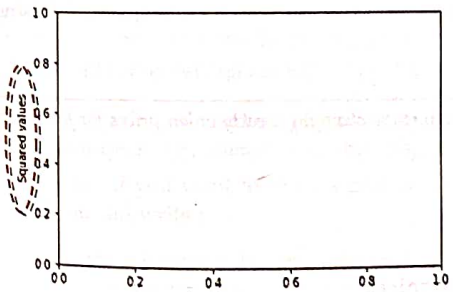
```
>>> pl.xlabel('Some values')
```



See, on IPython console's prompt, the three statements have produced three charts, even if you wanted a single chart

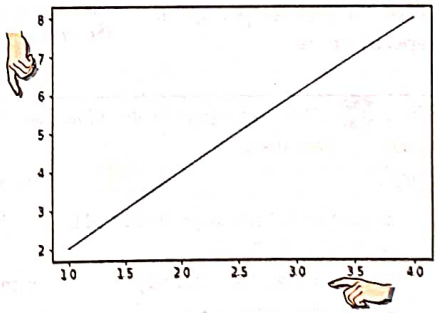
No axis titles because Python has not considered any of the previous statements

```
>>> pl.ylabel('Squared values')
```



No X-axis title only *Y-axis title* as only current statement is considered

```
>>> pl.plot(a, b)
```



See, Python has shown us *three* different charts, one for each statement we issued. But this is not what we wanted. We wanted all these three statements to take effect in a single plot.

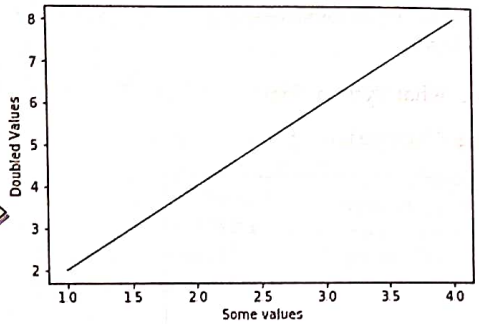
On the prompt, if you give these *three* statements, Python interactively draws plot for each statement separately.

To apply multiple statements on a single plot, do one of the following :

(*i*) Either write a script, store it with **.py** extension and then run it

(*ii*) **Or**, on the ***IPython console*** prompt, press **CTRL + ENTER** to end each statement and **ENTER** in the end. This will group multiple statements and execute them as one unit. In ***Python IDLE shell*** you can type a semicolon (;) after every code line and continue typing the next code line in the same line.

```
>>> pl.xlabel('Some values') ;
pl.ylabel('Doubled values') ;
pl.plot (a, b)
```



Now all these statements have impacted one plot as you can see the labels on both X and Y axes and graph plotted in the same plot

But the first method of writing plotting commands in a script (.py extension) is often a preferred choice for many. Also, in a script, use **<matplotlib.pyplot>.show( )** command in the end to show a plot as per given specifications.

NOTE

To apply multiple settings on single plot through a group of statements, in interactive mode, *i.e.*, by giving commands on the prompt, press Ctrl+Enter at the end of its statements and Enter only after the last statement of the group. This will cause multiple statements treated as a group and executed together. Thus, all the settings will be applied on the same plot.

Alternatively, you can store all the statements in a script (.py file) and then run the script.

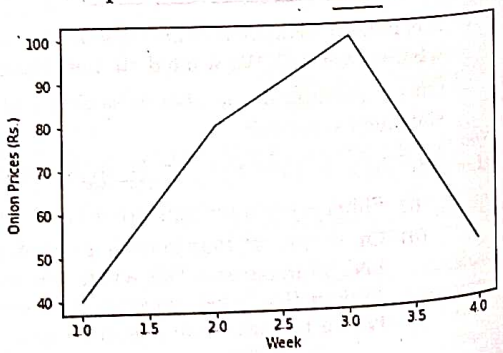
EXAMPLE 1 *Write a program to plot a line chart to depict the changing weekly onion prices for four weeks. Give appropriate axes labels.*

SOLUTION

```
import matplotlib.pyplot as plt
week = [1, 2, 3, 4]
Prices = [40, 80, 100, 50]
# Plotting line graph
plt.plot(week, Prices)
# Set the x axis and y-axis labels
plt.xlabel('Week')
plt.ylabel('Onion Prices (Rs.)')
plt.show()
```

onion prices

Output

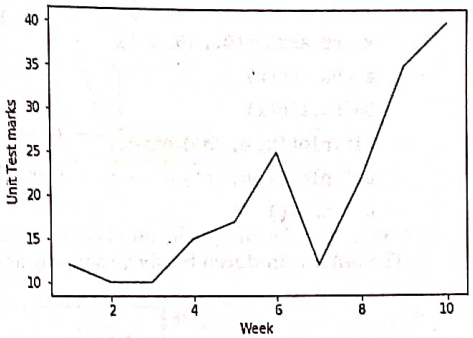


EXAMPLE 2 *Marks is a list that stores marks of a student in 10 unit tests. Write a program to plot the student's performance in these 10 unit tests.*

SOLUTION

```
import matplotlib.pyplot as plt
week = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Marks = [12, 10, 10, 15, 17, 25, 12, 22, 35, 40]
# Plotting line graph
plt.plot(week, Marks)
# Set the x axis and y-axis labels.
plt.xlabel('Week')
plt.ylabel('Unit Test marks')
plt.show()
```

Output



3.3.2 Specifying Plot Size and Grid

While plotting with PyPlot, you may find that the generated graph/plot is too small or too large. In such cases, you may want to set up size for your plot.

For this you may set up figure size as :

```
<matplotlib.pyplot>.figure( figsize = (<width>, length>)
```

e.g.,

```
import matplotlib.pyplot as plt
plt.figure(figsize = (15,7))  #15 units wide(x coordinates) & 7 units long(y coordinates)
```

You will find the use of *figure size* in many examples (*e.g., example 3*) in this chapter.

Also, if you want to show a grid on the plot, you can write :

TIP

Use <pyplot>.figure(figsize = (<width>, <length>)) to change the plot size as per your requirements.

```
import matplotlib.pyplot as plt
plt.grid(True)
```

3.3.3 Applying Various Settings in plot( ) Function

The **plot( )** function allows you specify multiple settings for your chart/graph such as :

- color (line color/marker color)
- marker size
- marker type
- and so forth.

Let us see how you can apply there settings through *plot( )* function.

3.3.3A Changing Line Color and Style

To Change Line Color

You can specify the color code next to the data being plotted in plot( ) function as shown below :

```
<matplotlib.pyplot>.plot(<data1>, [,data2], <color code> )
```

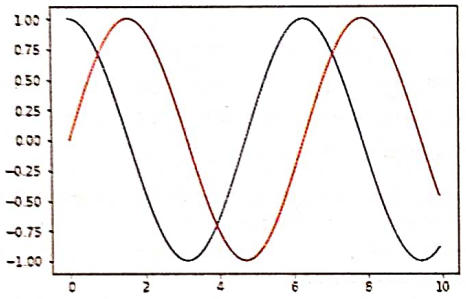
You can use color codes as: 'r' for red, 'y' for yellow, 'g' for green, 'b' for blue etc. (Complete color codes' list has been given in Table 3.1), *e.g.*,

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0., 10, 0.1)
a = np.cos(x)
b = np.sin(x)
plt.plot(x, a, 'b')   <-- Color code blue for ndarray a
plt.plot(x, b, 'r')   <-- Color code red for ndarray b
plt.show()
```

NOTE

Even if you skip the color information in **plot( )**, Python will plot multiple lines in the same plot with different colors but these colors are decided internally by Python.

The output produced by above script is as shown below :



Please note that even if you skip the color information in *plot( )*, Python will plot multiple lines in the same plot with different colors but these colors are decided internally by Python.

Table 3.1 *Base Color Codes of matplotlib*

character	color	character	color	character	color
'b'	blue	'm'	magenta	'c'	cyan
'g'	green	'y'	yellow	'w'	white
'r'	red	'k'	black		

In addition, you can also specify colors in many other ways, including full color names (*e.g.*, 'red', 'light green' 'olive', 'silver', 'purple', 'dark grey', 'brown' etc.), hex strings ('#008000') etc.

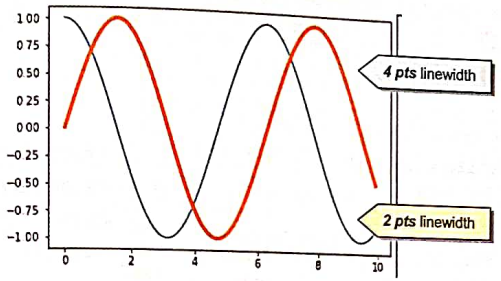
To Change Line Width

You can give additional argument in *plot( )* as **linewidth = <width>** where you have to specify the width value in points[2], *e.g.*,

```
: #x, a, b are same as earlier code
plt.plot(x, a, linewidth = 2)
plt.plot(x, b, linewidth = 4)
```

2. Line width or thickness is measured in points e.g., 0.75 points, 2.0 points etc.

The result produced by above code is as shown below :



To change the line style, you can add following additional optional argument in *plot( )* function :

```
linestyle or ls = ['solid' | 'dashed', 'dashdot', 'dotted']
```

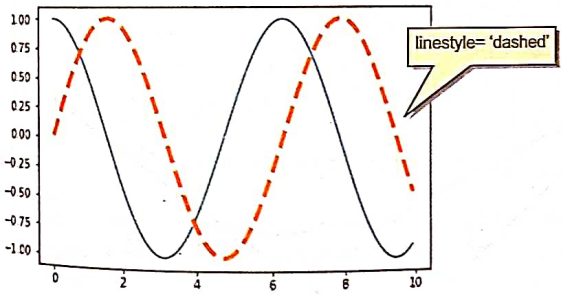
e.g.,

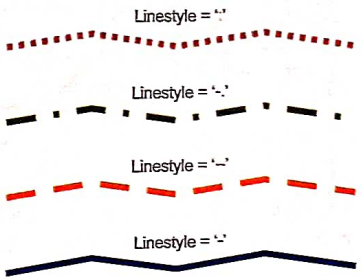
```
: #x, a, b are same as earlier code
plt.plot(x, a, linewidth = 2)
plt.plot(x, b, linewidth = 4, linestyle = 'dashed')
```

NOTE

Possible line styles are : **'solid'** for *solid line*, **'dashed'** for *dashed line*, **'dotted'** for *dotted line* and **'dashdot'** for *dashdotted line*.

The result produced by above code is as shown here :





EXAMPLE 3 *Tanushree is doing some research. She has a stored line of Pascal's triangle numbers as ar2 as shown below :*

```
ar2 = [1, 7, 21, 35, 35, 21, 7, 1]
```

*She wants to plot the sine (**numpy.sin( )**), cosine(**numpy.cos( )**) and tangent values (**numpy.tan()**) for the same array (ar2).*

- *She wants cyan color for sine plot line, red color for cosine plot line and the black color for tangent plot line.*
- *Also, the tangent line should be dashed.*

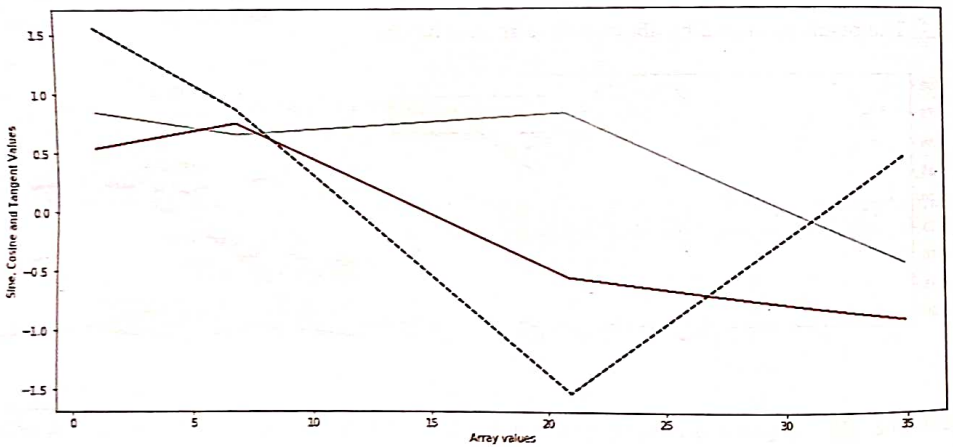
Write a program to accomplish all this.

SOLUTION

```
import matplotlib.pyplot as plt
import numpy as np
```

```
ar2 = [1, 7, 21, 35, 35, 21, 7, 1]
# calculating sin(), cos() and tan() values
s2 = np.sin(ar2)
c2 = np.cos(ar2)
t2 = np.tan(ar2)
# plotting line chart
plt.figure(figsize = (15, 7))          # figure size specified
plt.plot(ar2, s2, 'c')                 # sine line with cyan
plt.plot(ar2, c2, 'r')                 # cosine line with red
plt.plot(ar2, t2, 'k', linestyle = "dashed")   # tan line with black
# Set the x axis label of the current axis.
plt.xlabel('Array values')
# Set the y axis label of the current axis.
plt.ylabel('Sine, Cosine and Tangent Values')
plt.show()
```

Output



3.3.3B Changing Marker Type, Size and Color

The *data points* being plotted on a graph/chart are called **markers**. To change *marker type*, its *size* and color, you can give following additional optional arguments in *plot( )* function :

```
marker = <valid marker type> , markersize = <in points>, markeredgecolor = <valid color>
```

You can control the *type of marker i.e., dots or crosses or diamonds* etc. by specifying desired marker type from the table below. If you do not specify marker type, then data points will not be marked specifically on the line chart and its default type will be the same as that of the line type.

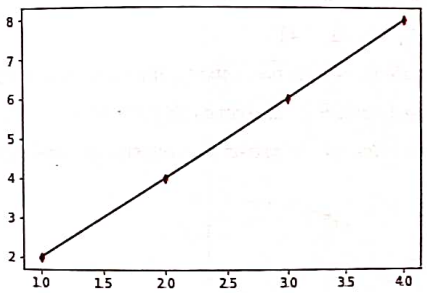
Also, the ***markersize*** is to be specified in points and ***markeredgecolor*** must be a valid color.

Table 3.2 *Marker Types for Plotting*

marker	description	marker	description	marker	description
'.'	point marker	's'	square marker	'3'	tri_left marker
','	pixel marker	'p'	pentagon marker	'4'	tri_right marker
'o'	circle marker	'*'	star marker	'v'	triangle_down marker
'+'	plus marker	'h'	hexagon1 marker	'^'	triangle_up marker
'x'	x marker	'H'	hexagon2 marker	'<'	triangle_left marker
'D'	diamond marker	'1'	tri_down marker	'>'	triangle_right marker
'd'	thin_diamond marker	'2'	tri_up marker	'|', '_'	vline, hline markers

Now consider following example where p = [1, 2, 3, 4] and q = [2, 4, 6, 8]

```
>>> plt.plot (p, q, 'k', marker = 'd', markersize = 5, markeredgecolor = 'red')
```



Line color is black (color 'k')
Marker type = small diamond ('d')
Marker size = 5 points
Marker color = 'red'

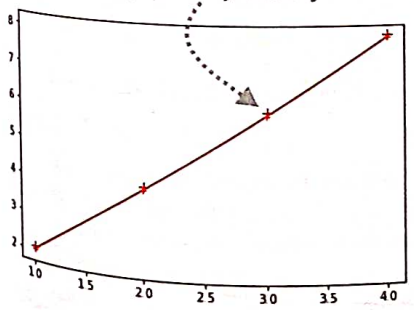
To see marker types in action

Scan QR Code

Note that you can combine the marker type with color code, *e.g.*, 'r+' when given for ***line color*** marks the *color* as **'red'** and ***markertype*** as plus('+'). Consider two following examples combining these :

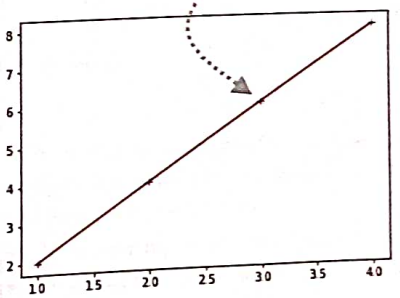
Line color and *marker style* combined so marker takes same color as line

```
>>> plt.plot (p, q, 'r+', linestyle = 'solid')
```



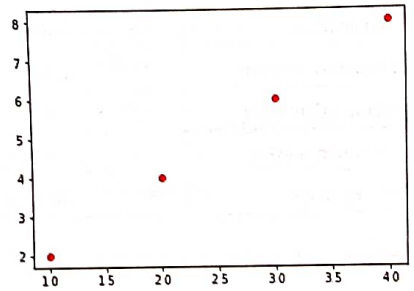
Here *marker color* separately specified

```
>>> plt.plot (p, q, 'r+', linestyle = 'solid',
                markeredgecolor = 'b')
```



See, when you do not specify **markeredgecolor** separately in **plot( )**, the marker takes the same color as the line. Also, if you do not specify the **linestyle** separately along with *linecolor- and-marketstyle-combination-string* (*e.g.*, 'r+' above), Python will only plot the markers and not the line, (*see* below). To get the line, specify **linestyle** argument also as shown above.

```
>>> plt.plot (p, q, 'ro')
```



if you do not specify the **linestyle** argument separately along with linecolor-&-marketstyle-string (e.g., 'r+' or 'bo' etc.), Python will only plot the markers and not the line

EXAMPLE 4 *First 10 terms of a Fibonacci series are stored in a list namely* **fib** :

```
fib = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

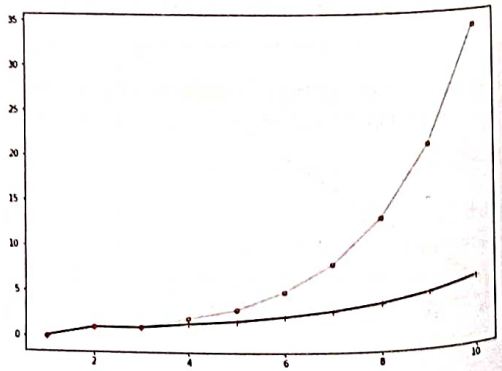
Write a program to plot Fibonacci terms and their square-roots with two separate lines on the same plot.

- *The Fibonacci series should be plotted as a cyan line with 'o' markers having size as 5 and edge-color as red.*
- *The square-root series should be plotted as a black line with '+' markers having size as 7 and edge-color as red.*

SOLUTION

```
import matplotlib.pyplot as plt
import numpy as np
fib = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
sqfib = np.sqrt(fib)
fib = [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
sqfib = np.sqrt(fib)
plt.figure(figsize = (10, 7))
plt.plot(range(1,11), fib, 'co', markersize = 5, linestyle = "solid", markeredgecolor = 'r')
plt.plot(range(1,11), sqfib, 'k+', markersize = 7, linestyle = "solid", markeredgecolor = 'r')
plt,show( )
```

Output



EXAMPLE 5 *Given a series* **nfib** *that contains reversed Fibonacci numbers with Fibonacci numbers as shown below :*

```
[ 0, –1, –1, –2, –3, –5, –8, –13, –21, –34, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

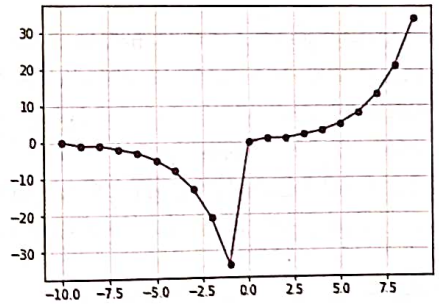
Write a program to plot **nfib** *with following specifications:*

- *The line color should be magenta*
- *The marker edge color should be black with size 5*
- *Grid should be displayed.*

SOLUTION

```
import matplotlib.pyplot as plt
import numpy as np
nfib = [ 0, –1, –1, –2, –3, –5, –8, –13, –21, –34, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
plt.plot(range(-10, 10), nfib, 'mo', markersize = 5, markeredgecolor = 'k', linestyle = 'solid')
plt.grid(True)
plt.show()
```

Output



3.3.4 Creating Scatter Charts

The scatter charts can be created through two functions of PyPlot library :

(*i*) plot( ) function

(*ii*) scatter( ) function

Let us first talk about scatter charts using plot( ) function.

SCATTER CHART

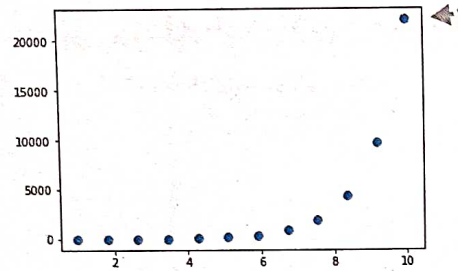
"The **scatter chart** is a graph of plotted points on two axes that show the relationship between two sets of data."

3.3.4A Scatter Charts using plot( ) Function

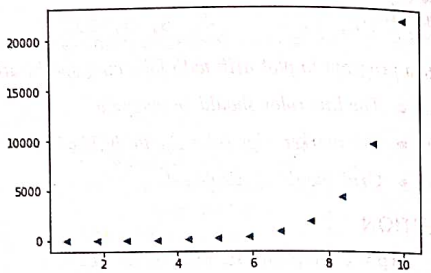
As it must be evident from the previous section that if you specify the **linecolor-and-markerstyle-string** (*e.g.*, "r+", or "bo" etc.) without the **linestyle** argument, then the plot created resembles a scatter chart as only the datapoints are plotted now.

In fact, in plot( ) function, whenever you specify market type/style, whether with color or without color, and do not give **linestyle** argument, plot( ) will create a scatter chart, *e.g.*,

```
>>> pl.plot(a1, a4, "o", markersize = 8)
```




```
>>> pl.plot (a1, a4, "<")
```



So with **marker-type specified** plot( ) creates the scatter chart in the absence of ***linestyle*** argument.

So full useful syntax of **plot( )** can be summarised as :

```
plot( <data1> [,<data2>] [, … <colorcode and marker type>] [, <linewidth>]
      [, <linestyle>] [,<marker>][, <markersize>] [,<markeredgecolor>] )
```

where,

- ***data1, data2*** are sequences of values to be plotted on x-axis and y-axis respectively
- Rest of the arguments affect the look and format of the line/scatter chart as given below :
  - **color code with markertype** color and marker symbol for the line chart
  - **linewidth** width of the line in points (a float value)
  - **linestyle** can be ['solid' | 'dashed', 'dashdot', 'dotted' or take markertype string]
  - **marker** a symbol denoting valid marker style such as '–', 'o', 'x', '+', 'd', and others etc.
  - **markersize** size of marker in points (a float value)
  - **markeredgecolor** color for markers; should be a valid color identified by PyPlot

** please note that this is still not the complete and full syntax of plot( ).*
Complete syntax coverage of plot( ) is beyond the scope of the book.

NOTE

If you are working on a Jupyter Notebook, you can make the plots appear inline in the notebook by using following magic function :

```
%matplotlib inline
```

If you don't use above function, the plots will appear in a pop up window as with other methods.

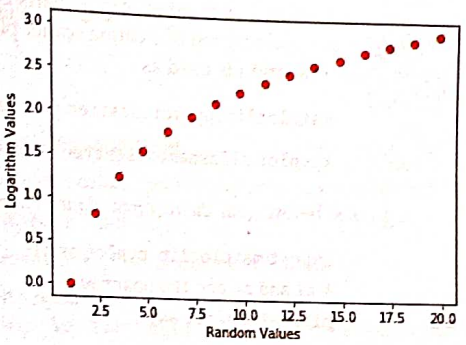
EXAMPLE **6** *Create an array, A, in the range 1 to 20 with values 1.25 apart. Another array, B, contains the log values of the elements in the array A.*

Write a program to create a scatter plot of first vs. second array (array A vs. B) with red circle markers ; specify the x-axis (containing first array's values) title as 'Random Values' and y-axis title as 'Logarithm Values'.

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
A = np.arange(1, 20, 1.25)
B = np.log(A)
plt.plot(A, B, 'ro')
plt.xlabel('Random Values')
plt.ylabel('Logarithm Values')
plt.show( )
```

Output

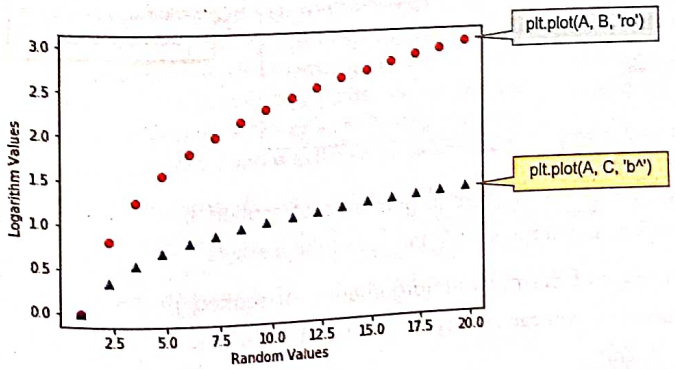


EXAMPLE **7** *Consider the arrays of previous example and create an array C that stores the log10 values of the elements in the array A. Write a program to modify previous example program so that a scatter plot of array A vs. C is also plotted with blue triangular markers.*

SOLUTION

```
# making changes in the previous example's program code
import numpy as np
import matplotlib.pyplot as plt
A = np.arange(1, 20, 1.25)
B = np.log(A)
C = np.log10(A)
plt.plot(A, B, 'ro')
plt.plot(A, C, 'b^')
plt.xlabel('Random Values')
plt.ylabel('Logarithm Values')
plt.show()
```

Output



3.3.4B Scatter Charts using scatter( ) Function

Another method of creating scatter charts is using the scatter( ) function of pyplot library. It is, more powerful method of creating scatter plots than the plot( ) function. In its simplest form, the scatter( ) function is used as :

```
matplotlib.pyplot.scatter (<array1>, <array2>)
```

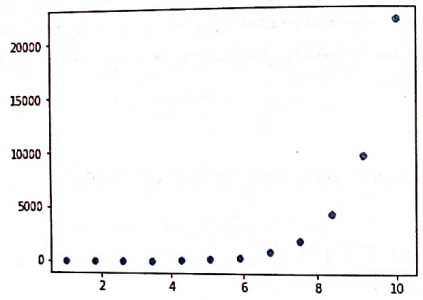
or

```
<pyplot aliasname>.scatter (<array1>, <array2>)
```

e.g., see below. (*All the required libraries have already been imported*)

```
import matplotlib.pyplot as pl
# a1 and a4 are two ndarrays
pl.scatter(a1, a4)
```

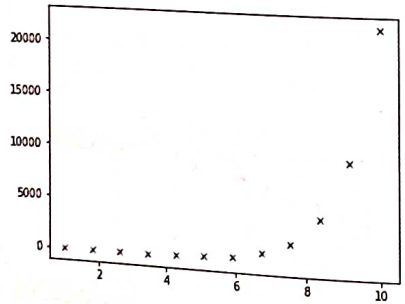
```
>>> pl.scatter (a1, a4)
```



Specifying Marker Type (argument marker of scatter( ))

You can specify marker type using **marker** argument of the scatter( ) function. The marker argument can take any value as per Table 3.2, given earlier, *e.g.,*

```
>>> pl.scatter(a1, a4, marker= "x")
```



Notice, data point markers are as per the **marker** argument of the scatter( ) function.

Specifying size of the markers (argument s of scatter( ))

Using argument **s**, you can specify the size of the markers.

Specifying color of the markers (argument c of scatter( ))

Using argument **c**, you can specify the color of the markers.

In other words, detailed syntax (not the full and complete syntax) of scatter( ) function of pyplot library is :

To see scatter( ) function in action

Scan QR Code

```
matplotlib.pyplot.scatter(x, y, s = None, c = None, marker = None)
```

Parameters :

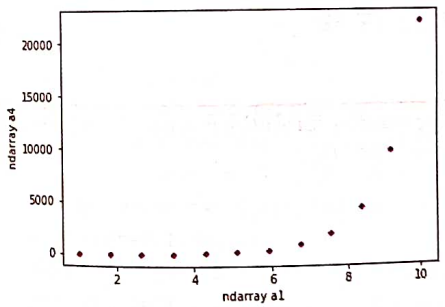
x, y	The data positions.
s	The marker size in points**2. Optional argument.
c	marker color, sequence, or sequence of color, optional argument (refer Table 3.1)
marker	MarkerStyle, optional argument (refer Table 3.2)

For complete syntax of scatter( ), you may refer to the PyPlot documentation.

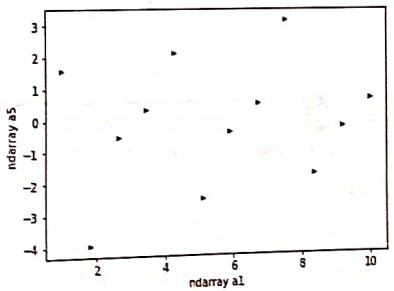
The primary difference of scatter( ) from plot( ) is that it can be used to create scatter plots where the **properties of each individual point** (*size, face color, edge color,* etc.) can be individually controlled or mapped to data.

Consider following examples of scatter( ). As earlier mentioned, you can execute group of statements in a group mode on IPython console by ending then with CTRL+ENTER instead of ENTER; or on IDLE console, you can put semicolon between multiple commands on the same line.

```
>>> pl.scatter (a1, a4, s = 9, c = 'm', marker = "D") ;
pl.xlabel("ndarray a1") ;
pl.ylabel("ndarray a4")
```




```
>>> pl.scatter (a1, a5, s = 9, c = 'b', marker = ">") ;
pl.xlabel("ndarray a1");
pl.ylabel("ndarray a5")
```



Specifying Varying Colors and Sizes for Data Points

All the examples of scatter chart that you have seen so far have used same color and size for the markers. But the scatter( ) allows you to specify different sizes and colors for individual data points. For this purpose, you need to specify an array of colors having the same shape as arrays being plotted (*x* and *y*) as the value of **c argument** and an array of sizes having the same shape as arrays being plotted (*x* and *y*) as the value of **s argument**.

When the *c* or *s* argument have a sequence/array with multiple color/size values, then each color/size is applied to different data points in order, *i.e.,* first color and size to the first data point, second color/size to the second data point, and so on.

NOTE

The primary difference of scatter( ) from plot( ) is that it can be used to create scatter plots where the **properties of each individual point** (*size, face color, edge color,* etc.) can be individually controlled or mapped to data.

To understand this, carefully read following lines :

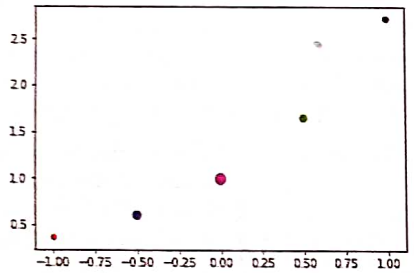
```
# import statements for pyplot and numpy
arr1 = np.linspace (–1, 1, 5)      # arr1 with 5 data points created
arr2 = np.exp(arr1)                # arr2 also has 5 data points created
colarr = ['r', 'b', 'm', 'g', 'k'] #colarr is a sequence of colors with same shape as arr1
sarr = [20, 60, 100, 45, 25]       # sarr is a sequence of sizescolor with same shape as arr1
pl.scatter(arr1, arr2, c = colarr, s = sarr)
```

See the c argument has sequence of colors; each color will be assigned to different data points.

The s argument has sequence of sizes, one size for each data point

And the output produced by above code will be :

```
>>> pl.scatter(arr1, arr2, c = colarr, s = sarr)
```

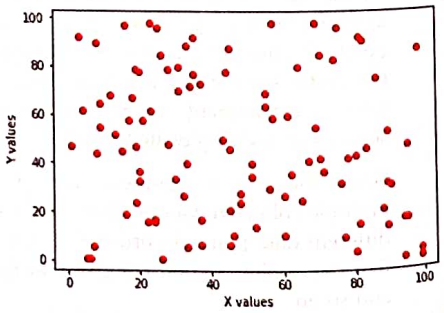


EXAMPLE 8 *Write a program to plot a scatter graph taking a random distribution in X and Y (both with shape as (100,) having randomly generated integers) and plotted against each other.*

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
X = np.random.randint(1, 100, size =(100,))
Y = np.random.randint(1, 100, size =(100,))
plt.scatter(X, Y, color = 'r')
plt.xlabel("X values")
plt.ylabel("Y values")
plt.show()
```

Output

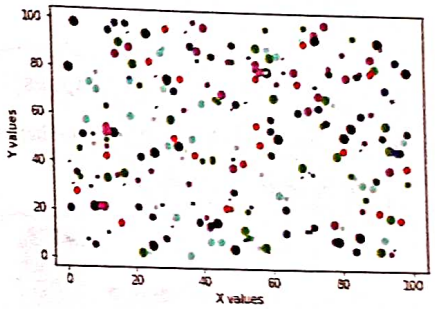


EXAMPLE 9 *Write a program to plot a scatter graph taking two random distributions in X1 and Y1 (both with shape as (250,) having randomly generated integers) and plotted against each other with varying sizes and varying colors.*

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
X1 = np.random.randint(1, 100, size = (250,))
Y1 = np.random.randint(1, 100, size = (250,))
size = range(1, 60, 5)
colr = ['r', 'b', 'c', 'm', 'g', 'k']
plt.scatter(X1,Y1, s = size, c = colr)
plt.xlabel("X values")
plt.ylabel("Y values")
plt.show( )
```

Output

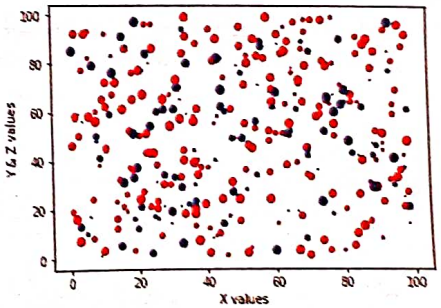


EXAMPLE 10 *Write a program to plot a scatter graph taking three random distributions in X1, Y1 & Z1 (all three with shape as (250,) having randomly generated integers) and plotted against each other with varying sizes.*

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
X1 = np.random.randint(1, 100, size =(250,))
Y1 = np.random.randint(1, 100, size =(250,))
Z1 = np.random.randint(1, 100, size =(250,))
size = range(1, 60, 5)
plt.scatter(X1,Y1, color = 'r', s = size)
plt.scatter(X1,Z1, color = 'r', s = size)
plt.xlabel("X values")
plt.ylabel("Y & Z values")
plt.show()
```

Output



3.4 Creating Bar Charts and Pie Charts

A ***Bar Graph*** or a ***Bar Chart*** is a graphical display of data using bars of different heights. A bar chart can be drawn *vertically* or *horizontally* using rectangles or bars of different heights/widths PyPlot offers **bar( )** function to create a bar chart where you can specify the sequences for *x*-axis and corresponding sequence to be plotted on *y*-axis. Each *y* value is plotted as bar on corresponding *x*-value on *x*-axis.

But before creating any bar chart, like you did in other chart programs, make sure to *import* ***matplotlib.pyplot*** *and also* ***numpy/pandas****, if you are using any numpy function, series or dataframe objects. Likewise, if you want that multiple commands affect a common bar chart, then either store all the related statements in a Python script(.py file) with last statement being <matplotlib.pyplot>.show( ), or group them using Ctrl + Enter on IPython console (Spyder) or using semicolon between statements on IDLE console.*

BAR GRAPH/CHART

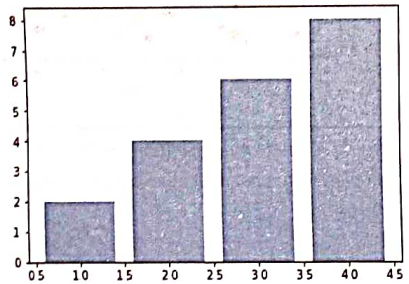
" A Bar Graph / Chart is a graphical display of data using bars of different heights. "

Now consider the following code. If we have *three* sequences **a**, **b**, and **c** as :

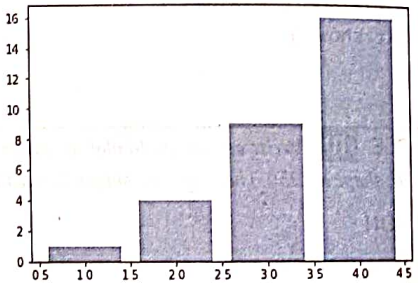
```
a, b, c = [1, 2, 3, 4], [2, 4, 6, 8], [1, 4, 9, 16]
```

then commands **matplotlib.pyplot.bar(a, b) and matplotlib.pyplot.bar(a,c)** will produce result as:

```
>>> plt.bar(a, b)
```



```
>>> plt.bar(a, c)
```



Notice, first sequence given in the **bar( )** forms the *x-axis* and the second sequence values are plotted on *y*-axis. If you want to specify *x-axis* and *y-axis labels*, then you need to give commands :

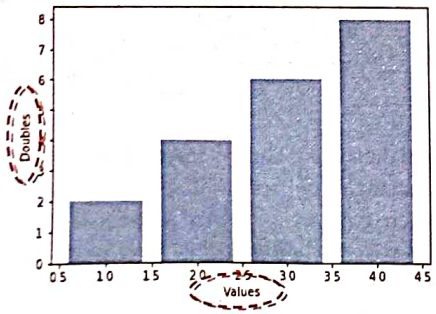
```
matplotlib.pyplot.xlabel(<label string>)
matplotlib.pyplot.ylabel(<label string>)
```

NOTE

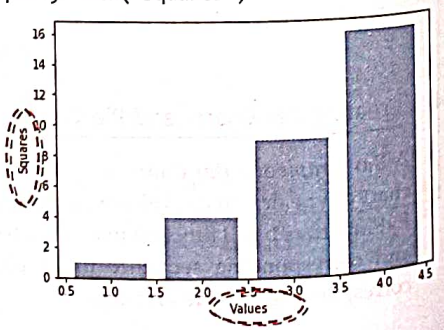
The first sequence given in the **bar( )** forms the *x-axis* and the second sequence values are plotted on *y-axis*.

Also, you need to group multiple chart commands either in a script file (*.py* file) or by pressing CTRL+ENTER in IPython (Spyder) or in between semicolons on Python IDLE (*see* below).

```
>>> plt.bar(a, b) ; plt.xlabel('Values') ;
plt.ylabel('Doubles')
```




```
>>> plt.bar(a, c) ; plt.xlabel('Values') ;
plt.ylabel('Squares')
```

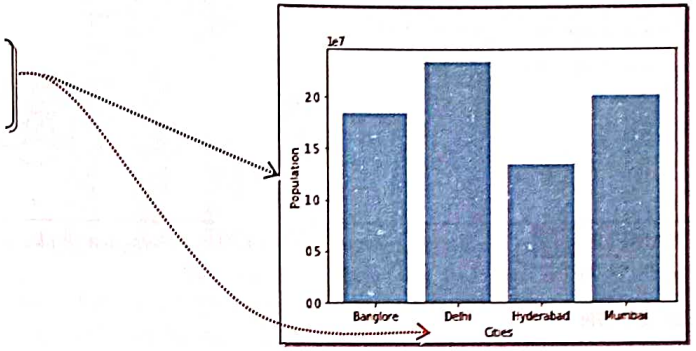


Consider another example :

```
Cities = ['Delhi', 'Mumbai', 'Bangalore', 'Hyderabad']
Population = [23456123, 20083104, 18456123, 13411093]
plt.bar(Cities, Population)
plt.xlabel('Cities')
plt.ylabel('Population')
plt.show()
```

NOTE

The order of bars plotted may be different from the order in actual data sequence.



Note that the order of bars plotted is different from the order in actual data sequence *e.g.*, the **Cities** sequence being plotted contains values in the order ("Delhi", "Mumbai", "Bangalore", "Hyderabad") but when plotted their order is different, *i.e.*, "Bangalore", "Delhi", "Hyderabad" and "Mumbai" – the sorted order.

To see Bar charts in action

Scan QR Code

Used in Examples : Medal tally of Commonwealth games 2018

Country	Gold	Silver	Bronze	Total
Australia	80	59	59	198
England	45	45	46	136
India	26	20	20	66
Canada	15	40	27	82
New Zealand	15	16	15	46
South Africa	13	11	13	37
Wales	10	12	14	36
Scotland	9	13	22	44
Nigeria	9	9	6	24
Cyprus	8	1	5	14

Reference 3.1

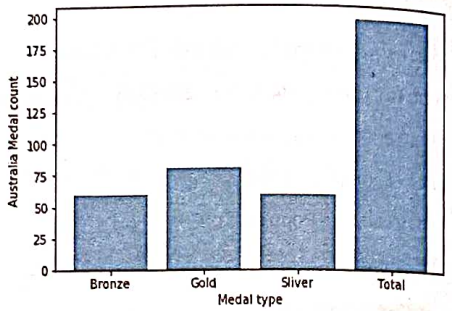
EXAMPLE 11 *Consider the reference table 3.1. Write a program to plot a bar chart from the medals won by Australia.*

SOLUTION

```
import matplotlib.pyplot as plt
Info = ['Gold', 'Silver', 'Bronze', 'Total']
```

```
Australia =[80, 59, 59,198]
plt.bar(Info, Australia)
plt.xlabel("Medal type")
plt.ylabel("Astralia Medal count")
plt.show( )
```

Output

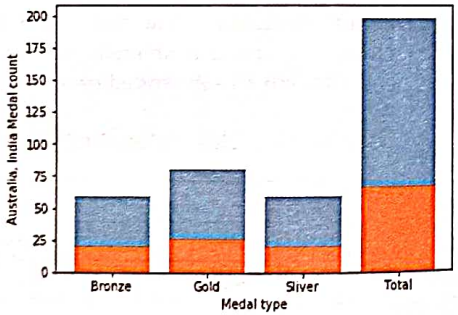


EXAMPLE **12** *Consider the reference table 3.1. Write a program to plot a bar chart from the medals won by Australia. In the same chart, plot medals won by India too.*

SOLUTION

```
import matplotlib.pyplot as plt
Info = ['Gold', 'Silver', 'Bronze', 'Total']
Australia = [80, 59, 59,198]
India = [26, 20, 20, 66]
plt.bar(Info, Australia)
plt.bar(Info, India)
plt.xlabel("Medal type")
plt.ylabel("Australia, India Medal count")
plt.show()
```

Output



If you look at the output of example 12, you can see that two countries' medals have been shown by two different colours, BUT still you cannot make out which colour depicts which country. For this, the proper way to depict is to add legends, which we shall cover in a later Section **3.5 Customising the Plot.**

3.4.1 Changing Widths of the Bars in a Bar Chart

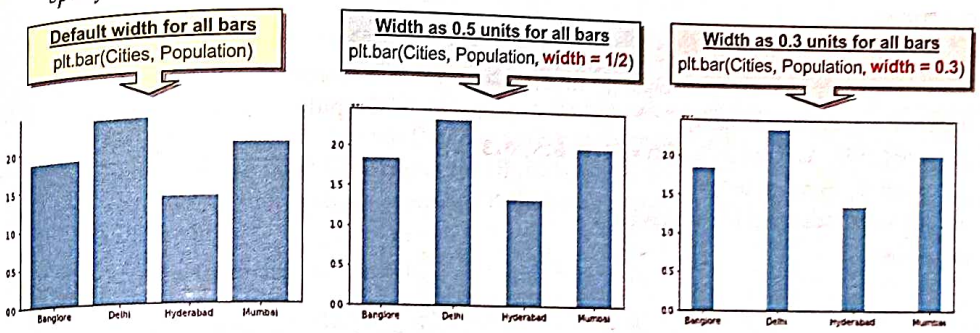
By default, bar chart draws bars with equal widths and having a default width of 0.8 units on a bar chart. That is, all bars have the width same as the *default width*. But you can always change the widths of the bars.

- You can specify a different width (other than the default width) for all the bars of a bar chart.
- You can also specify different widths for different bars of a bar chart.

(i) **To specify common width (other than the default width) for all bars,** you can specify ***width*** **argument** having a scalar float value in the **bar( )** function, *i.e.*, as :

```
<matplotlib.pyplot>.bar(<x-sequence>, <y-sequence>, width = <float value>)
```


Consider the following three bar charts where the first chart has the bars with default ***widths***, second chart with a ***width as per expression 1/2*** (*i.e.*, 0.5 units) and the third chart has ***width specified as 0.3 units.***



So, when you specify a scalar value *i.e.*, a single value for **width** argument, then that width is applied to all the bars of the bar chart.

NOTE

If you specify a scalar value (a single value) for **width** argument, then that width is applied to all the bars of the bar chart.

(ii) **To specify different widths for different bars of a bar chart,** you can specify ***width*** **argument** having a sequence (such as lists or tuples) containing widths for each of the bars, in the **bar( )** function, *i.e.*, as :

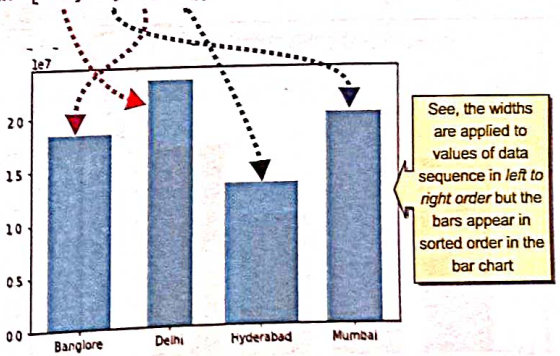
```
<matplotlib.pyplot>.bar(<x-sequence>, <y-sequence>, width = <width values sequence>)
```

The widths given in the sequence are applied from left to right, *i.e.*, the first value of the sequence specifies the width of first value of data sequence, second value specifies the width for the second value of data sequence, and so on. The width values are *float* values. Please note **that the width sequence must have widths for all the bars** (*i.e.*, its length must match the length of data sequences being plotted) otherwise Python will report an error, the [*ValueError: shape mismatch error*]. Consider the following bar chart :

```
plt.bar(Cities, Population, width =[0.5, .6, .7, .8])
```

So as per above **bar( )**, the widths 0.5, .6, .7, .8 are applicable to **Cities** values in order *i.e.*, "**Delhi**"'s bar will be 0.5 units wide; "**Mumbai**"'s bar will be 0.6 units wide ; "**Bangalore**"'s bar will be 0.7 units wide ; and "**Hyderabad**"'s bar will be 0.8 units wide.

The output produced by above command will be as shown.



NOTE

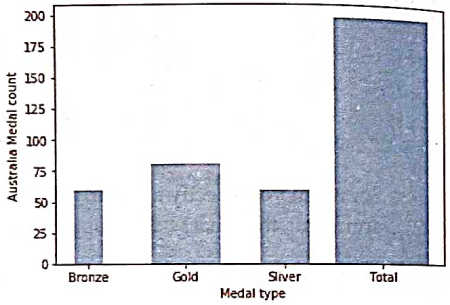
The width values' sequence in a **bar( )** must have widths for all the bars, *i.e.*, its length must match the length of data sequences being plotted, otherwise Python will report an error.

EXAMPLE 13 *Consider the reference table 3.1. Write a program to plot a bar chart from the medals won by Australia. Make sure that the Gold, Silver, Bronze and Total tally is represented through different widths.*

SOLUTION

```
import matplotlib.pyplot as plt
Info = ['Gold', 'Silver', 'Bronze', 'Total']
Australia =[80, 59, 59,198]
plt.bar(Info, Australia, width = [0.7, 0.5, 0.3, 1])
plt.xlabel("Medal type")
plt.ylabel("Australia Medal count")
plt.show()
```

Output



3.4.2 Changing Colors of the Bars in a Bar Chart

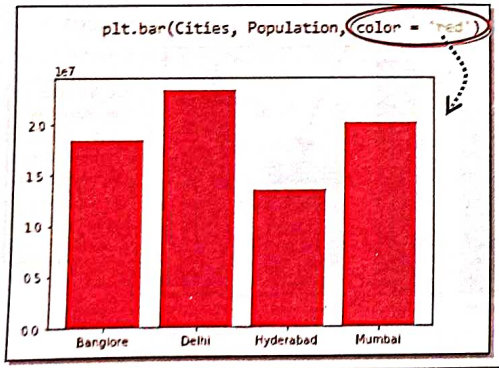
By default, a bar chart draws bars with same default color. But you can always change the color of the bars.

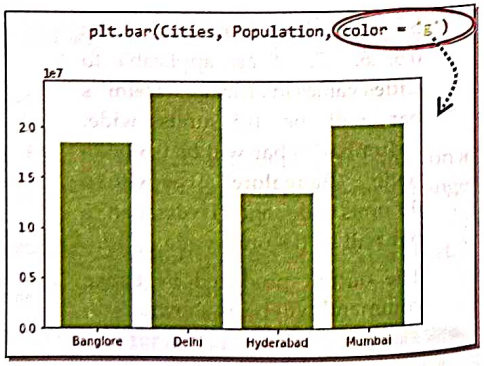
- You can specify a different color for all the bars of a bar chart.
- You can also specify different colors for different bars of a bar chart.

(*i*) **To specify common color (other than the default color) for all bars**, you can specify *color* **argument** having a **valid color code/name** in the **bar( )** function, *i.e.*, as :

```
<matplotlib.pyplot>.bar(<x-sequence>, <y-sequence>, color = <color code/name>)
```

The *color* given with **color** argument will be applied to all the bars, *e.g.*, consider this :





When you specify single color name or single color code with color argument of the bar() function, the specified color is applied to all the bars of the bar chart, *i.e.*, all bars of the bar chart have the same common color.

(*ii*) **To specify different colors for different bars of a bar chart**, you can specify *color* argument having a sequence (such as lists or tuples) containing colors for each of the bars, in the **bar( )** function, *i.e.*, as :

```
<matplotlib.pyplot>.bar(<x-sequence>, <y-sequence>, color = <color names/codes sequence>)
```

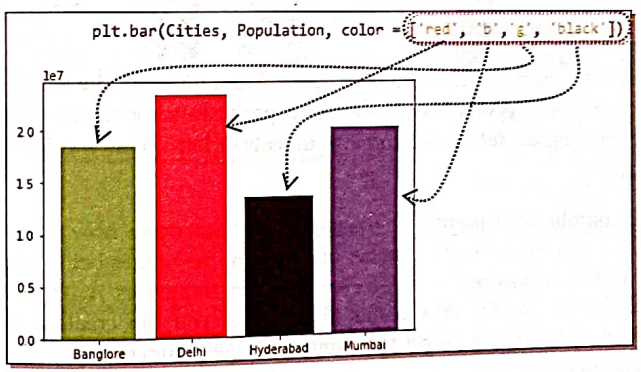
The colors given in the sequence are applied from left to right, *i.e.*, the first value of the sequence specifies the color of first value of data sequence, second value specifies the color for the second value of data sequence, and so on. The color values are *valid color codes or names*. Please note **that the color sequence must have colors for all the bars** (*i.e.*, its length must match the length of data sequences being plotted) otherwise Python will report an error, the [*ValueError: shape mismatch error*].

Consider the following bar chart :

```
plt.bar(Cities, Population, color = ['red', 'b', 'g', 'black'])
```

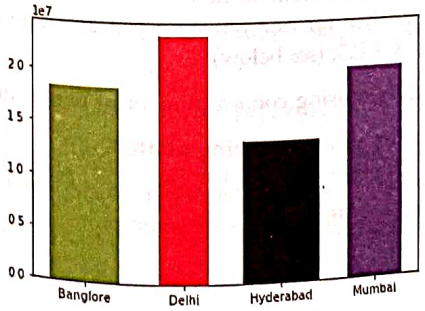
So as per above **bar( )**, the colors *'red', 'b', 'g', 'black'* are applicable to **Cities** values in order *i.e.*, "**Delhi**"'s bar will have 'red' color ; "**Mumbai**"'s bar will have blue color (color code 'b'); "**Bangalore**"'s bar will have 'green' color (color code 'g') ; and "**Hyderabad**"'s bar will have 'black' color.

The output produced by above command will be as shown below :



Both *width* and *color* arguments are optional and you can combine them both also, *e.g.*,

```
plt.bar(Cities, Population, width =[0.5,.6, .7, .8], color = ['red', 'b','g', 'black'])
```

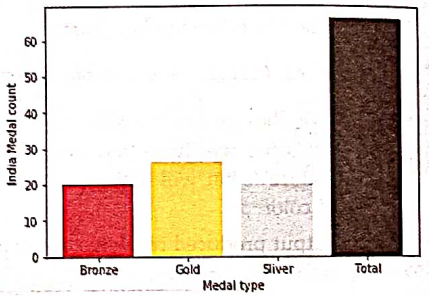


EXAMPLE 14 *Consider the reference table 3.1. Write a program to plot a bar chart from the medals won by India. Make sure that the Gold, Silver, Bronze and Total tally is represented through different colours.*

SOLUTION

```
import matplotlib.pyplot as plt
Info = ['Gold', 'Silver', 'Bronze', 'Total']
India = [26, 20, 20, 66]
plt.bar(Info, India, color = ['gold', 'silver', 'brown', 'black'])
plt.xlabel("Medal type")
plt.ylabel("Medal count")
plt.show()
```

Output



3.4.3 Creating Multiple Bars Chart

Often in real life, you may need to plot multiple data ranges on the same bar chart creating multiple bars. As such PyPlot does not provide a specific function for this, BUT you can always create one exploiting the ***width*** and ***color*** arguments of **bar( )** that you learnt above.

Let's see how.

(*i*) **Decide number of X points.** Firstly, you need to determine how many X points you will need. For this, calculate the number of entries in the ranges being plotted. Say, you are plotting two sequences **A** and **B**, so length of sequences **A** or **B** being plotted will determine the number of X points in our case. You can create a sequence with this length of A or B using ***range( )*** or **numpy.arange( )** functions.

Do note that both the sequences A and B must have similar shape *i.e.*, same length (number of elements).

(*ii*) **Decide thickness of each bar and accordingly adjust X points on X-axis.** So, if you want to plot two bars per X point, then carefully think of the thickness of each bar. Say you want the thickness of each bar as .3 , then for first range keep it as X and for the second data range, make the X point as X + 0.3. (*see* below)

(*iii*) **Give different color to different data ranges** using color argument of bar( ) function.

(*iv*) **The *width* argument remains the same for all ranges being plotted.**

(*v*) **Plot using bar( ) for each range separately,** *i.e.*, the number of bars decide the number of **bar( )** functions you will be using to plot different bars on same plot.

Following example will make it clear.

Say we want to plot ranges $A = [2, 4, 6, 8]$ and $B = [2.8, 3.5, 6.5, 7.7]$

(*i*) Deciding X points and thickness. Say, we want the thickness of each bar as 0.35, then **for the first range, X point will be X,** and **for the second range,** the X will shift by first bar's thickness, *i.e.*, X + 0.35. (If there were three bars, then X points would have been X, X + 0.35 *i.e.*, X + thick of one bar on its left, and X + 0.70 , *i.e.*, X + thickness of two bars on its left.)

(*ii*) **Deciding colors.** Say we want *red* color for the first range and *blue* color for the second range.

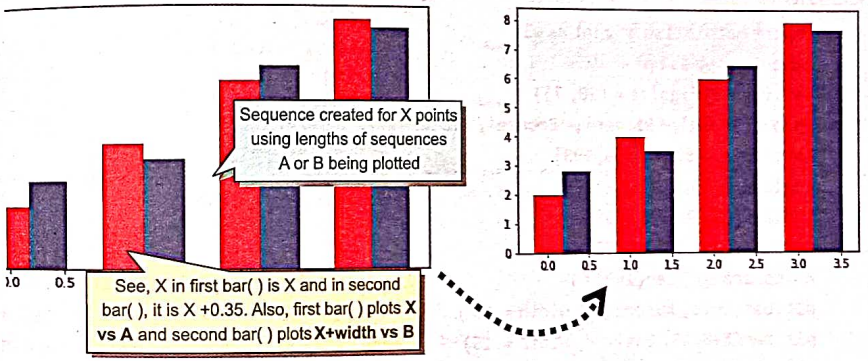
(*iii*) The **width** argument will take value as 0.35 in this case.

(*iv*) Plot using multiple bar( ) functions.

TIP

Add bar width to X coordinate for successive ranges being plotted to make appear bars distinct and not overlapped.

As there are two ranges to be plotted, we shall need to use two bar( ) functions one for plotting X *vs.* A and the second one for plotting X + width *vs.* B, as depicted below :



Following example also creates a multi-bar chart plotting three different data ranges.

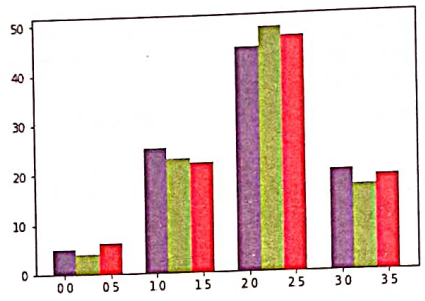
EXAMPLE 15 *Val is a list having three lists inside it. It contains summarised data of three different trials conducted by company A. Create a bar chart that plots these three sublists of Val in a single chart. Keep the width of each bar as 0.25.*

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
Val = [[5., 25., 45., 20.], [4., 23., 49., 17.], [6., 22., 47., 19.]]
X = np.arange(4)
plt.bar(X + 0.00, Val[0], color = 'b', width = 0.25)
plt.bar(X + 0.25, Val[1], color = 'g', width = 0.25)
plt.bar(X + 0.50, Val[2], color = 'r', width = 0.25)
plt.show()
```

See, the X range for the third bar(), is deviated with X+ widths of two bars on the left of it. i.e., X+0.50 as width of one bar is 0.25

The output produced by above code is :

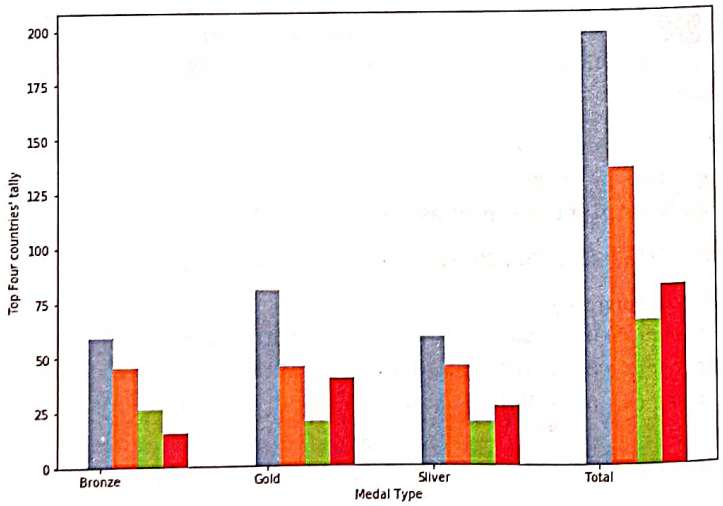


EXAMPLE **16** *Consider the reference table 3.1. Write a program to plot a bar chart from the medals won by top four countries. Make sure that bars are separately visible.*

SOLUTION

```
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize = (10, 7))
Info = ['Gold', 'Silver', 'Bronze', 'Total']
Australia =[80, 59, 59,198]
England = [45, 45, 46, 136 ]
India = [26, 20, 20, 66]
Canada = [15, 40, 27, 82]
X = np.arange( len(Info) )
plt.bar(Info, Australia, width =.15 )
plt.bar(X+0.15, England, width =.15)
plt.bar(X+0.30, India, width =.15)
plt.bar(X+0.45, Canada, width =.15)
plt.show()
```

Output

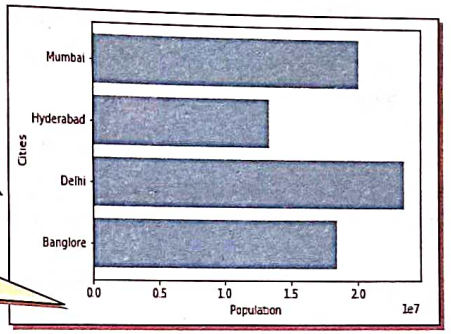


3.4.4 Creating a Horizontal Bar Chart

To create a horizontal bar chart, you need to use **barh( )** function (bar horizontal), in place of bar( ). Also, you need to give *x* and *y* axis labels carefully – the label that you gave to *x* axis in bar( ), will become *y*-axis' label in barh( ) and vice-versa.

For instance, consider the following code where we have replaced bar( ) with barh( ) in an earlier code :

```
In [2]: Cities =['Delhi', 'Mumbai', 'Banglore', 'Hyderabad']
...: Population= [23456123, 20083104, 18456123, 13411093]
...: plt.barh(Cities, Population)
...: plt.ylabel('Cities')
...: plt.xlabel('Population')
...: plt.show()
```



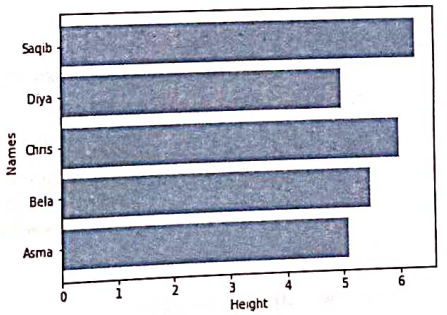
Also, notice that the labels of x and y axis have been swapped for **barh()** when compared to **bar()**

EXAMPLE **17** *Write a program to plot a horizontal bar chart from the heights of some students.*

SOLUTION

```
import matplotlib.pyplot as plt
height = [5.1, 5.5, 6.0, 5.0, 6.3]
Names = ('Asma', 'Bela', 'Chris', 'Diya', 'Saqib')
# Create horizontal bars
plt.barh(Names, height)
plt.xlabel("Height")
plt.ylabel("Names")
plt.show()
```

Output



3.4.5 Creating Pie Charts

Recall that a pie chart (or a circle chart) is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. Typically, a Pie Chart is used to show parts to the whole, and often a % share. The PyPlot interface offers **pie( )** function for creating a pie chart.

PIE CHARTS

"The **pie chart** is a type of graph in which a circle is divided into sectors that each represent a proportion of the whole."

Before we proceed with **pie( )** function's examples, it is important to know two things :

(*i*) The pie( ) function, **plots a single data range only**. It will calculate the share of individual elements of the data range being plotted *vs.* the whole of the data range.

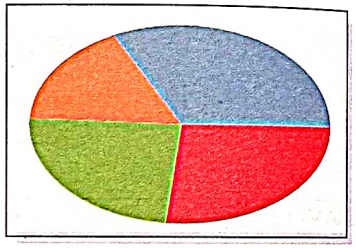
(*ii*) The default shape of a pie chart is oval but you can always change to circle by using **axis( )** of pyplot, sending "equal" as argument to it. That is, issue following command before you use **pie( )** function :

```
matplotlib.pyplot.axis("equal")
```
← *In place of **matplotlib.pyplot**, you may also use its alias name, the name you have given to it while importing*

Now consider following example code to understand working of pie( ).

Your school houses have collected amount for PM charity fund. To plot a pie chart from this data, you can do the following :

```
contri = [17, 8.8, 12.75, 14]          # contribution in thousands
plt.pie(contri)
```
← *The data range being plotted in pie( )*



The output produced by above code will be :

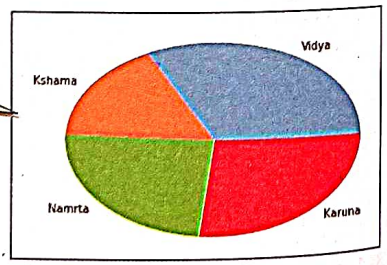
The adjacent chart shows the values of **contri** list as slices of a pie.

3.4.5A Labels of Slices of Pie

The above shown pie chart is just incomplete. You cannot make out which slice belongs to what. So, there must be labels next to the slices to make more readable and understandable. For this purpose, you need to create a sequence containing the labels and then specify this sequence as value for labels argument of pie() function. The first label is given to first value; second label to second value and so on, *e.g.*, see below :

```
contri = [17, 8.8, 12.75, 14]          # contribution in thousands
houses = ['Vidya', 'Kshama', 'Namrta', 'Karuna']
plt.pie(contri, labels = houses)
```

Now the labels for the slices will be taken from the sequence namely houses.



Now the output looks like :

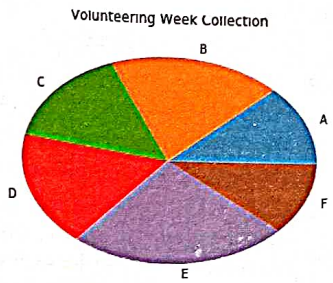
EXAMPLE 18 *TSS school celebrated volunteering week where each section of class XI dedicated a day for collecting amount for charity being supported by the school. Section A volunteered on Monday, B on Tuesday, C on Wednesday and so on. There are six sections in class XI. Amounts collected by sections A to F are 8000, 12000, 9800, 11200, 15500, 7300.*

Write a program to create a pie chart showing collection amount section wise.

SOLUTION

```
import matplotlib.pyplot as plt
Col = [8000, 12000, 9800, 11200, 15500, 7300]
Section = ['A', 'B', 'C', 'D', 'E', 'F']
plt.title("Volunteering Week Collection")
plt.pie(Col, labels = Section )
plt.show()
```

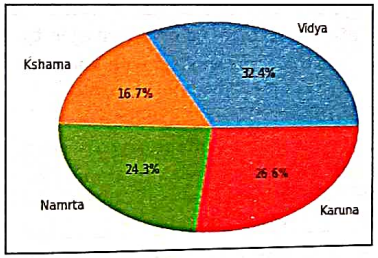
Output



3.4.5B Adding Formatted Slice Percentages to Pie

To view percentage of share in a pie chart, you need to add an argument **autopct** with a format string, such as **"%1.1F%%"**. It will show the percent share of each slice to the whole, formatted in a certain way, *e.g.*, see below :

```
plt.pie(contri, labels = houses, autopct ="%1.1f%%")
```



The percentage of each value plotted is calculated as : *single value/(sum of all values) * 100* and this also determines the size of the slice, *i.e.*, more the value bigger the slice.

For instance, if a list being plotted contains values as [30, 50, 10, 6], then

(*i*) First of internally sum of all these is calculated which is 30 + 50 + 10 + 6 = **96**.

(*ii*) Next each value's percentage share and its slice size is calculated as :

Slice 1 : 30/96 = 31.25%
Slice 2 : 50/96 = 52.08%
Slice 3 : 10/96 = 10.42%
Slice 4 : 6/96 = 6.25%

The format string used with **autopct** will determine the format of the percentage being displayed. The format string begins with the "%" operator, which specifies the format of the percentage value being displayed. The general syntax for a format placeholder is

```
"%[flags][width][.precision]type"
```

- The first % symbol is a special character that signifies that it is a special string which will determine the format of the values to be displayed.

- The **width** determines the total number of characters to be displayed (digits before and after decimal point + 1 for decimal point). If the value being displayed has lesser number of digits than the width specified, then the value is padded as per the ***flag*** specified. If however, the value has more digits than the ***width*** specifies, then the full value is displayed. The examples below will make it clear.
- In pie charts, the most useful **flag** is 0 (zero), which when specified will pad the value being displayed with preceding zeros if the digits of value is less the ***width***.
- The ***precision*** is specified with digits and it specifies the number of digits after decimal point.
- The ***type*** specifies the type of value : *d* or *i* means integer type and *f* or *F* means float type.
- To print a % sign, given %% (**two percentage signs**) in the format string. The two %% signs are needed to print a percentage sign, otherwise single % is interpreted as a special character with a special meaning and thus used accordingly. By doubling %, *i.e.*, %%, you suppress its special meaning and thus a percentage sign is printed.

Now consider following examples of format strings as given below.

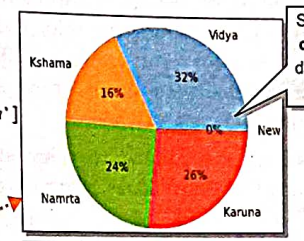
<table>
<tr><th>Format string</th><th>Description</th></tr>
<tr><td>"%5d" , "%5i"</td><td>width = 5, type = d or i (integer type)
Print the value with 5 characters e.g., if the value being printed is 123 then it will be printed as __123, because 123 consists only of 3 digits, the output is padded with 2 leading blanks.</td></tr>
<tr><td>"%05d", "%05i"</td><td>flag = 0, width = 5, type = d or i (integer type i.e., no decimal point)
Print the value with 5 characters and pad with leading zeros if necessary. E.g., if the value being printed is 123, then it will be printed as 00123, because value 123 consists only of 3 digits, the output is padded with 2 leading zeros to make it have width of 5 digits.</td></tr>
<tr><td>"%03d %%", "%03i %%"</td><td>width = 3, type = d or i (integer type), percentage sign in the end
Print the value with 3 characters, pad with leading zeros if necessary and end with a % sign. E.g., if the value being printed is 123 then it will be printed as 123 % ; no leading zeros as the width 3 is fully filled.</td></tr>
<tr><td>"%6.2f", "%6.2F"</td><td>width = 6, precision = 2, type = f (float type)
Print the value with space of 6 characters; after decimal point, keep the precision of 2 digits (rounding off may take place for this precision); pad with leading blank if necessary. Consider following examples :
<table>
<tr><th>Value</th><th>Formatted As</th><th>Remarks</th></tr>
<tr><td>12.679</td><td>_12.68</td><td>Rounded off for 2 precise digits ; 1 leading blank</td></tr>
<tr><td>0.018</td><td>_ _0.02</td><td>Rounded off for 2 precise digits ; 2 leading blanks</td></tr>
<tr><td>211.2</td><td>211.20</td><td>1 trailing zero added for 2 precise digits.</td></tr>
<tr><td>7</td><td>_ _7.00</td><td>2 trailing zeros added for 2 precise digits ; 2 leading blanks.</td></tr>
<tr><td>1009.27</td><td>1009.27</td><td>This time 7 characters will get printed (more than the width) as the number before the decimal point cannot be reduced and also, precision digits cannot be reduced as .27 is already precise to 2 digits.</td></tr>
<tr><td>1009.277</td><td>1009.28</td><td>7 characters will get printed as the number before the decimal point cannot be reduced and for 2 precision digits, 0.277 is rounded off as .28.</td></tr>
</table></td></tr>
<tr><td>"%6.2f%%", "%6.2F%%"</td><td>Same as previous but with a percentage sign in the end.</td></tr>
</table>

Now consider some examples below where we added one new value each to lists **contri** and **houses** used above.

```
In [30]: contri
Out[30]: [17, 8.8, 12.75, 14, 0.1]

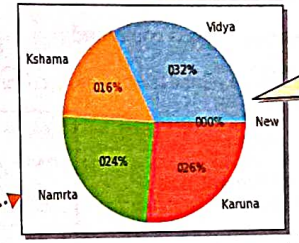
In [31]: houses
Out[31]: ['Vidya', 'Kshama', 'Namrta', 'Karuna', 'New']
```

```
plt.axis("equal")     # for circle shape
plt.pie(contri, labels = houses, autopct ="%3d%%" )
```



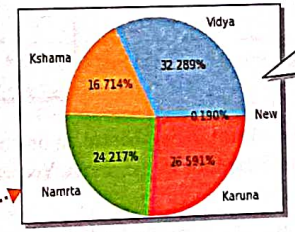
See, 'New' house's **contri value 0.1** is depicted as 0% in **d** (integer format)

```
plt.axis("equal")     # for circle shape
plt.pie(contri, labels = houses, autopct ="%03d%%")
```



See, leading zeros have been padded this time to make it 3 characters wide

```
plt.axis("equal")     # for circle shape
plt.pie(contri, labels = houses, autopct ="%05.3f%%")
```



See, trailing zeros have been added this time to make it 3 digits precise

To see Pie charts in action

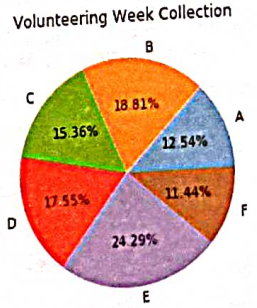
Scan QR Code

EXAMPLE **19** *Considering the TSS school's charity collection (as given in previous example). Write a program to create a pie chart showing collection amount percentage (%) section wise. Also make sure that the pie chart is perfectly circular.*

SOLUTION

```
import matplotlib.pyplot as plt
Col = [8000, 12000, 9800, 11200, 15500, 7300]
Section = ['A', 'B', 'C', 'D', 'E', 'F' ]
plt.title("Volunteering Week Collection")
plt.axis("equal")
plt.pie(Col, labels = Section, autopct = "%5.2f%%")
plt.show()
```

Output



3.4.5C Changing Colors of the Slices

By default, the **pie( )** function shows each slice with a different colour. If you are not satisfied with the default colors, you can specify own colours for the slices. For this purpose, you need to create a sequence containing the color codes or names for each slice and then specify this sequence as a value for **colors** argument of **pie( )** function. The first color is given to first value; second color to second value and so on, *e.g.*, see below (considering the same **contri** and **houses** lists containing five values each).

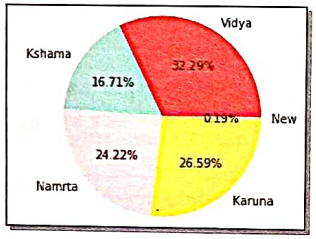
We need a sequence containing five colors for five values of list **contri**, so we created a sequence **colr** containing five color names (you may also give color codes such as 'r', 'b', 'g', etc.)

```
colr = ['red', 'cyan', 'pink', 'yellow', 'silver']
```

Now we issued followed command :

```
plt.pie(contri, labels = houses, colors = colr, autopct ="%2.2f%%" )
```

And the result we got was :



See, the colors to each slices have been given from **colr** sequence in order (1st color to 1st value,, 2nd color to 2nd value and so on)

3.4.5D Exploding a Slice

Sometimes you want to emphasize on one or more slices and show them little pulled out. This feature is called **explode** in pie charts. You can provide explode values for the slices in the form of a sequence, *e.g.*,

- if you want to **explode first slice** out of five slices being plotted with a distance of 0.2 units then your sequence for explode will be like : [0.2, 0, 0, 0,0]
- To **explode 3rd and 5th** slices by a distance of 0.15 and 0.3 units out of five slices, the explode sequence will be [0, 0, 0.15, 0, 0.3]

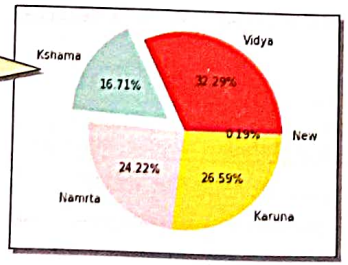
You need to specify this explode sequence as value to explode argument of **pie( )** function :

```
expl = [0, 0.2, 0, 0, 0]
plt.pie(contri, labels = houses, explode = expl, colors = colr, autopct ="%2.2f%%" )
```

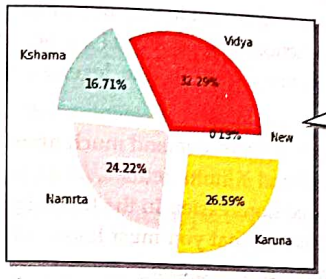
```
In [46]: expl = [0, 0.2, 0, 0, 0]

In [47]: plt.axis("equal")
    ...: plt.pie(contri, labels = houses, explode = expl, colors = colr, autopct ="%2.2f%%" )
    ...:
Out[47]:
```

See, the second slice is exploded with a distance of 0.2 units



If you change the **expl** sequence to [0, 0.2, 0, 0.3, 0] where there are explode values for 2nd and 4th slices, the pie chart will look like :



This time both 2nd and 4th slices are exploded as they both have their explode values

NOTE

PyPlot provides a procedural interface to the matplotlib object-oriented plotting library. It is modeled closely after Matlab™.

Thus, we can summarize that the complete form of **pie( )** function is as shown below :

```
<matplotlib.pyplot>.pie(data [, labels = <labels sequence>] [, explode = <explode sequence>] [, colors = <color sequence>] [, autopct = <format string> ])
```

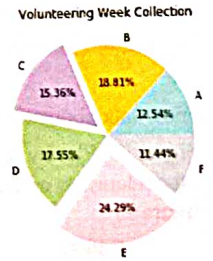
EXAMPLE **20** *Considering the TSS school's charity collection (as given in previous example). Write a program to create a pie chart showing collection amount percentage (%) section wise.*

- *Make sure that the pie chart is circular.*
- *Show the collection of sections C and E with exploded pies.*

SOLUTION

```
import matplotlib.pyplot as plt
Col = [8000, 12000, 9800, 11200, 15500, 7300]
Section = ['A', 'B', 'C', 'D', 'E', 'F' ]
expl= [ 0, 0, 0.15, 0, 0.2, 0]
col = [ 'cyan', 'gold', 'violet', 'lightgreen', 'pink', 'silver']
plt.title("Volunteering Week Collection")
plt.axis("equal")
plt.pie(Col, labels = Section, explode = expl, colors = col, autopct = "%5.2f%%")
plt.show()
```

Output



3.5 Customizing the Plot

Till now you have learnt to create line charts, bar charts and pie charts using pyplot interface. But here, let me ask you a question – are you satisfied with the outcome or the plots that you created using **plot( )**, **bar( )** and **pie( )** functions? Hmm, even I am not satisfied. These plots do not show any details like what a chart is showing *i.e.*, its title, legends, X and Y axes' details etc. are not shown and so on.

As mentioned earlier that '*Data-visualization-needs*' demand much more from a graph/plot. The graph or plot should have a proper *title*, X and Y limits defined, *labels*, *legends* etc. All this makes understanding the plot and taking the decisions easier. In the following lines, we are covering how you can customize your plots but before that you must know what all makes a plot and what all you can customize, *i.e.*, the anatomy of a chart.

3.5.1 Anatomy of a Chart

Any graph or chart that you create using **matplotlib's PyPlot** interface, is created as per a specific structure of a plot or shall we say a *specific anatomy*. As ***anatomy***, generally refers to study of bodily structure (or parts) of something, here we shall discuss various parts of a plot that you can create using PyPlot.

PyPlot charts have hierarchical structures or in simple words they are actually like containers containing multiple items/things inside it. Look at the figure given below carefully and then go through the terms given below it describe it.

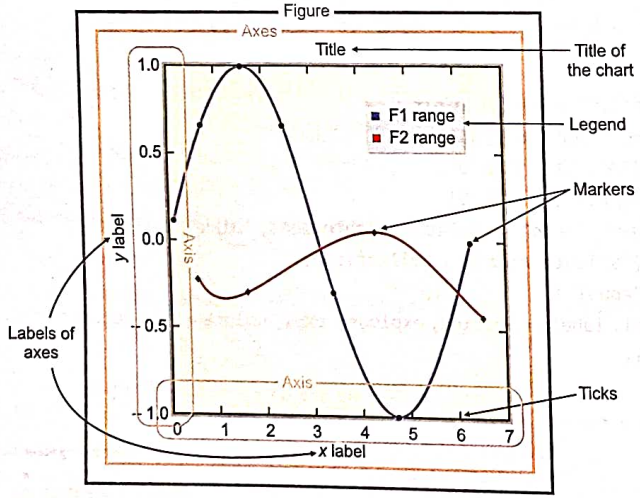


Figure 3.2 Anatomy of a Chart.

Let us now talk about various parts of a chart as shown in Fig. 3.2 above.

- **Figure.** PyPlot by default plots every chart into an area called *Figure*. A figure contains other elements of the plot in it. You can set the size of the plot using **figsize** argument of figure as :

```
<pyplot>.figure (figsize = (<width>, <length>)
```


- **Axes.** The axes define the area (mostly rectangular in shape for simple plots) on which actual plot (line or bar or graph etc.) will appear. Axes have properties like ***label, limits*** and ***tick marks*** on them.

  There are two axes in a plot : (*i*) **X–axis**, the horizontal axis , (*ii*) **Y-axis**, the vertical axis.
  - **Axis label.** It defines the name for an axis. It is individually defined for X-axis and Y-axis each.
  - **Limits.** These define the range of values and number of values marked on X-axis and Y-axis.
  - **Tick_Marks.** The tick marks are individual points marked on the X-axis or Y-axis.
- **Title.** This is the text that appears on the top of the plot. It defines what the chart is about.
- **Legends.** These are the different colors that identify different sets of data plotted on the plot. The legends are shown in a corner of the plot.

Now that you know what components a plot can have, let us learn how you can customize these and create informative, user friendly charts using PyPlot.

3.5.2 Adding a Title

To see Customizing charts in action

Scan QR Code

To add a title to your plot, you need to call function title( ) before you show your plot. The syntax of title( ) function is as :

```
<matplotlib.pyplot>.title(<title string>)
```

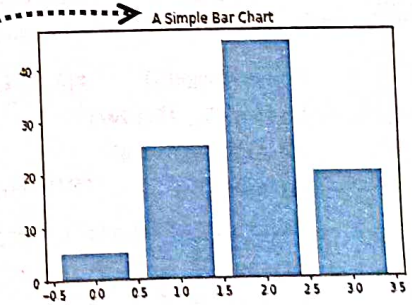
e.g.,

```
plt.title( "A Bar chart")
```

will add a title string "A Bar Chart" to your plot at the top, outside the axes area (assuming that the *matplotlib.pyplot* has been imported with alias name as **plt**).

Consider the following example :

```
X = np.arange(4)        #[0, 1, 2, 3]
Y = [5., 25., 45., 20.]
plt.bar(X, Y)
plt.title("A Simple Bar Chart")
plt.show()
```



You can use **title( )** function for all types of plots *i.e.*, for *plot( )*, for *bar( )*, for *pie( )*, and so on.

3.5.3 Setting X and Y Labels, Limits and Ticks

You already know about setting labels for X- and Y-axes as we have done it earlier. Recall that functions **xlabel( )** and **ylabel( )** can be used to set labels for X and Y-axes respectively. Thus, we shall not cover them here again, but we shall talk about setting of *limits* and *ticks*.

3.5.3A Setting Xlimits and Ylimits

When you specify and X and Y ranges for plotting, PyPlot automatically tries to find best fitting range for X-axis and Y-axis depending on the data being plotted. But sometimes, you may need to have own limits specified for X and Y axes. For this, you can use **xlim( )** and **ylim( )** functions to set limits for *X*-axis and *Y*-axis respectively. Make sure to set the limits keeping in mind the data being plotted.

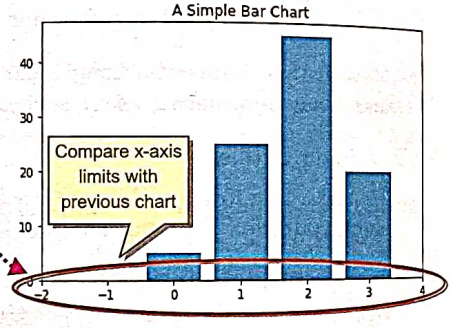
Both **xlim( )** and **ylim( )** are used as per following format :

```
<matplotlib.pyplot>.xlim(<xmin>, <xmax>)      # set the X-axis limits as xmin to xmax
<matplotlib.pyplot>.ylim(<ymin>, <ymax>)      # set the Y-axis limits as ymin to ymax
```

Let us reconsider the above example where we added a title to a bar chart in section 3.5.2 ; let us change the Xlimits for the above chart :

```
X = np.arange(4)
Y = [5., 25., 45., 20.]
plt.xlim(-2.0, 4.0)
plt.bar(X, Y)
plt.title("A Simple Bar Chart")
plt.show()
```

X-axis limits changed

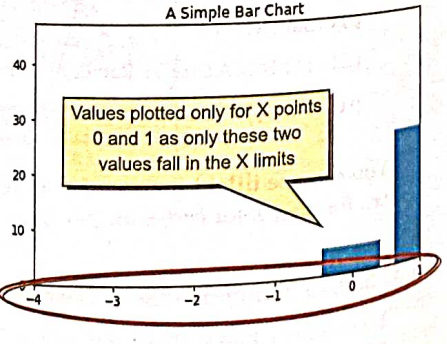


Important

While setting up the limits for axes, you must keep in mind that only the data that falls into the limits of *X* and *Y*-axes will be plotted; rest of the data will not show in the plot. In other words, if you have set *X*-axis or *Y*-axis limits which are not compatible with the data values being plotted, you may either get incomplete plot or the data being plotted is not visible at all.

Let us reconsider the previous chart and change the *Xlimits* – carefully go through the code below and its implications :

```
X = np.arange(4)        #[0, 1, 2, 3]
Y = [5., 25., 45., 20.]
plt.xlim(-4.0, 1.0)
plt.bar(X, Y)
plt.title("A Simple Bar Chart")
plt.show()
```



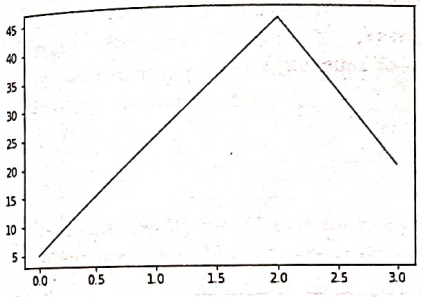
NOTE

Only the data values mapping on xlimits and ylimits will get plotted. If no data values maps to the xlimits or ylimits, nothing will show on the plot.

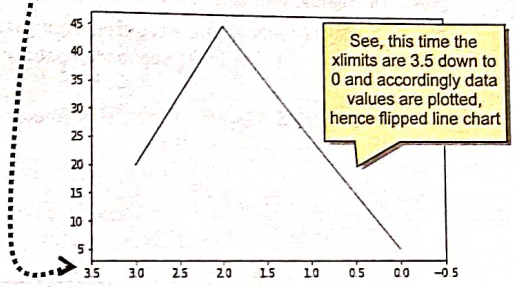
Interesting

You can use decreasing axes by flipping the normal order of the axis limits *i.e.*, if you swap the limits *(min, max)* as *(max, min)*, then the plot gets flipped, *e.g.*, see below :

```
In [11]: X = [0, 1, 2, 3]
   ...: Y = [5., 25., 45., 20.]
   ...: plt.plot(X, Y)
   ...: plt.show()
   ...:
```



```
In [15]: X = [0, 1, 2, 3]
   ...: Y = [5., 25., 45., 20.]
   ...: plt.xlim(3.5, -0.5)
   ...: plt.plot(X, Y)
   ...: plt.show()
   ...:
```



The *Ylim( )* function works just the same as *Xlim( )*, only thing is that it works on *Y*-axis.

NOTE

If you do not specify *x* or *y* limits, PyPlot will automatically decide the limits for *X* and *Y* axes as per the values being plotted.

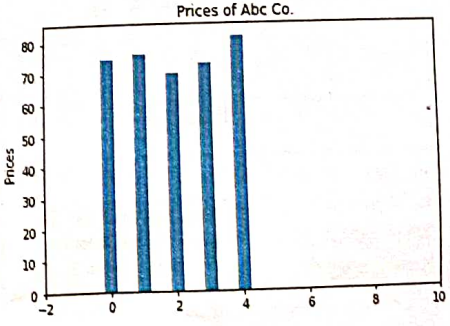
EXAMPLE **21** *Generally ten different prices of a stock are stored. However, for ABC Co. only 5 prices are available for a day* : [74.25, 76.06, 69.5, 72.55, 81.5]

Write a program to create a bar chart with the given prices but the graph should be plotted between the limits –2 to 10 on x-axis.

SOLUTION

```
import matplotlib.pyplot as plt
pr = [74.25, 76.06, 69.5, 72.55, 81.5]
plt.bar(range(len(pr)), pr, width = 0.4)
plt.xlim(-2, 10)
plt.title("Prices of Abc Co.")
plt.ylabel("Prices")
plt.show()
```

Output



3.5.3B Setting Ticks for Axes

By default, PyPlot will automatically decide which data points will have ticks on the axes, but you can also decide which data points will have tick marks on *X*- and *Y*-axes.

To set own tick marks :

- for X-axis, you can use **xticks( )** function as per format :

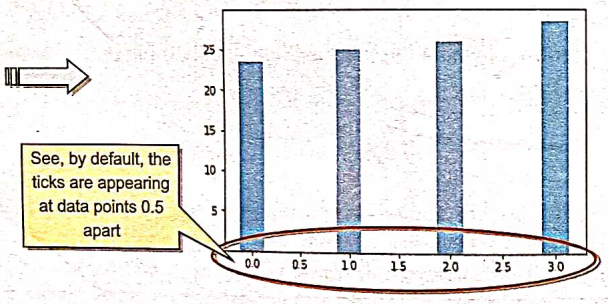
```
xticks(<sequence containing tick data points>,
       [<Optional sequence containing tick labels>])
```

- for Y-axis, you can use **yticks( )** function as per format :

```
yticks(<sequence containing tick data points>,
       [<Optional sequence containing tick labels>])
```

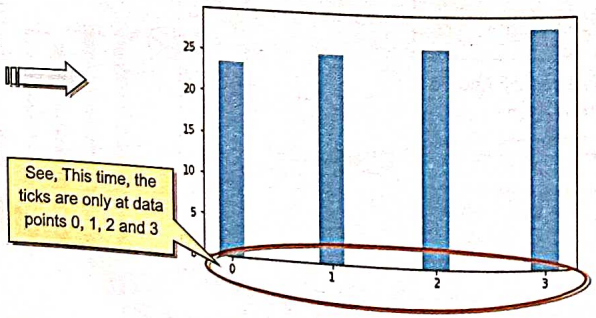
Let us understand this with the help of one example :

```
q = range(4)
s = [23.5, 25, 26, 28.5]
plt.bar(q, s, width = 0.25)
```



If you want that tick marks should appear only at data points 0, 1, 2 and 3, you will need to give code as :

```
q = range(4)
s = [23.5, 25, 26, 28.5]
plt.xticks([0,1, 2, 3])
plt.bar(q, s, width = 0.25)
```

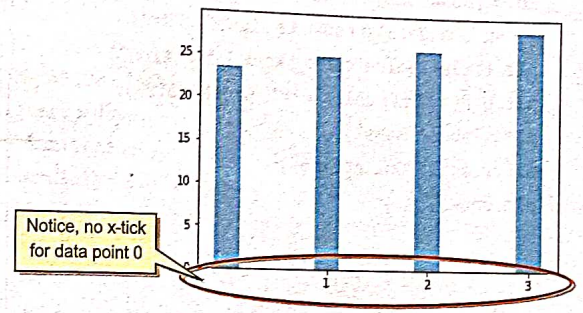


Now carefully go through the following code (it is very similar to the code above) :

```
q = range(4)
s = [23.5, 25, 26, 28.5]
plt.xticks([1, 2, 3, 4])
plt.bar(q,s, width = 0.25)
```

Can you figure out how the resultant plot will look like ?

Hmm. Well, you guessed it right – this time *X*-ticks will appear at data points 1, 2, 3.. but not on data point 0, *i.e.*, as :



You can specify own labels for tick points too by specifying an additional sequence containing labels for ticks. To understand this, consider following example.

EXAMPLE **22** *Generally ten different prices of a stock are stored. However, for ABC Co. only 5 prices are available for a day :* [74.25, 76.06, 69.5, 72.55, 81.5]

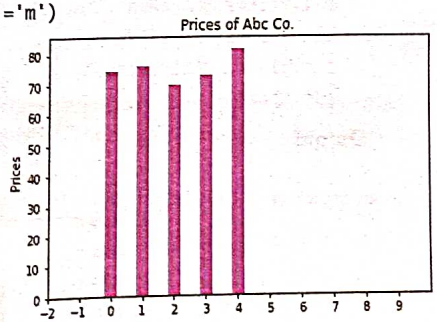
Write a program to create a bar chart with the given prices :

- *the graph should be plotted between the limits –2 to 10 on x-axis.*
- *there should be tick for every plotted point.*

SOLUTION

```
import matplotlib.pyplot as plt
pr = [74.25, 76.06, 69.5, 72.55, 81.5]
plt.bar(range(len(pr)), pr, width = 0.4, color ='m')
plt.xlim(-2, 10)
plt.title("Prices of Abc Co.")
plt.xticks(range(-2, 10))
plt.ylabel("Prices")
plt.show()
```

Output



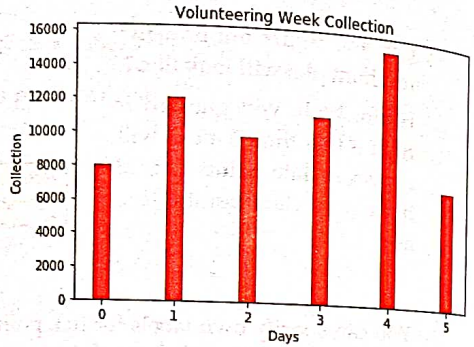
EXAMPLE **23** *TSS school celebrated volunteering week where each section of class XI dedicated a day for collecting amount for charity being supported by the school. Section A volunteered on Monday, B on Tuesday, C on Wednesday and so on. There are six sections in class XI. Amounts collected by sections A to F are 8000, 12000, 9800, 11200, 15500, 7300.*

Write a program to create a bar chart showing collection amount. The graph should have proper title and axes titles.

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
Col = [8000, 12000, 9800, 11200, 15500, 7300]
X = np.arange(6) # range to signify 6 days
plt.title("Volunteering Week Collection")
plt.bar(X, Col, color = 'r', width = 0.25)
plt.xlabel("Days")
plt.ylabel("Collection")
plt.show()
```

Output

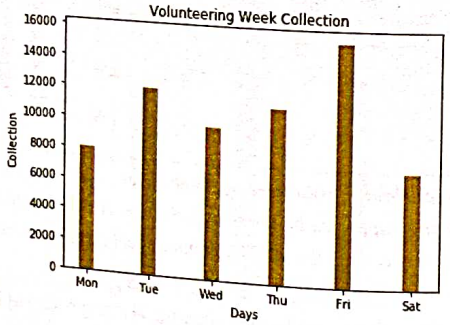


EXAMPLE **24** *Considering the TSS school collection for charity of previous example. Write a program to plot the collected* ***amount*** *vs.* ***days*** *using a bar chart. The* ***ticks*** *on X-axis should have* ***Day names****. The graph should have proper title and axes titles.*

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
Col = [8000, 12000, 9800, 11200, 15500, 7300]
X = np.arange(6) # range to signify 6 days
plt.title("Volunteering Week Collection")
plt.bar(X , Col, color = 'olive', width = 0.25)
plt.xticks( X, ['Mon','Tue','Wed','Thu','Fri','Sat'])
plt.xlabel("Days")
plt.ylabel("Collection")
plt.show()
```

Output

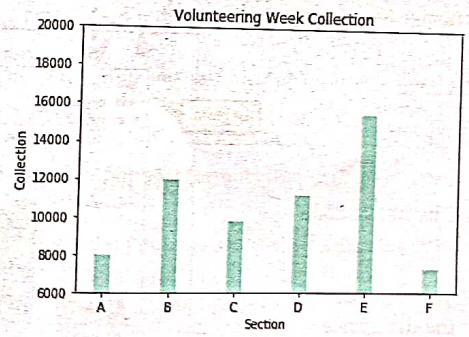


EXAMPLE **25** *Considering the TSS school collection for charity of previous example, write a program to plot the collected* ***amount*** *vs.* ***sections*** *using a bar chart. The* ***ticks*** *on X-axis should have* ***section names****. The graph should have proper title and axes titles. Make sure that the limits for y-axis are in the range 6000 to 20000.*

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
Col = [8000, 12000, 9800, 11200, 15500, 7300]
X = np.arange(6) # range to signify 6 days
plt.title("Volunteering Week Collection")
plt.bar(X , Col, color = 'cyan', width = 0.25)
plt.xticks(X, ['A', 'B', 'C', 'D', 'E', 'F'])
plt.ylim(6000, 20000)
plt.xlabel("Section")
plt.ylabel("Collection")
plt.show()
```

Output



3.5.4 Adding Legends

When we plot multiple ranges on a single plot, it becomes necessary that legends are specified. Recall that a legend is a color or mark linked to a specific data range plotted. To plot a legend, you need to do two things :

(*i*) In the plotting functions like plot( ), bar( ) etc., give a specific label to data range using argument **label**.

(*ii*) Add legend to the plot using legend( ) as per format :

```
<matplotlib.pyplot>.legend(loc = <position number or string>)
```

The **loc** argument can either take values 1, 2, 3, 4 signifying the position strings *'upper right', 'upper left', 'lower left', 'lower right'* respectively. Default position is *'upper right'* or 1.

Let us understand this process with an example. Let us reconsider example 15, where we plotted three sublists in a bar chart. Let us add legends to this graph.

Code :

```
import numpy as np
import matplotlib.pyplot as plt

Val = [[5., 25., 45., 20.], [4., 23., 49., 17.], [6., 22., 47., 19.]]
X = np.arange(4)
```

#Step1 : specify label for each range being plotted using label argument, *i.e.,* :

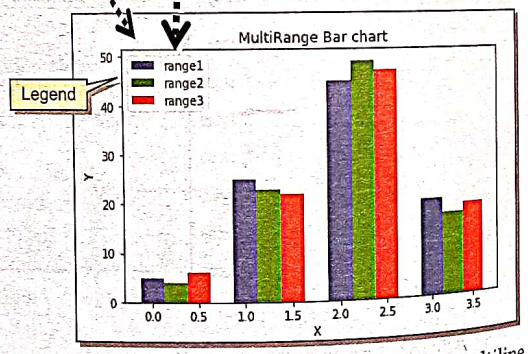
```
plt.bar(X + 0.00, Val[0], color = 'b', width = 0.25, label = 'range1')
plt.bar(X + 0.25, Val[1], color = 'g', width = 0.25, label = 'range2')
plt.bar(X + 0.50, Val[2], color = 'r', width = 0.25, label = 'range3')
```

Step2 : add legend, *i.e.,* :

```
plt.legend(loc = 'upper left')
```

additional and optional chart formatting :

```
plt.title("MultiRange Bar chart")
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



The output produced by above code will be :

Compare it with the output of example 15 and see the difference.

You can use ***label*** argument and ***legend( )*** function with *plot( )* function too for multiline charts.

Consider the following example.

EXAMPLE **26** *Create multiple line charts on common plot where three data ranges are plotted on same chart. The data range(s) to be plotted is/are :*

```
Data = [[5., 25., 45., 20.], [8., 13., 29., 27.], [9., 29., 27., 39.]]
```

SOLUTION

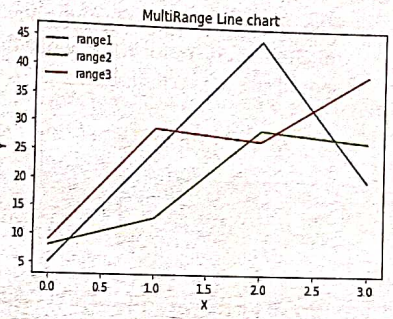
```
import numpy as np
import matplotlib.pyplot as plt

Data = [[5., 25., 45., 20.], [8., 13., 29., 27.], [9., 29., 27., 39.]]
X = np.arange(4)
```



```
plt.plot(X , Data[0], color = 'b', label = 'range1')
plt.plot(X, Data[1], color = 'g', label = 'range2')
plt.plot(X, Data[2], color = 'r', label = 'range3')
plt.legend(loc = 'upper left')
plt.title("MultiRange Line chart")
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

Output



NOTE

You can plot using any type of sequence (*e.g., lists/tuples*), arrays (*e.g., ndarrays*) or even individual rows/ columns of dataframe as data. Solved problem 15 uses a dataframe columns to create a scatter chart.

3.5.5 Saving a Figure

If you want to save a plot created using pyplot functions for later use or for keeping records, you can use **savefig( )** to save the plot.

You can use the pyplot's savefig( ) as per format :

```
<matplotlib.pyplot>.savefig(<string with filename and path>)
```

- You can save figures in popular formats like *.pdf, .png, .eps* etc. Specify the format as file extension.
- Also, while specifying the path, use double slashes to suppress special meaning of single slash character.

Consider the following examples :

```
plt.savefig("multibar.pdf")                  # save the plot in current directory
plt.savefig("c:\\data\\multibar.pdf")        # save the plot at the given path
plt.savefig("c:\\data\\multibar.png")        # save the plot at the given path in png format
```

You can open the folder to check yourself. After running the code, your plot will be saved there in the specified file format.

Important

Please note that we have kept our discussion of pyplot functions very simple. Pyplot functions are capable of doing much more. Covering complete functionality of these functions is way beyond the scope of the book. However, if you want to learn and try more, then you may refer to matplotlib documentation.

3.6 Creating Histograms with PyPlot

A histogram is a summarisation tool for discrete or continuous data. A histogram provides a visual interpretation of numerical data by showing the number of data points that fall within a specified range of values (called *bins*). It is similar to a vertical bar graph. However, a histogram, unlike a vertical bar graph, shows no gaps between the bars (see Fig. 3.3)

Histograms are a great way to show results of ***continuous data***, such as : weight, height, how much time, and so forth. But when the data is in categories (such as *Country* or *Subject* etc.), one should use a bar chart.

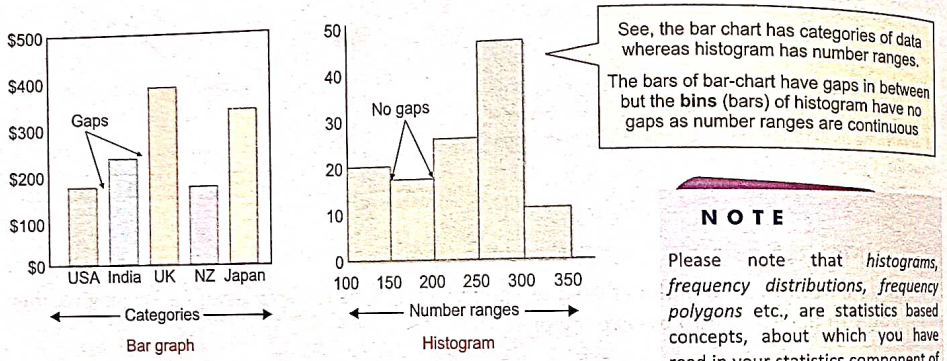


Figure 3.3 Bar graph *vs* histograms.

NOTE

Please note that *histograms, frequency distributions, frequency polygons* etc., are statistics based concepts, about which you have read in your statistics component of Economics. Therefore, we are not discussing the statistical explanations of these concepts here, instead we are just focussing on how these can be plotted using PyPlot.

Let us now talk about how you can create histograms in Python.

Histogram using hist( ) Function

The hist( ) of PyPlot module lets you create and plot histogram from a give sequence(s) of numbers. The syntax for using hist( ) function of plyplot is :

```
matplotlib.pyplot.hist(x, bins = None, cumulative = False, histtype = 'bar',
                       align = 'mid', orientation = 'vertical', )
```

Parameters :

x (n,) array or sequence of (n,) arrays to be plotted on histogram.

bins int[3], optional. If an integer is given, bins + 1 bin-edges are calculated and returned. Default value is automatically provided internally[4].

cumulative bool, optional; If *True*, then a histogram is computed where each bin gives the counts in that bin plus all bins for smaller values. The last bin gives the total number of datapoints. Default is *False*.

histtype {*'bar', 'barstacked', 'step', 'stepfilled'*}, optional ; the type of histogram to draw.

'bar' is a traditional bar-type histogram. If multiple data are given the bars are arranged side by side.

3. bins can take a sequence too. For complete syntax and working refer to pyplot documentation.
4. taken from **rcParams** dictionary of **matplotlib** that contains default values for various parameters.

'barstacked' is a bar-type histogram where multiple data are stacked on top of each other.

'step' generates a *lineplot* that is by default unfilled.

'stepfilled' generates a *lineplot* that is by default filled.

Default is 'bar' type of histtype

orientation {'horizontal', 'vertical'}, optional ; If 'horizontal', barh will be used for bar-type histograms.

Have a look at following examples that plot histograms from two *ndarrays* **x** and **y** each having randomly generated numbers (more than 100 number ; both arrays shown below).

```
x = array([-0.04773042,  -0.54508323, 0.85572137, ..., 0.44027371, -0.26309049, -0.87732363])
y = array([-0.02914181, -0.33280057, 0.52246068, ..., 0.26880911, -0.1606299, -0.53564994])
```

Before you plot, make sure to import the required library/ modules, *e.g.*, we have given following import statements in the beginning :

```
import matplotlib.pyplot as pl
import numpy as np
```

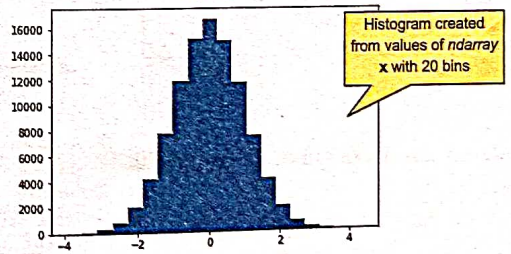
NOTE

The **pyplot** is a module in the **matplotlib** data visualization package of Python. That is why you always import it as **matplotlib.pyplot.**

1. **Plot a histogram from an ndarray x with 20 bins**

   To specify bins, the **bins** argument of **hist( )** is used.

```
pl.hist(x, bins = 20)
```

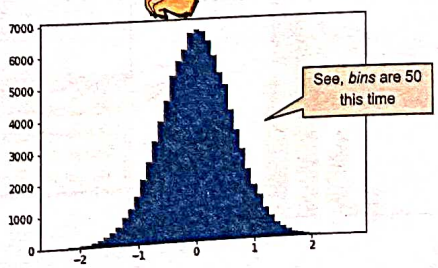


To see Histogram plot in action

Scan QR Code

2. **Plot a histogram from an ndarray y with 50 bins**

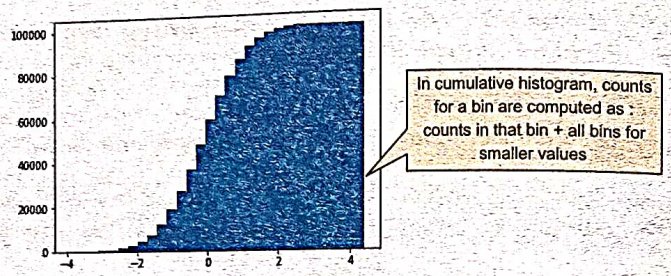
```
pl.hist(y, bins = 50)
```



3. Plot a cumulative histogram of ndarray x with 30 bins

For cumulative histogram, argument **cumulative** of hist( ) function is used.

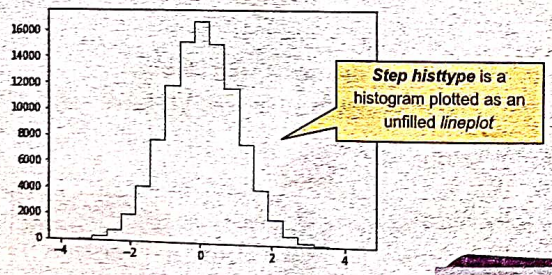
```
pl.hist(x, bins = 30, cumulative = True)
```



4. Plot ndarray x's histogram as *'step'* type histogram with 20 bins

To specify histogram type, argument **histtype** is to be used

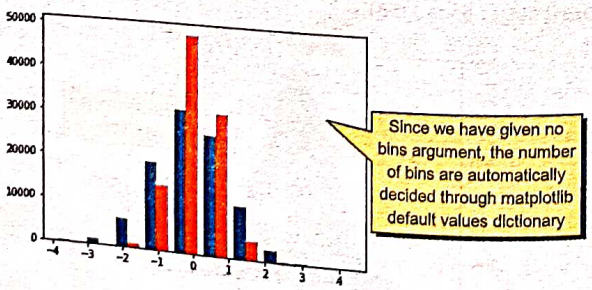
```
pl.hist(x, bins=20, histtype = 'step')
```



5. Plot both ndarray *x* and *y* in same histogram with

```
pl.hist([x, y])
```

To plot multiple arrays, provide them in form a sequence e.g., we have given arrays x and y in a list as [x, y]



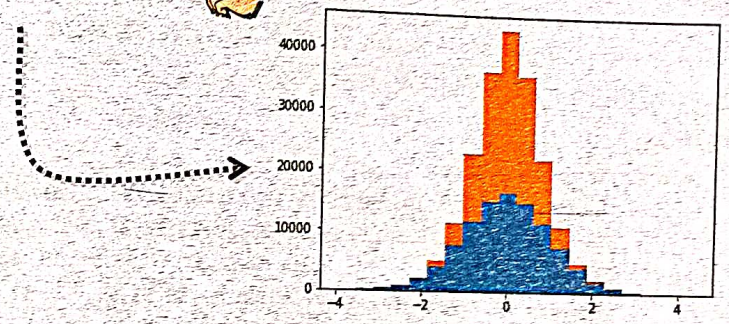
NOTE

A histogram is a summarisation tool for discrete or continuous data. A histogram provides a visual interpretation of numerical data by showing the number of data points that fall within a specified range of values (called ***bins***).

6. Plot a *stacked bar* type histogram from both ndarray *x* and *y*

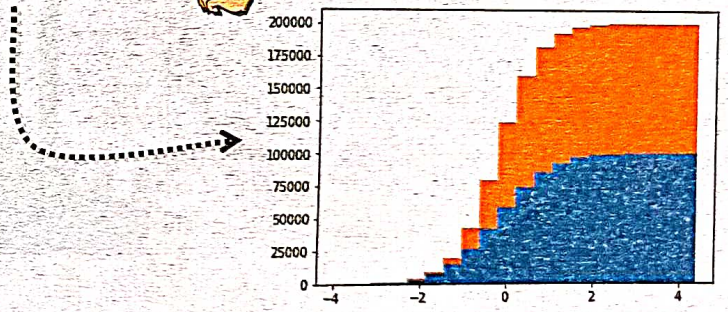
(a) regular histogram

```
pl.hist([x,y], histtype = 'barstacked')
```



(b) cumulative histogram

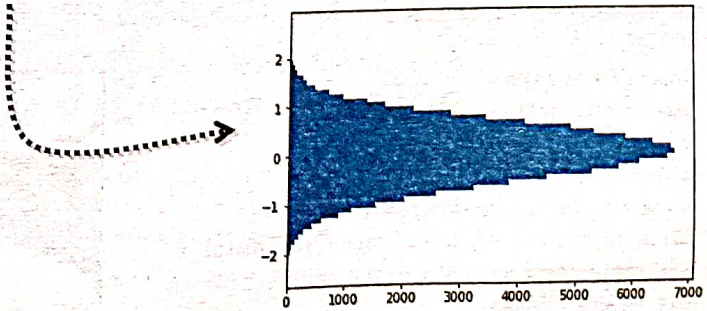
```
pl.hist([x,y], histtype = 'barstacked', cumulative = True)
```



7. Plot a horizontal histogram from ndarray *y* with 50 bins

To change the orientation of the histogram, we can use **orientation** argument.

```
pl.hist(y, bins = 50, orientation = 'horizontal')
```



EXAMPLE 27 *A survey gathers height and weight of 100 participants and recorded the participants' ages as :*

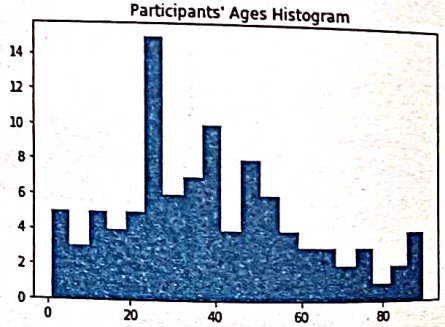
```
ages = [ 1, 1,2,3,5,7,8,9,10, 10,11,13,13,15,16,17,18, 19,20, 21,
         21,23,24,24,24, 25,25,25,25,26,26,26,27,27,27,27,27, 29,30,
         30,30,30,31,33,34,34,34,35,36,36,37,37,37,38,38,39, 40,40, 41,
         41,42, 43,45,45,46,46, 46, 47,48,48,49,50,51,51, 52, 52, 53, 54,
         55,56,57,58,60, 61,63,65,66,68,70,72,74, 75,77,81,83,84,87,89,90,91]
```

Write a program to plot a histogram from above data with 20 bins.

SOLUTION

```
import matplotlib.pyplot as plt.
ages = [ 1, 1,2,3,5,7,8,9,10, 10,11,13,13,15,16,17,18, 19,20, 21,
         21,23,24,24,24, 25,25,25,25,26,26,26,27,27,27,27,27,29,30,
         30,30,30,31,33,34,34,34,35,36,36,37,37,37,38,38,39, 40,40,41,
         41,42, 43,45,45,46,46, 46, 47,48,48,49,50,51,51, 52, 52, 53, 54,
         55,56,57,58,60, 61,63,65,66,68,70,72,74, 75,77,81,83,84,87,89,90,91]
plt.hist(ages, bins = 20)
plt.title ("Participants' Ages Histogram")
plt.show()
```

Output



EXAMPLE 28 *Prof Awasthi is doing some research in the field of Environment. For some plotting purposes, he has generated some data as :*

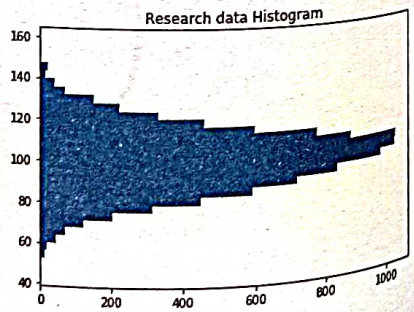
```
mu = 100
sigma = 15
x = mu + sigma * numpy.random.randn(10000)
```

Write a program to plot this data on a horizontal histogram with this data.

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
mu = 100
sigma = 15
x = mu + sigma * np.random.randn(10000)
plt.hist(x, bins = 30, orientation = 'horizontal')
plt.title('Research data Histogram')
plt.show()
```

Output



EXAMPLE 29 *Prof Awasthi is doing some research in the field of Environment. For some plotting purposes, he has generated some data as:*

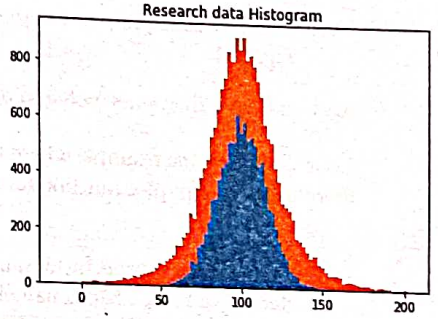
```
mu = 100
sigma = 15
x = mu + sigma * numpy.random.randn(10000)
y = mu + 30 * np.random.randn(10000)
```

Write a program to plot this data on a bar-stacked horizontal histogram with both x and y.

SOLUTION

```
import numpy as np
import matplotlib.pyplot as plt
mu = 100
sigma = 15
x = mu + sigma * np.random.randn(10000)
y = mu + 30 * np.random.randn(10000)
plt.hist([x,y], bins = 100, histtype ='barstacked')
plt.title('Research data Histogram')
plt.show()
```

Output



3.7 Creating Frequency Polygons

A frequency polygon is a type of frequency distribution graph. In a frequency polygon, the number of observations is marked with a single point at the midpoint of an interval. A straight line then connects each set of points. Frequency polygons make it easy to compare two or more distributions on the same set of axes[5].

Let's look at an example of a frequency polygon.

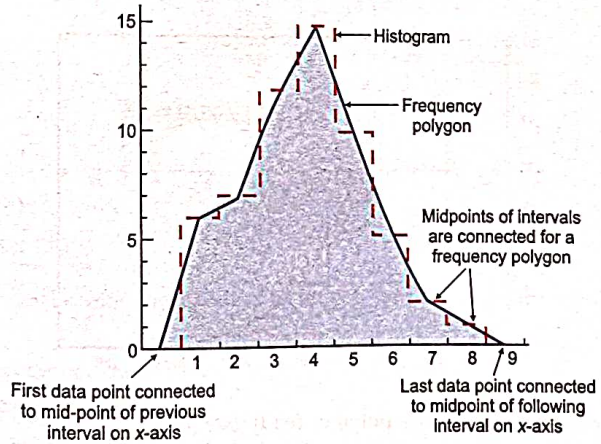


Figure 3.4 Frequency polygon

5. Hennekens, 1987

Python's **pyplot** module of **matplotlib** provides no separate function for creating frequency polygon. Therefore, to create a frequency polygon, what you can do is :

(*i*) Plot a histogram from the data.

(*ii*) Mark a single point at the midpoint of an interval/bin.

(*iii*) Draw straight lines to connect the adjacent points.

(*iv*) Connect first data point to the midpoint of previous interval on *x*-axis (as explained in Fig. 3.4).

(*v*) Connect last data point to the midpoint of following interval on x-axis (as explained in Fig. 3.4).

NOTE

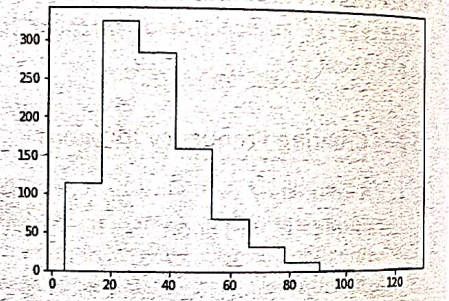
A frequency polygon is a type of frequency distribution graph. In a frequency polygon, the number of observations is marked with a single point at the midpoint of an interval.

And you have frequency polygon drawn on your histogram.

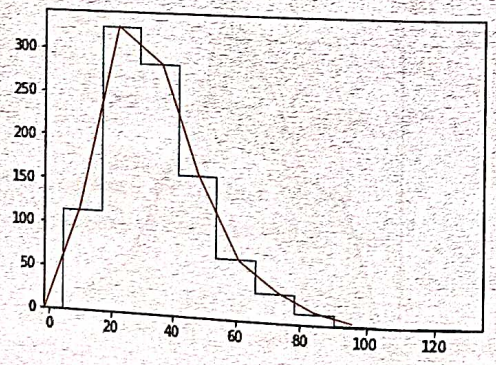
Consider following example where we are creating a frequency polygon. Please remember that there is not any pyplot function for this, so we shall do the following :

(*i*) Create a step type histogram from the data. We have a Series namely **com** that stores some 1000 values. Plotting a step histogram from the same.

```
pl.hist(com, bins = 10, histtype = 'step')
```



(*ii*) Join midpoints of each set of adjacent bins to create frequency polygon.



(*iii*) We have desired frequency polygon ready (*see* above).

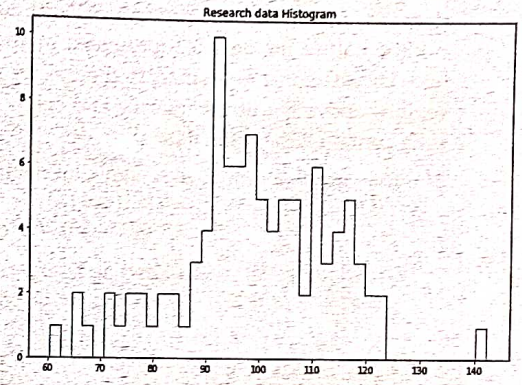
There are formulas to create a best-fit line programmatically but covering that is beyond the scope of the book.

EXAMPLE **30** *Using Prof Awasthi's research data (of previous example), create a frequency polygon.*

SOLUTION

Step 1 : Creating histogram first

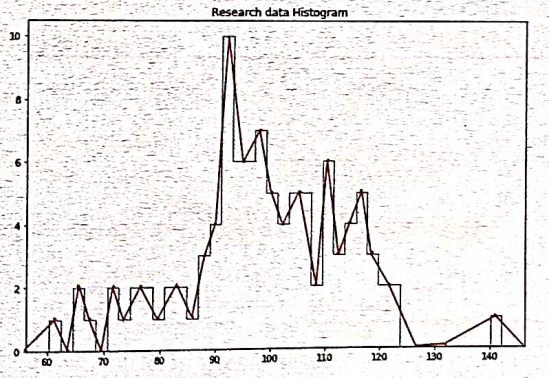
```
import numpy as np
import matplotlib.pyplot as plt
mu = 100
sigma = 15
x = mu + sigma * np.random.randn(100)
plt.figure( figsize= (10, 7))
y = np.arange(len(x))
plt.hist(x, bins = 40, histtype = 'step')
plt.title('Research data Histogram')
plt.show()
```



Step 2 : Plot a best fit line manually

2.1 **Mark midpoints of every bin**

2.2 **Join midpoints of every bin with a line**



3.8 Creating Box Plots

The box plot has become the standard technique for presenting the 5-number summary which consists of :

(*i*) the minimum range value

(*ii*) the maximum range value

(*iii*) the upper quartile,

(*iv*) the lower quartiles, and

(*v*) the median.

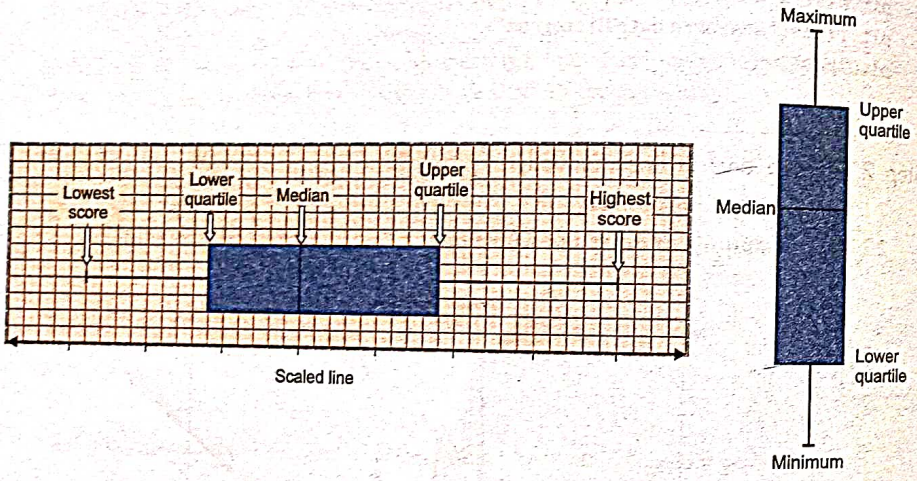
This collection of values is a quick way to summarize the distribution of a dataset. In addition, this reduced representation afforded by the 5-number summary provides a more straightforward way to compare datasets, since only these characteristic values need to be analysed.

A box plot is used to show the range and middle half of ranked data. Ranked data is numerical data such as *numbers* etc. The middle half of the data is represented by the box. The highest and lowest scores are joined to the box by straight lines. The regions above the upper quartile and below the lower quartile each contain 25% of the data.

The five-number summary is shown in the diagram below.

NOTE

The box plot uses five important numbers of a data range : the extremes (the highest and the lowest numbers), the median, and the upper and lower quartiles, making up the five-number summary.



(*a*) Anatomy of a box plot.

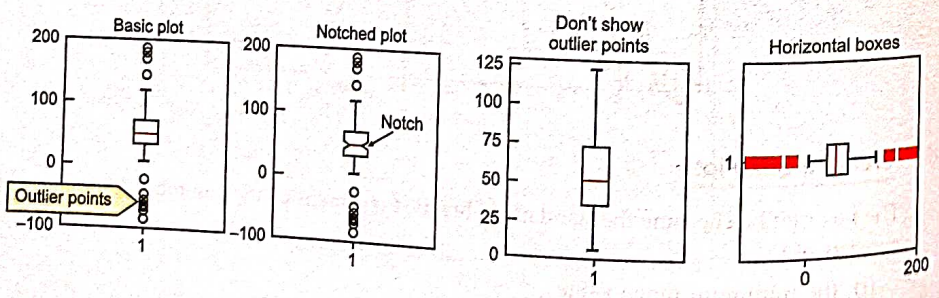


Figure 3.5 (*b*) Box plot with various settings.

NOTE The box plot is used to show the range and the middle half of the ranked data.

Creating boxplots with boxplot( ) of Pyplot

Pyplot module's **boxplot( )** allows you to create boxplots.

The syntax of **boxplot( )**[6] is :

```
matplotlib.pyplot.boxplot(x, notch = None, vert = None, meanline = None,
                          showmeans = None, showbox = None, )
```

Parameters :

x	Array or a sequence of vectors. The input data.
notch	bool, optional (*False*) ; If True, will produce a notched box plot. Otherwise, a rectangular boxplot is produced.
vert	bool, optional (*True*) ; If True (default), makes the boxes vertical. If False, everything is drawn horizontally.
meanline	bool, optional (*False*) ; If True (and showmeans is True), will try to render the mean as a line spanning the full width of the box.
showbox	bool, optional (*True*) ; Show the central box.
showmeans	bool, optional (*False*) ; Show the arithmetic means.
patch_artist	bool, optional (*None*) ; Fills the box if *True*.
labels	list sequence with labels of multiple sequences being plotted.

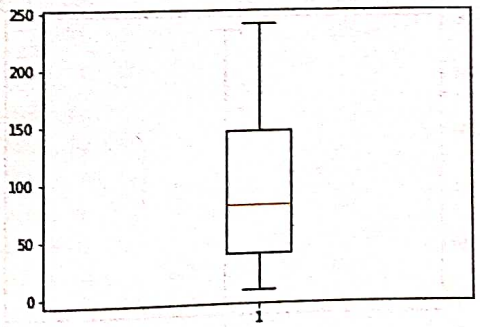
Have a look at following examples :

Given sample data :

```
# pyplot is imported as pl
ary = [5, 20, 30, 45, 60, 80, 100, 140, 150, 200, 240]
```

1. **Using the above sample data (ary), draw the plain boxplot**

```
pl.boxplot(ary)
```



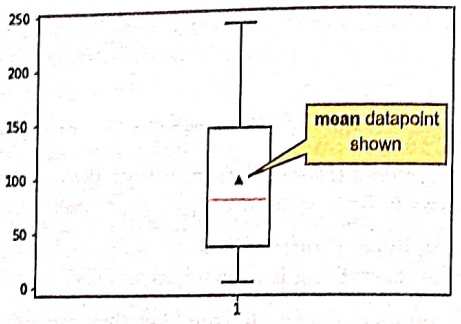
6. This is minimal, usable form ; for complete syntax refer to documentation.

2. **Using the above sample data (ary), draw the boxplot with mean shown**
To show the mean value, argument **showmeans** is to be used.

```
pl.boxplot(ary, showmeans = True)
```

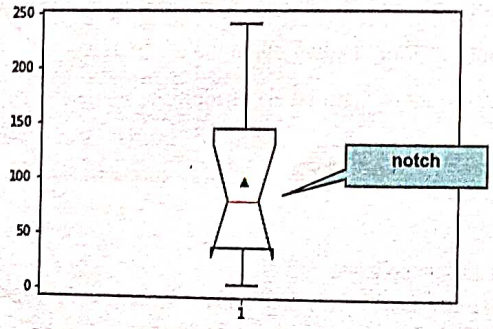
To see boxplot( ) chart in action

Scan QR Code



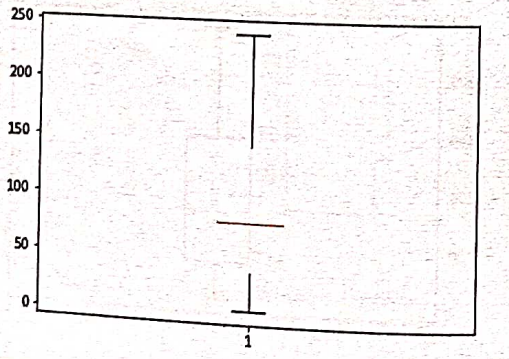
3. **Draw a notched boxplot for the same**

```
pl.boxplot(ary, notch = True, showmeans = True)
```



4. **Draw the boxplot with above data without the central box**

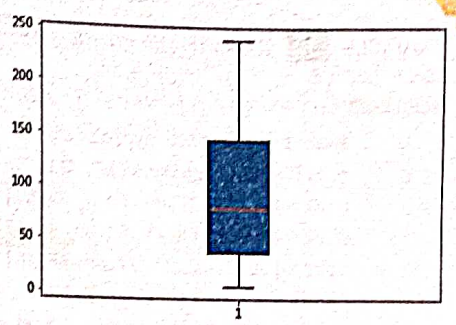
```
pl.boxplot(ary, showbox = False)
```



Using **vert** argument of boxplot( ), you can change the orientation of the boxplot.

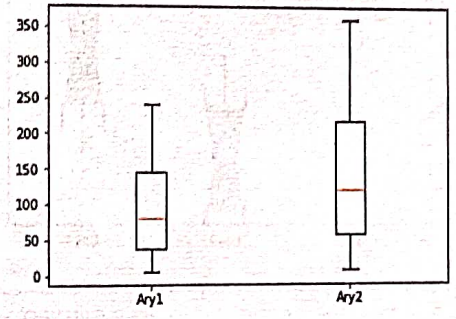
5. *Draw the boxplot with a filled box*

```
pl.boxplot(ary, patch_artist = True)
```



6. *Draw multiple data sequences in the same boxplot with individual labels.*

```
ary1 = np.array(ary)
ary2 = ary1 * 1.502
plt.boxplot([ary1, ary2], labels = ['Ary1', 'Ary2'])
```



EXAMPLE **31** *Three days' prices are available in three lists as shown below:*

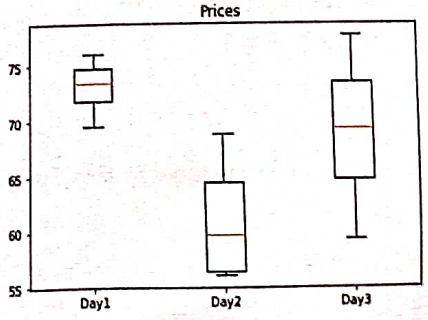
```
Day1 = [74.25, 76.06, 69.5, 72.55]
Day2 = [56.03, 68.71, 62.89, 56.42]
Day3 = [59.3, 72.07, 77.65, 66.46]
```

Write a program to create filled boxplot from this data with their unique labels.

SOLUTION

```
import matplotlib.pyplot as plt
Day1 = [74.25, 76.06, 69.5, 72.55]
Day2 = [56.03, 68.71, 62.89, 56.42]
Day3 = [59.3, 72.07, 77.65, 66.46]
finlist = [Day1, Day2, Day3]
plt.boxplot(finlist, labels = ['Day1', 'Day2', 'Day3'])
plt.title("Prices")
plt.show( )
```

Output

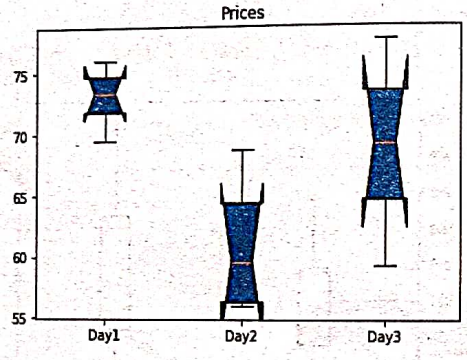


EXAMPLE 32 *Considering the Price data of previous example, write a program to create notched boxplot, which should be filled.*

SOLUTION

```
import matplotlib.pyplot as plt
plt.figure(figsize = (10, 7))
Day1 = [74.25, 76.06, 69.5, 72.55]
Day2 = [56.03, 68.71, 62.89, 56.42]
Day3 = [59.3, 72.07, 77.65, 66.46]
finlist = [Day1, Day2, Day3]
plt.boxplot(finlist, patch_artist = True, notch = True, labels = ['Day1', 'Day2', 'Day3'])
plt.title("Prices")
plt.show()
```

Output

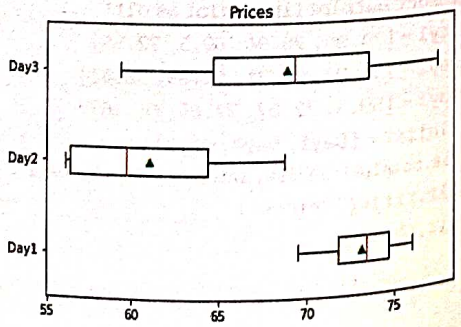


EXAMPLE 33 *Considering the Price data of previous example, write a program to create a vertical boxplot, with means shown.*

SOLUTION

```
import matplotlib.pyplot as plt
Day1 = [74.25, 76.06, 69.5, 72.55]
Day2 = [56.03, 68.71, 62.89, 56.42]
Day3 = [59.3, 72.07, 77.65, 66.46]
finlist = [Day1, Day2, Day3]
plt.boxplot(finlist, vert = 0, showmeans = True, labels = ['Day1', 'Day2', 'Day3'])
plt.title("Prices")
plt.show()
```

The output produced by above code is :



3.9 Plotting Data from a DataFrame

Till now you have plotted data from either linear lists, or 1D arrays. You can also plot data from a DataFrame using its columns selectively.

You can do it in *two* ways :

(*i*) Using PyPlot's graph functions

(*ii*) Using DataFrame's **plot( )** function. It is available from version 0.17.0 onwards.

3.9.1 Plotting a DataFrame's Data using PyPlot's Graph Functions

To plot a DataFrame's data, just pass its column name to the Pyplot's graph functions (**plot( ), bar( ), barh( ), scatter( ), boxplot( ), hist( )**). It will treat the passed column's data as a Series and plot it, *e.g.*, if we have a DataFrame **df2** as shown below :

DataFrame df2

	Age	Projects
0	30	13
1	27	17
2	32	16
3	40	20
4	28	21
5	32	14

DataFrame df

	Name	Sex	Position	City	Age	Projects	Budget
0	Rabia	F	Manager	Bangalore	30	13	8
1	Evan	M	Programmer	New Delhi	27	17	13
2	Jia	F	Manager	Chennai	32	16	32
3	Lalit	M	Manager	Mumbai	40	20	21
4	Jaspreet	M	Programmer	Chennai	28	21	17
5	Suji	F	Programmer	Bangalore	32	14	10

DataFrame fdf

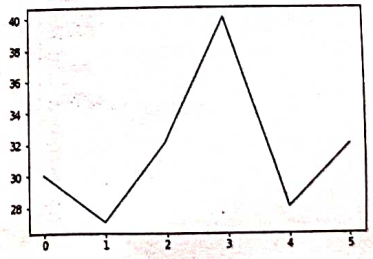
	Day1	Day2	Day3	Day4	Day5
0	74.25	56.03	59.30	69.00	89.65
1	76.06	68.71	72.07	78.47	79.65
2	69.50	62.89	77.65	65.53	80.75
3	72.55	56.42	66.46	76.85	85.08

Reference 3.2

And if you write the code as :

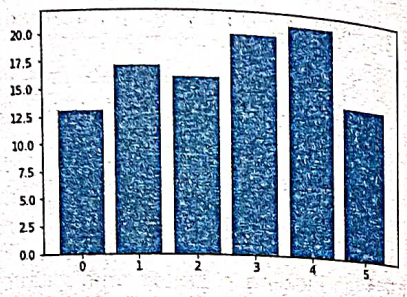
```
import pandas as pd
import matplotlib.pyplot as plt
:   #df2 created or loaded
plt.plot(df2.Age)
```

It will give you a plot as shown here.



You can also plot a bar chart using this DataFrame's data as :

```
plt.bar(df2.index, df2.Projects)
```



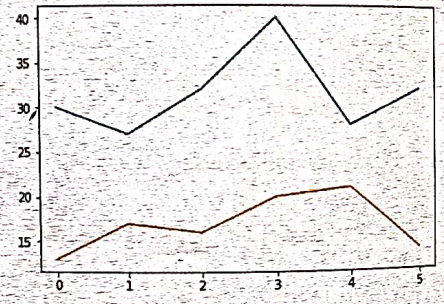
With all plotting functions of PyPlot, you can send individual column names of a DataFrame to plot their data. However, with plot( ), you can send just the name of DataFrame.

The plot( ) can take a DataFrame's name and will plot all columns

Unlike other plotting functions of PyPlot, with *plot( )* you can send just the dataframe's name and it will plot all the columns of a DataFrame. It is handy if your DataFrame has all numeric columns, *e.g.*, if we have a DataFrame **df2** as shown in *reference* 3.2.

And if you write the code as :

```
import pandas as pd
import matplotlib.pyplot as plt
:   # df2 created or loaded
plt.plot(df2)
```

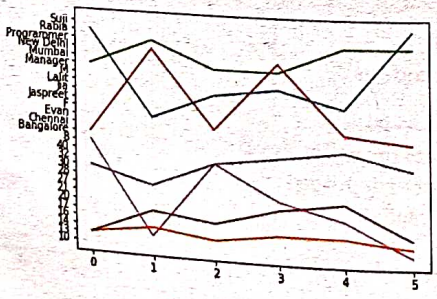


It will give you a plot as shown adjacent, where it has plotted all the columns (*Age* and *Projects*) in the DataFrame **df2.**

But what if your DataFrame has some non-numeric columns. It will still try to plot everything, *e.g.*, considering the DataFrame **df** (*Reference* 3.2), if you write the code as :

```
import pandas as pd
import matplotlib.pyplot as plt
:   #df created or loaded
plt.plot(df)
```

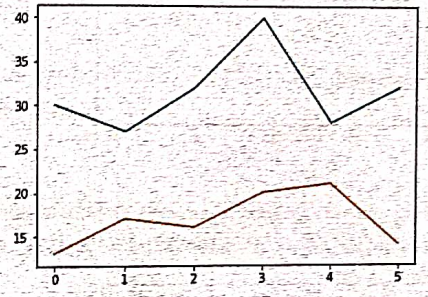
It will give you a plot as shown adjacent, which won't make much of sense.



To avoid a situation like above (plotting of non-numeric columns), you can explicitly specify the column names to be plotted, *e.g.*, to plot only the *Age* and *Projects* columns against the index values of the above DataFrame **df**, you may write :

```
import pandas as pd
import matplotlib.pyplot as plt
:   #df2 created or loaded
plt.plot(df.index, df.Age, df.Projects)
```

It will give you a plot as shown below where it has plotted only ***Age*** and ***Projects*** columns against ***index.***



3.9.2 Plotting a DataFrame's Data using DataFrame's plot( )

Pandas provides a function ***plot( )*** which you can use with DataFrame as :

```
<DF>.plot( )
```

And which will plot from the data of the DataFrame automatically. The DataFrame's **plot( )** is a versatile function, which can plot all types of chart by just specifying **kind** argument.

Various argument that **<DF>.plot( )** can take are[7] :

dataSeries or DataFrame : The object for which the method is called.

kind : type of the plot, can take values as

'line' : line plot (default)
'bar' : vertical bar plot
'barh ' : horizontal bar plot
'hist' : histogram
'box' : boxplot
'pie' : pie plot
'scatter' : scatter plot
And some other plot types

NOTES

- With scatter, you have to explicitly specify the x and y arguments.
- If you do not give kind argument at all, line chart will be plotted.

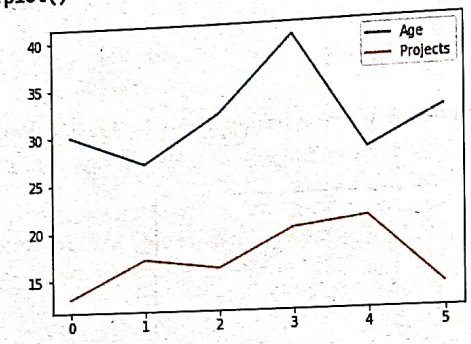
Before we show you the use of <DF>.plot( ), let us talk about the advantages of it :

- It plots only the numeric columns unlike ***plot( )*** of PyPlot when used with a DataFrame.
- It automatically adds legends for the plotted data.

7. Please note we are not giving full syntax of this function here as it is beyond the scope of this book.

Consider the **df** and **df2** DataFrame you have used above and plot these using pandas plot( ):

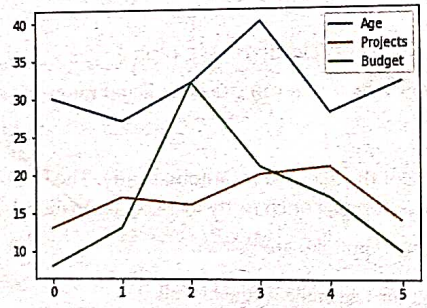
```
import pandas as pd
import matplotlib.pyplot as plt
:    #df2 created or loaded
df2.plot()        # df2 contains numeric columns only
```



See, *<DF>.plot( )* has added legends automatically.

The kind argument is missing, hence line chart is plotted.

```
:
df.plot()         # df contains numeric as well as non-numeric columns
# above code line is equivalent to df.plot(kind = 'line')
```



See, *<DF>.plot( )* has :

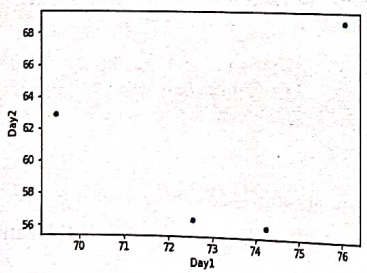
(i) Plotted only the numeric columns of DataFrame *df*

(ii)added legends automatically.

Just play with the kind argument and create different chart types.

- **Creating scatter charts with DF.plot( )** *(see left figure below)*

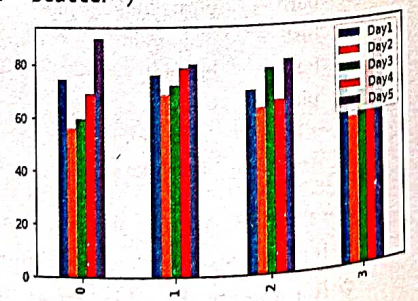
```
fdf.plot(x = 'Day1', y ='Day2', kind ='scatter')
```



- **Creating bar charts with DF.plot( )**

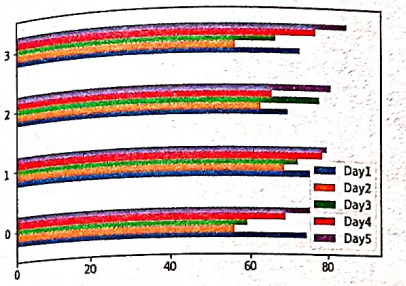
```
fdf.plot(kind = 'bar')
```

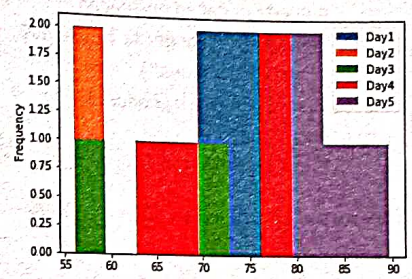
(see right figure above)



- **Creating horizontal bar charts with DF.plot( )** *(see left figure below)*

```
fdf.plot(kind = 'barh')
```



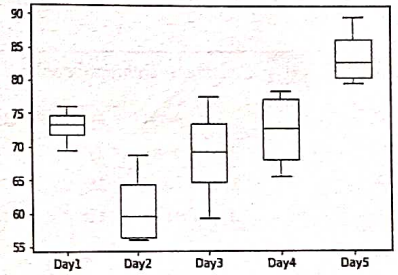


- **Creating histogram with DF.plot( )** *(see right figure above)*

```
fdf.plot(kind = 'hist')
```

- **Creating boxplots with DF.plot( )**

```
fdf.plot(kind = 'box')
```



EXAMPLE **34** *Consider following DataFrame* ***prodf.***

	Fruits	Pulses	Rice	Wheat
Andhra P.	7830.0	931.0	7452.4	NaN
Gujrat	11950.0	818.0	1930.0	2737.0
Kerala	113.1	1.7	2604.8	NaN
Punjab	7152.0	33.0	11586.2	16440.5
Tripura	44.1	23.2	814.6	0.5
Uttar P.	24169.2	2184.4	13754.0	30056.0

Write a program to plot a scatter chart with the columns Pulses.

SOLUTION

Since the given DataFrame **prodf** does not have numeric *index*, we cannot use it for plotting because *x* has be numeric for **scatter( )**.

We can create numeric values for *x* axis as :

```
x = range(0, len(prodf) )        # will generate [0,1, 2, 3, 4, 5]
```

OR

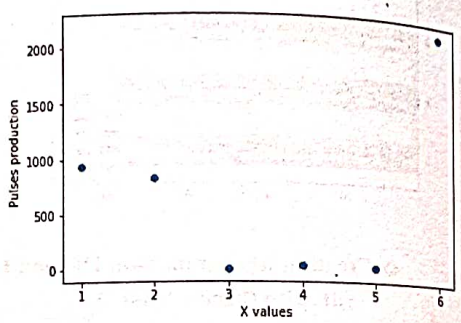
```
x = range(1, len(prodf) + 1 )    # will generate [1, 2, 3, 4, 5, 6]
```

Now the program code for above given problem will be :

With pyplot's scatter, you have to specify a DataFrame's column name along with the DataFrame's name

```
import matplotlib.pyplot as plt
:    # prodf created or loaded
plt.scatter(x = range(1, len(prodf) + 1 ), y = prodf.Pulses)
plt.xlabel("X values")
plt.ylabel("Pulses production")
plt.show()
```

The output produced by above code is :



EXAMPLE **35** *Consider a DataFrame mksdf as shown below :*

	Name	Age	PreBoardMarks	BoardMarks
0	Karan	17	4	25
1	Alex	19	24	94
2	Ani	18	31	57
3	Javed	18	2	62
4	Amrit	17	3	70

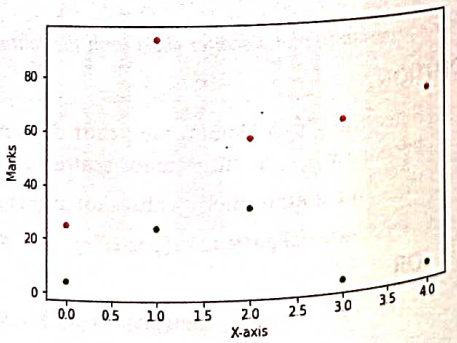
Write a program to plot pre-board-marks and board marks from above DataFrame on the same scatter chart.

SOLUTION

With pyplot's scatter, you have to specify a DataFrame's column name along with the DataFrame's name

```
import pandas as pd
import matplotlib.pyplot as plt
:       # mksdf created or loaded
plt.scatter(x = mksdf.index, y = mksdf.PreBoardMarks, c = 'g', s = mksdf.Age)
plt.scatter(x = mksdf.index, y = mksdf.BoardMarks, c = 'r', s = mksdf.Age)
plt.xlabel("X-axis")
plt.ylabel("Marks")
plt.show()
```

The output produced by above code is :



EXAMPLE **36** *Consider the DataFrame of previous example (mksdf). Write a program to plot 'BoardMarks' against 'PreBoardMarks' in a scatter chart.*

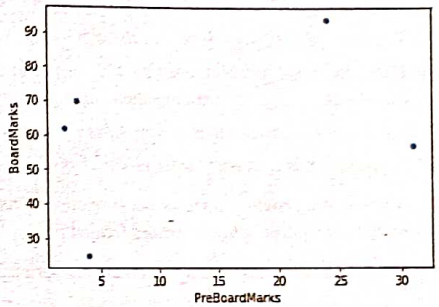
SOLUTION

For x argument, a numeric argument is chosen.

With <DF>.plot( ), the DataFrame's columns name are simply written without writing

```
import pandas as pd
import matplotlib.pyplot as plt
:       # mksdf created or loaded
mksdf.plot(kind = 'scatter', x = 'PreBoardMarks', y = 'BoardMarks')
plt.xlabel('PreBoardMarks')
plt.ylabel('BoardMarks')
plt.show()
```

The output produced by above code is :



With this we have come to the end of this chapter. Let us quickly revise what we have learnt so far.

Check Point **3.1**

1. What is data visualization ?
2. Name the Python library generally used for data visualization.
3. Is Pyplot a Python library ? What is it ?
4. Name the function you will use to create a horizontal bar chart.
5. Which argument will you provide to change the following in a line chart ?
   (i) width of the line
   (b) colour of the line
6. What is a marker ? How can you change the marker type and colour in a plot ?
7. Using which function of Pyplot can you plot histograms ?
8. Are bar charts and histograms the same ?
9. Name various types of histogram plots that you can create using pyplot.
10. What is a frequency polygon ?
11. What is the use of boxplot ?
12. Using which function of pyplot, can you create boxplots ?

LET US REVISE

- *Data visualization basically refers to the graphical or visual representation of information and data using visual elements like charts, graphs, and maps etc.*
- *The **matplotlib** is a Python library that provides many interfaces and functionality for 2D-graphics similar to MATLAB's in various forms.*
- *PyPlot is a collection of methods within matplotlib which allow user to construct 2D plots easily and interactively. PyPlot essentially reproduces plotting functions and behavior of MATLAB.*
- *In order to use pyplot on your computers for data visualization, you need to first import it in your Python environment by issuing **import** command for **matplotlib.pyplot**.*
- *Some commonly used chart types are line chart, bar chart, pie chart, scatter chart etc.*
- *A line chart or line graph is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments.*
- *You can create line charts by using pyplot's **plot( )** function.*
- *You can change line color, width, line-style, marker-type, marker-color, marker-size in plot( ) function.*
- *The **scatter chart** is a graph of plotted points on two axes that show the relationship between two sets of data.*

- *You can create scatter charts using either plot( ) function or scatter( ) function.*
- *A Bar Graph/Chart is a graphical display of data using bars of different heights.*
- *You can create bar chart using pyplot's bar( ) function.*
- *You can change colors of the bars, widths of the bars in bar( ) function.*
- *Use barh( ) function to create horizontal bar chart.*
- *The pie( ) creates a pie chart.*
- *The plot area is known as figure and every other element of chart is contained in it.*
- *The axes can be labelled using **xlabel( )** and **ylabel( )** functions.*
- *The limits of axes can be defined using **xlim( )** and **ylim( )** functions.*
- *The tick marks for axes values can be defined using **xticks( )** and **yticks( )** functions.*
- *The **title( )** function adds title to the plot.*
- *Using **legend( )** function, one can add legends to a plot where multiple data ranges have been plotted, but before that the data ranges must have their **label** argument defined in plot( ) or bar( ) function.*
- *The **loc** argument of legend( ) provides the location for legend, which by default is 1 or "upper right".*
- *A histogram is a summarisation tool for discrete or continuous data.*
- *A histogram provides a visual interpretation of numerical data by showing the number of data points that fall within a specified range of values (called bins).*
- *Pyplot module's hist( ) lets you create histograms.*
- *A frequency polygon is a type of frequency distribution graph.*
- *In a frequency polygon, the number of observations is marked with a single point at the midpoint of an interval.*
- *The box plot is used to show the range and the middle half of the ranked data.*
- *The boxplot( ) of pyplot lets you draw boxplots.*
- *A dataframe's data can be plotted in various chart types using <DF>.plot(kind = <type>) function.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. PyPlot is an interface of Python's _____ library.
   (a) seaborn (b) plotly (c) ggplot (d) matplotlib
2. For 2D plotting using a Python library, which library interface is often used ?
   (a) seaborn (b) plotly (c) matplotlib (d) matplotlib.pyplot
3. Which of the following is not a valid chart type ?
   (a) histogram (b) statistical (c) pie (d) box
4. Which of the following is not a valid plotting function of pyplot ?
   (a) plot( ) (b) bar( ) (c) line( ) (d) pie( )
5. Which of the following plotting functions does not plot multiple data series ?
   (a) plot( ) (b) bar( ) (c) pie( ) (d) barh( )
6. The plot which tells the trend between two graphed variables is the _____ graph/chart.
   (a) line (b) scatter (c) bar (d) pie

7. The plot which tells the correlation between two variables which may not be directly related is _____ graph/chart.
   (a) line (b) scatter (c) bar (d) pie
8. A _____ is a summarisation tool for discrete or continuous data.
   (a) quartile (b) histogram (c) mean (d) median
9. A visual representation of the statistical five number summary of a given dataset is known as _____ .
   (a) histogram (b) frequency distribution
   (c) boxplot (d) frequency polygon
10. Which of the following functions is used to create a line chart ?
    (a) line( ) (b) plot( ) (c) chart( ) (d) plotline( )
11. Which of the following function will produce a bar chart ?
    (a) plot( ) (b) bar( ) (c) plotbar( ) (d) barh( )
12. Which of the following function will create a vertical bar chart ?
    (a) plot( ) (b) bar( ) (c) plotbar( ) (d) barh( )
13. Which of the following function will create a horizontal bar chart ?
    (a) plot( ) (b) bar( ) (c) plotbar( ) (d) barh( )
14. To specify the style of line as dashed, which argument of plot( ) needs to be set ?
    (a) line (b) width (c) style (d) linestyle
15. The datapoints plotted on a graph are called _____ .
    (a) points (b) pointers (c) marks (d) markers
16. A _____ graph is a type of chart which displays information as a series of data points connected by straight line segments.
    (a) line (b) bar (c) pie (d) boxplot
17. To create scatter charts using plot( ), which argument is skipped ?
    (a) marker (b) linestyle
    (c) markeredgecolor (d) linewidth
18. In scatter( ), which argument is used to specify the size of datapoints ?
    (a) size (b) s (c) marker (d) markersize
19. Which argument of bar( ) lets you set the thickness of bar ?
    (a) thick (b) thickness (c) width (d) barwidth
20. Which function lets you set the title of the plot ?
    (a) title( ) (b) plottitle( ) (c) graphtitle( ) (d) all of these
21. Which function would you use to set the limits for *x*-axis of the plot ?
    (a) limits( ) (b) xlimits( ) (c) xlim( ) (d) lim( )
22. Which function is used to show legends ?
    (a) display( ) (b) show( ) (c) legend( ) (d) legends( )
23. Which argument must be set with plotting functions for **legend( )** to display the legends ?
    (a) data (b) label (c) name (d) sequence

24. Which function is used to create a histogram ?
(*a*) histo( ) (*b*) histogram( ) (*c*) hist( ) (*d*) histtype
25. Which argument in hist( ) is used to create a stacked bar type histogram ?
(*a*) histt (*b*) histtype (*c*) type (*d*) barstacked
26. Which of the following functions can plot only one data series ?
(*a*) plot( ) (*b*) bar( ) (*c*) boxplot( ) (*d*) pie( )
27. Which argument must be provided to create wedges out of a pie chart ?
(*a*) label (*b*) autopct (*c*) explode (*d*) wedge
28. Which argument should be set to display percentage share of each pie on a pie chart ?
(*a*) label (*b*) autopct (*c*) explode (*d*) wedge
29. Which function creates a boxplot ?
(*a*) box( ) (*b*) plot( ) (*c*) boxplot( ) (*d*) showbox( )
30. Which argument of boxplot( ) is used to create a filled boxplot ?
(*a*) fill (*b*) box (*c*) patch_artist (*d*) patch

Fill in the Blanks

1. A _____ is a plot that shows the underlying frequency distribution of a set of continuous data.
2. Pyplot interface is a collection of methods within _____ library of Python.
3. Pyplot's _____ function is used to create line charts.
4. Pyplot's _____ function is used to create horizontal bar charts.
5. Pyplot's _____ function is used to create scatter charts.
6. Pyplot's _____ function is used to create histogram.
7. The datapoints plotted on a graph are called _____ .
8. The _____ argument of plot( ) specifies the width for the line.
9. The _____ argument of plot( ) specifies the style of the line.
10. The _____ argument of bar( ) specifies the bar width.
11. The _____ function is used to specify ticks for *x*-axis.
12. To save a plot, _____ function is used.
13. The _____ argument of hist( ) is set to create a horizontal histogram.
14. The _____ argument shows the arithmetic mean on a boxplot.
15. The _____ argument in a boxplot( ) creates a notched boxplot.
16. The _____ argument of legend( ) provides the location of legend.

True/False Questions

1. PyPlot is a sub-library of matplotlib library.
2. Statement **import pyplot.matplotlib** is a valid statement for working on pyplot functions.
3. By default, pie chart is printed in elliptical or oval shape.
4. The default shape of pie chart cannot be change from oval.

5. A line chart can be plotted using pyplot library's line( ) function.
6. A line chart can be plotted using pyplot library's plot( ) function.
7. A bar chart can be plotted using pyplot library's bar( ) function.
8. A bar chart can be plotted using pyplot library's barh( ) function.
9. It is not possible to plot multiple series of values in the same bar graph.
10. A standard marker of representing a non-number data in Python libraries is NaN.
11. If the linestyle argument is missing along with markerstyle-string in a plot( ), a scatter type chart get created.
12. The bar( ) function can also create horizontal bar charts.
13. The pie( ) function can plot multiple data series.
14. The plot is always as per the data series being plotted irrespective of the **xlim( )**.
15. Frequency polygon is created from histogram.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *What is data visualization ? What is its significance ?*

Solution. Data visualization is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. In simple words, Data visualization is the process of displaying data/information in graphical charts, figures and bars.

Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization techniques or tools such as line chart, bar chart, pie chart, histogram, scatter chart etc. Thus with data visualization tools, information can be processed in efficient manner and hence better decisions can be made.

2. *What is Python's support for Data visualization ?*

Solution. Python supports data visualizations by providing some useful libraries for visualization. Most commonly used data visualization library is **matplotlib**.

Matplotlib is a Python library, also sometimes known as the plotting library. The matplotlib library offers very extensive range of 2D plot types and output formats. It offers complete 2D support along with limited 3D graphic support. It is useful in producing publication quality figures in interactive environment across platforms. It can also be used for animations as well.

There are many other libraries of Python that can be used for data visualization but matplotlib is very popular for 2D plotting.

3. *What is pyplot ? Is it a Python library ?*

Solution. The *pyplot* is one of the interfaces of *matplotlib* library of Python. This interface offers simple MATLAB style functions that can be used for plotting various types of charts using underlying *matplotlib* library's functionality.

Pyplot is an interface, *i.e.*, a collection of methods for accessing and using underlying functionality of a library, not a library. The *matplotlib* library has may other interfaces too, along with *pyplot* interface.

4. *Name some commonly used chart types.*

   Solution. Some commonly used chart types are line chart, bar chart, pie chart, scatter chart etc.

5. *Name the functions you will use to create a (i) line chart, (ii) bar chart, (iii) scatter chart.*

   Solution.

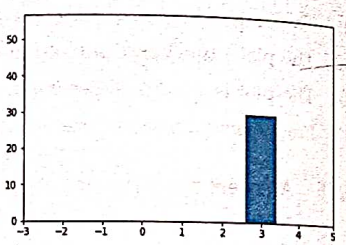
   (*i*) matplotlib.pyplot.plot( )

   (*ii*) matplotlib.pyplot.bar( )

   (*iii*) matplotlib.pyplot.plot( ) and matplotlib.pyplot.scatter( )

6. *Consider the code given below (all required libraries are imported) and the output produced by it. Why is the chart showing one bar only while we are plotting four values on the chart ?*

```
:
a = [3, 6, 9, 12]
b = [30, 48, 54, 48]
plt.xlim(–3, 5)
plt.bar(a,b)
plt.show()
```

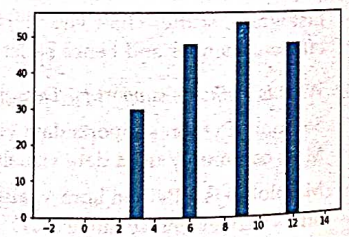


   Solution. The given chart is showing a single bar as the limits of *x*-axis have been set as –3 to 5. On this range, only one value from the data range being plotted falls *i.e.*, only a[0] and b[0] fall on this range. Thus only a single value b[0] *i.e.*, 30 is plotted against a[0] *i.e.*, 3.

7. *What changes will you make to the code of previous question so that the bars are visible for all four points ? But do keep in mind that the x-axis must begin from the point –3.*

   Solution. If we change the limits of X-axis so that all the points being plotted fall in the range of limits, all values will show. Thus, we have changes the limits from –3 to 15, in place of –3 to 5.

```
plt.xlim(–3, 15)
plt.bar(a,b)
plt.show()
```



8. *Why is following code not producing any result ? Why is it giving errors ?*

   *(**Note.** All required libraries have been imported and are available)*

```
a = range(10, 50, 12)
b = range(90, 200, 20)
matplotlib.pyplot.plot(a, b)
```

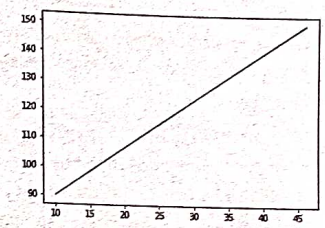
   Solution. The above code is producing error because the two sequences being plotted *i.e.*, **a** and **b** do not match in shape. While sequence '**a**' contains 4 elements, sequence '**b**' contains 6 elements. For plotting, it is necessary that the two sequences being plotted must match in their shape.

9. *What changes will you recommend to rectify the error in previous question's code ?*

   Solution. Since both the sequences being plotted must match in their shape, we can achieve this either by adding two elements to sequence a so that it has the same shape as sequence **b** (*i.e.*, 6 elements) or by removing two elements from sequence **b** so that it matches the shape of sequence **a** (*i.e.*, 4 elements).

   For instance,

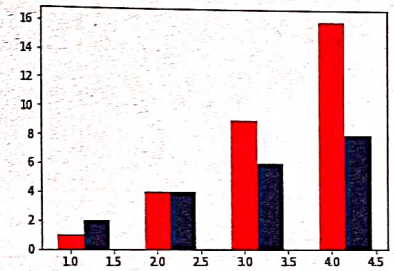
```
a = range(10, 50, 12)
b = range(90, 160, 20)
matplotlib.pyplot.plot(a, b)
```



10. *Given an ndarray p as ([1, 2, 3, 4]). Write code to plot a bar chart having bars for p and p**2 (with red color) and another bar for p vs p*2 (with blue color). (assume that libraries have been imported)*

    Solution.

```
plt.bar(p, p**2, color = 'r', width = 0.3)
plt.bar(p+0.3, p*2, color = 'b', width = 0.3)
```



11. *What is scatter chart? How is it different from line chart ?*

    Solution. The scatter chart is a graph of plotted points that show the relationship between two sets of data. With a scatter plot, a mark or marker (usually a dot or small circle), represents a single data point. With one mark (point) for every data point a visual distribution of the data can be seen. Depending on how tightly the points cluster together, you may be able to discern a clear trend in the data.

    The difference is that with a scatter plot, the decision is made from the data points such that the individual points should not be connected directly together with a line but, instead express a trend.

12. *What is histogram ? How is it useful ?*

    Solution. A histogram is a statistical tool used to summarise discrete or continuous data. It provides a visual interpretation of numerical data by showing the number of data points that fall within a specified range of values (called "***bins***").

13. *Following code is plotting the desired graph but legends are not showing despite giving the legend() of PyPlot. What could be the reason? Suggest a solution for the problem.*

```
plt.plot(x, y)
plt.plot(x, z)
plt.legend(loc = "upper left")
```

    Solution. The above code won't print the legends because with the plot( ), the labels are missing. The legend( ) will work only when we specify label for data series being plotted in the plot( ).

    The solution for above problem will be :

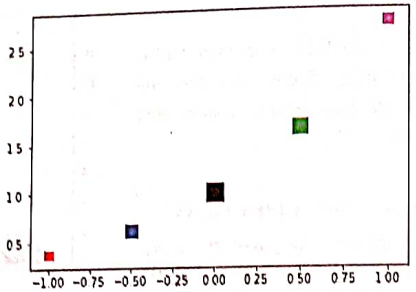
```
plt.plot(x, y, label = "Y data")
plt.plot(x, z, label = "Z data")
plt.legend(loc = "upper left")
```

Practical Problems

14. *Given two arrays namely* ***arr1*** *and* ***arr2****, each having 5 values. Create a scatter chart so that each data points gets a different color, different size. Keep the marker style as square.*

Solution.

```
import matplotlib.pyplot as pl
# arr1, arr2 defined here
colors = ['r', 'b', 'k', 'g', 'm']
sizes = [50, 120, 220, 150, 80]
pl.scatter(arr1, arr2, c = colors, s = sizes, marker = 's')
```



15. *Given a dataframe* ***df*** *as shown below :*

```
impot pandas as pd
x = { 'speed' : [10, 15, 20, 18, 19], \
      'meters' : [122, 150, 190, 230, 300], \
      'weight' : [0.2, 0.3, 0.1, 0.85, 0.0] }
df = pd.DataFrame(x)
df
```

	meters	speed	weight
0	122	10	0.20
1	150	15	0.30
2	100	20	0.10
3	230	18	0.85
4	300	19	0.00

Write code to create scatter graphs from

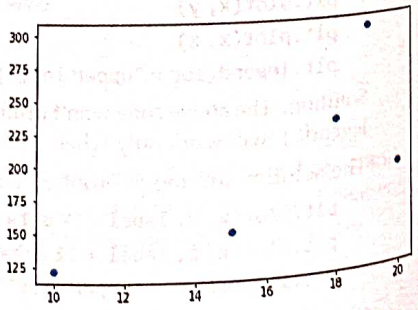
(*a*) ***speed*** and ***meters*** columns of df

(*b*) ***meters*** and ***weight*** columns of df

Solution.

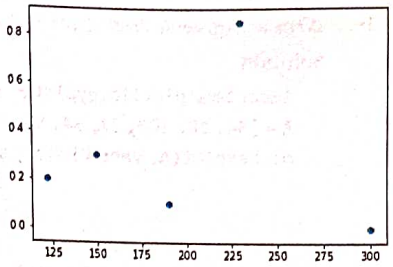
(*a*)

```
import matplotlib.pyplot as pl
:   # df created
pl.scatter(df['speed'], df['meters'])
```



(*b*)

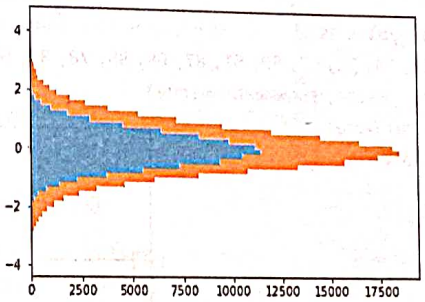
```
import matplotlib.pyplot as pl
:   # df created
pl.scatter( df['meters'], df['weight'])
```



16. *Write a program to create a histogram that plots two ndarrays x and y with 48 bins, in stacked horizontal histogram.*

Solution.

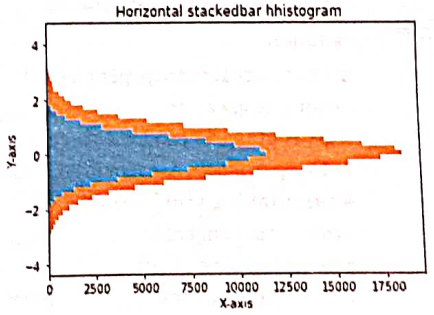
```
import matplotlib.pyplot as pl
:   # df created
pl.scatter( df['meters'], df['weight'])
pl.hist([y,x], bins = 48, orientation = 'horizontal', histtype = 'barstacked')
```



17. *Write code to add plot title and axes titles to above plot.*

Solution.

```
import matplotlib.pyplot as pl
:  # df created
pl.scatter( df['meters'], df['weight'])
pl.hist ([y, x], bins = 48, orientation = 'horizontal', histtype = 'barstacked')
pl.title ("horizontal stackedbar hhistogram")
pl.xlabel ("X-axis")
pl.ylabel ("Y-axis")
```

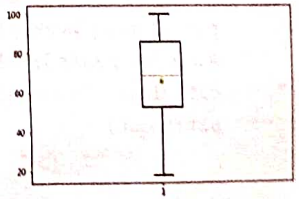


18. *Write a program to create a boxplot from the following set of data :*

34, 18, 100, 27, 54, 52, 93, 59, 61, 87, 68, 85, 78, 82, 91

Solution.

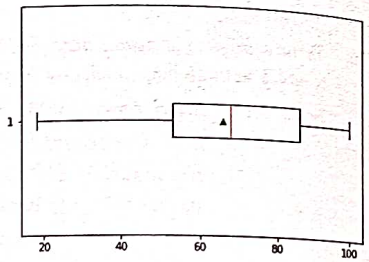
```
import matplotlib.pyplot as pl
A = [34, 18, 100, 27, 54, 52, 93, 59, 61, 87, 68, 85, 78, 82, 91]
pl.boxplot(A, showmeans = True)
```



19. *Write a program to create the same box plot as previous question, but change the orientation to horizontal.*

Solution.

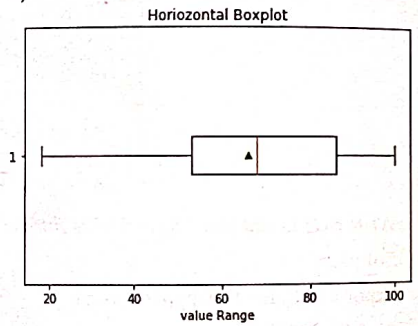
```
import matplotlib.pyplot as pl
A = [34, 18, 100, 27, 54, 52, 93, 59, 61, 87, 68, 85, 78, 82, 91]
pl.boxplot(A, vert = False, showmeans = True)
```



20. *Write a program to add title as "Horizontal Boxplot" and y-axis title as "Value –range" to previous boxplot.*

Solution.

```
import matplotlib.pyplot as pl
A = [34, 18, 100, 27, 54, 52, 93, 59, 61, 87, 68, 85, 78, 82, 91]
pl.boxplot(A, vert = False, showmeans = True)
pl.title("Horizontal Boxplot")
pl.xlabel("value Range")
```

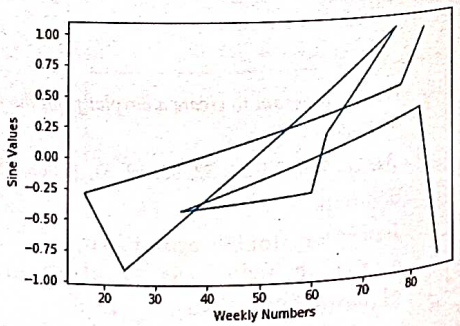


21. *A company randomly generates a weekly winning number (between 0..100) for 10 weeks. An employee of the company wants to plot the sine values of winning numbers (**nump.sin( )** function) on a line chart. Write a program to help him accomplish this.*

Solution.

```
import matplotlib.pyplot as plt
import numpy as np
# generating 10 random integers
ar1 = np.random.randint(100, size = (10,))
# calculating sin() values
sinar = np.sin(ar1)
# plotting line chart
plt.plot(ar1, sinar)
# Set the x axis label of the current axis.
plt.xlabel('Weekly Numbers')
# Set the y axis label of the current axis.
plt.ylabel('Sine Values')
plt.show()
```

Output



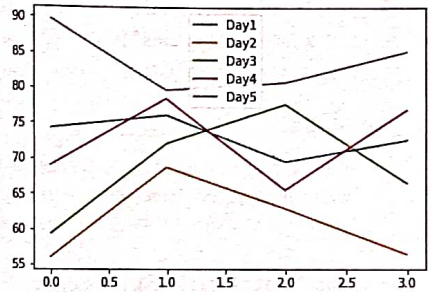
22. *Write a Python program to draw line charts from the given financial data of ABC Co. for 5 days in the form a DataFrame namely fdf as shown below :*

	Day1	Day2	Day3	Day4	Day5
0	74.25	56.03	59.30	69.00	89.65
1	76.06	68.71	72.07	78.47	79.65
2	69.50	62.89	77.65	65.53	80.75
3	72.55	56.42	66.46	76.85	85.08

Solution.

```
import matplotlib.pyplot as plt
import pandas as pd
:   # fdf created or loaded
# it will plot all the numeric columns of fdf
fdf.plot()
```

Output



23. *Consider following DataFrame **prodf**.*

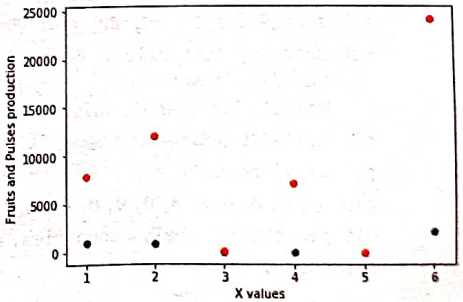
	Fruits	Pulses	Rice	Wheat
Andhra P.	7830.0	931.0	7452.4	NaN
Gujarat	11950.0	818.0	1930.0	2737.0
Kerala	113.1	1.7	2604.8	NaN
Punjab	7152.0	33.0	11586.2	16440.5
Tripura	44.1	23.2	814.6	0.5
Uttar P.	24169.2	2184.4	13754.0	30056.0

*Write a program to plot a scatter chart with the columns **Fruits** and **Pulses**.*

Solution.

```
import matplotlib.pyplot as plt
:  # prodf created or loaded
plt.scatter(x = range(1, len(prodf)+1 ), y = prodf.Pulses)
plt.scatter(x = range(1, len(prodf)+1), y = prodf.Fruits)
plt.xlabel("X values")
plt.ylabel("Fruits and Pulses production")
plt.show()
```

Output

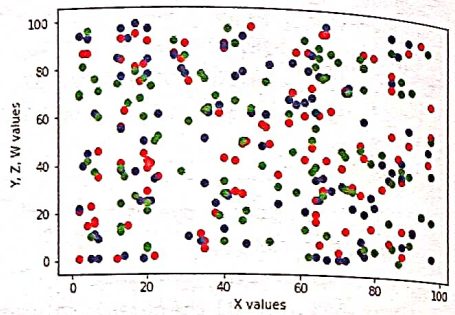


24. *Write a program to plot a scatter graph taking a random distribution in X, Y, Z and W (all three with shape as (100,) having random integers) and plotted against each other.*

Solution.

```
import numpy as np
import matplotlib.pyplot as plt
X = np.random.randint(1, 100, size = (100,))
Y = np.random.randint(1, 100, size = (100,))
Z = np.random.randint(1, 100, size = (100,))
plt.scatter(X,Y, color = 'r')
plt.scatter(X,Z, color = 'b')
plt.xlabel("X values")
plt.ylabel("Y values")
plt.show()
```

Output

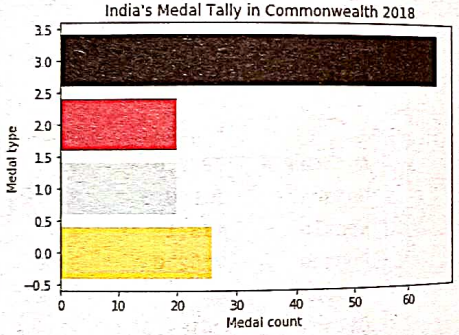


25. *Write a program to create a horizontal bar chart for India's medal tally (from reference table 3.1)*

Solution.

```
import matplotlib.pyplot as plt
Info = ['Gold', 'Silver', 'Bronze', 'Total']
India = [26, 20, 20, 66]
plt.ylabel("Medal type")
plt.xlabel("Medal count")
plt.title("India's Medal Tally in
                Commonwealth 2018")
X = range(len(Info))
plt.barh( X, India, color = ['gold', 'silver', 'brown', 'black'])
plt.show()
```

Output

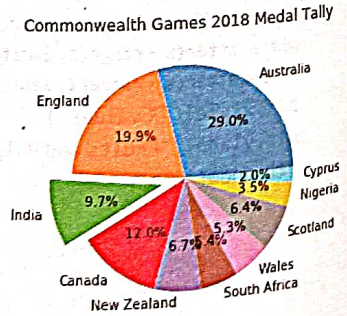


26. *Write a program to create a pie bar chart for India's total medals (reference table 3.1)*

Solution.

```
import matplotlib.pyplot as plt
total = [198, 136 , 66, 82, 46, 37, 36, 44, 24, 14]
countries = ['Australia', 'England', 'India',
    'Canada', 'New Zealand', 'South Africa',
    'Wales', 'Scotland', 'Nigeria', 'Cyprus']
plt.title("Commonwealth Games 2018 Medal Tally")
plt.axis("equal")
expl = [ 0, 0, 0.2, 0, 0, 0, 0, 0, 0, 0]
plt.pie(total, labels = countries, explode = expl, autopct = "%5.1f%%")
plt.show()
```

Output



27. *Prof Awasthi is doing some research in the field of Environment. For some plotting purposes, he has generated some data as :*

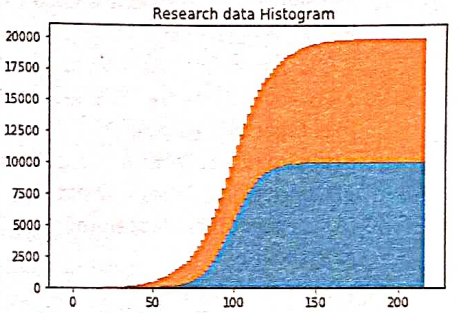
```
mu = 100
sigma = 15
x = mu + sigma * numpy.random.randn(10000)
y = mu + 30 * np.random.randn(10000)
```

Write a program to plot this data on a cumulative bar-stacked horizontal histogram with both x and y.

Solution.

```
import numpy as np
import matplotlib.pyplot as plt
mu = 100
sigma = 15
x = mu + sigma * np.random.randn(10000)
y = mu + 30 * np.random.randn(10000)
plt.hist([x,y], bins = 100, histtype ='barstacked', cumulative = True)
plt.title('Research data Histogram')
plt.show()
```

Output

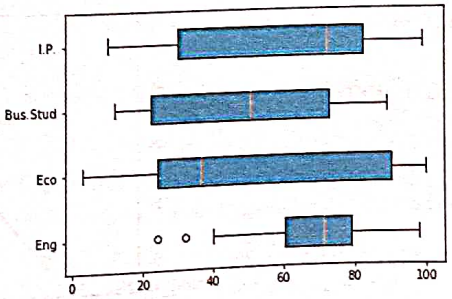


28. *Marks of some students in four different subjects are available in four lists. Write a program to plot this data in a filled horizontal boxplot.*

Solution.

```
import matplotlib.pyplot as plt
Marks1 = [82,76,24,40,67,62,75,78,71,32,98,89,78,67,72,82,87,66,56,52]
Marks2 = [62,5,91,25,36,32,96,95,3,90,95,32,27,55,100,15,71,11,37,21]
Marks3 = [23,89,12,78,72,89,25,69,68,86,19,49,15,16,16,75,65,31,25,52]
Marks4 = [59,73,70,16,81,61,88,98,10,87,29,72,16,23,72,88,78,99,75,30]
Mks_data = [Marks1, Marks2, Marks3, Marks4]
plt.boxplot(Mks_data, vert = 0, patch_artist = True, labels = ['Eng', 'Eco', 'Bus.Stud', 'I.P.'])
plt.show()
```

Output

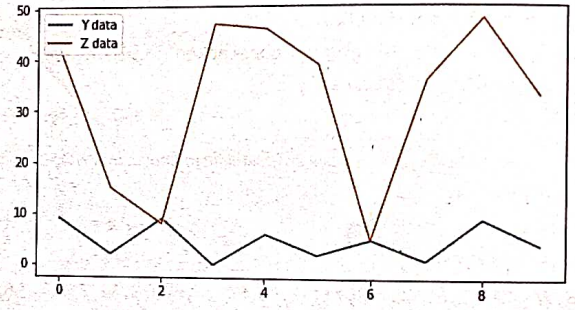


29. *Given two ndarrays. Write a program to plot them in the same line graph and show legends.*

Solution.

```
import matplotlib.pyplot as plt
import numpy as np
# given series of data are y and z to be plotted on Y-axis
x = np.arange(0,10)
y = np.random.randint(10,size = (10,))
z = np.random.randint(50,size = (10,))
plt.figure(figsize = (8, 4))
plt.plot(x, y, label = "Y data")
plt.plot(x, z, label = "Z data")
plt.legend(loc = "upper left")
plt.show()
```

Output



30. *Given four sequences as given below :*

```
X = [1, 2, 3, 4]
Y = [10, 20, 25, 30]
A = [0.3, 3.8, 1.2, 2.5]
B = [11, 25, 9, 26]
```

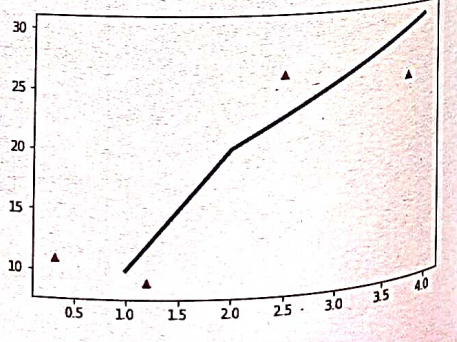
Write a program to plot them in the same chart as :

- *A line graph plotted with X and Y with blue color and having line width as 3.*
- *A scatter graph plotted with A and B with triangular marker of magenta color.*

Solution.

```
import matplotlib.pyplot as plt
X = [1, 2, 3, 4]
Y = [10, 20, 25, 30]
A = [0.3, 3.8, 1.2, 2.5]
B = [11, 25, 9, 26]
plt.plot(X,Y , color = 'b', linewidth = 3)
plt.scatter(A,B, color = 'm', marker = '^')
```

Output



31. *Write a program to create a horizontal bar chart from two data sequences as given below :*

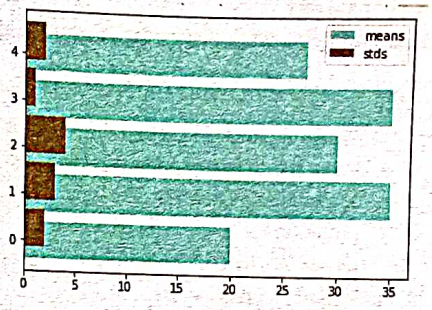
```
means = [20, 35, 30, 35, 27]
stds = [2, 3, 4, 1, 2]
```

Make sure to show legends.

Solution.

```
import matplotlib.pyplot as plt
import numpy as np
means = [20, 35, 30, 35, 27]
stds = [2, 3, 4, 1, 2]
indx = np.arange(len(means))
plt.barh(indx, means, color = 'cyan', label = 'means')
plt.barh(indx + 0.25, stds, color = 'olive', label = 'stds')
plt.legend()
```

Output



32. *In an engineering college, number of admissions stream wise in the current year are :*

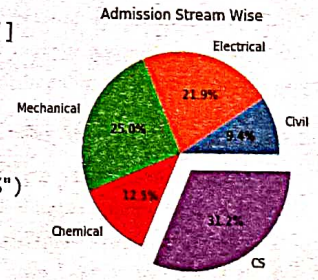
```
Civil = 15, Electrical = 35, Mechanical = 40, Chemical = 20, CS = 50
```

Write a program to print above information on a circular pie chart, creating a wedge for CS stream.

Solution.

```
import matplotlib.pyplot as plt
streams = ['Civil', 'Electrical', 'Mechanical','Chemical', 'CS']
adm = [15, 35, 40, 20, 50]
plt.title("Admission Stream Wise")
plt.axis("equal")
expl= [ 0, 0, 0, 0, 0.2]
plt.pie(adm, labels = streams, explode = expl, autopct = "%5.1f%%")
plt.show( )
```

Output



GLOSSARY

Bar chart	A graphical display of data using bars of different heights.
Bar graph	Bar chart
Data visualization	The graphical or visual representation of information and data using visual elements like charts, graphs, and maps etc.
Line chart	A type of chart which displays information as a series of data points called 'markers' connected by straight line segments.
Line graph	Line chart
Scatter chart	Graph of plotted points on two axes that show the relationship between two sets of data.
Pie chart	A type of graph in which a circle is divided into sectors that each represent a proportion of the whole.
Histogram	A summarisation tool for discrete or continuous data.
Frequency polygon	A type of frequency distribution graph.
Box Plot	The visual representation of the statistical five number summary of a given data set.

Assignment

For selective assignment solutions

Scan QR Code

Type A : Short Answer Questions/Conceptual Questions

1. Name the library of which the PyPlot is an interface.
2. Write the statement to import PyPlot in your script.
3. Name the functions to create the following :
   (*a*) line chart (*b*) bar chart (*c*) horizontal bar chart (*d*) histogram
   (*e*) scatter chart (*f*) boxplot (*g*) pie chart
4. What is a line chart ?
5. What is a scatter chart ?
6. What is a pie chart ?
7. What is a bar chart ?
8. What is a histogram ?
9. What is a boxplot ?
10. What is a frequency polygon ?
11. Name the function to label axes.
12. Name the function to give title to a plot.
13. Name the function to set figure size of a plot.
14. Name the function to set limits for the axes.
15. Name the function to show legends on a plot.
16. Name the function to add ticks on axes.

Type B : Application Based Questions

1. What is the significance of data visualization ?
2. How does Python support data visualization ?
3. What is the use of matplotlib and pyplot ?
4. What are the popular ways of plotting data ?
5. Compare bar( ) and barh( ) functions.
6. What is the role of legends in a graph/chart ?
7. What will happen if you use legend( ) without providing any label for the data series being plotted ?
8. What do you understand by xlimit and ylimit ? How are these linked to data being plotted ?
9. When should you use (*i*) a line chart, (*ii*) a bar chart, (*iii*) a scatter chart, (*iv*) pie chart, (*v*) boxplot ?
10. A list namely *temp* contains average temperatures for seven days of last week. You want to see how the temperature changed in last seven days. Which chart type will you plot for the same and why ?
11. What is histogram ? How do you create histograms in Python ?
12. What are various types of histograms that can be created through hist( ) function ?
13. When should you create histograms and when should you create bar charts to present data visually ?
14. What is cumulative histogram ? How do you create it using PyPlot ?
15. What is frequency polygon ? How do you create it ?
16. What is 5 point summary ?
17. What is Boxplot ? How do you create it in Pyplot ?

Type C : Practical/Knowledge Based Questions

1. Execute the following codes and find out what happens ? (Libraries have been imported already ; plt is the alias name for matplotlib.pyplot)

(*a*)
```
A = np.arange(2, 20, 2)
B = np.log(A)
plt.plot(A, B)
```

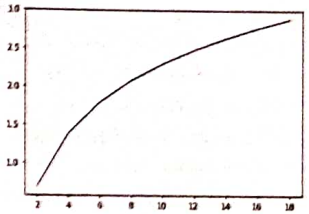
(*b*)
```
A = np.arange(2, 20, 2)
B = np.log(A)
plt.bar(A, B)
```

(*c*)
```
X = np.arange(1, 18, 2.655)
B = np.log(X)
plt.scatter(X, Y)
```

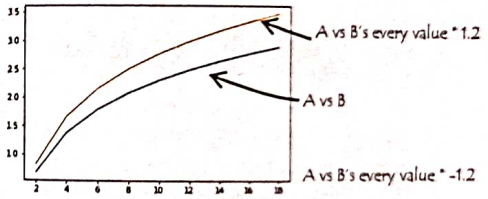
Will any code produce error ? Why/Why not ?

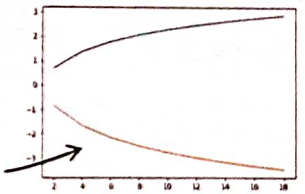
2. Write a program to add titles for the *X*-axis, *Y*-axis and for the whole chart in above codes.

3. **plt.plot(A, B)** produces (A and B are the sequences same as created in question 2) chart as :



Write codes to produce charts as shown below :





4. Given a data frame **df1** as shown below :

	1990	2000	2010
a	52	340	890
b	64	480	560
c	78	688	1102
d	94	766	889

Write code to create :
   (*a*) A scatter chart from the 1990 and 2010 columns of dataframe df1
   (*b*) A line chart from the 1990 and 2000 columns of dataframe df1
   (*c*) Create a bar chart plotting the three columns of dataframe df1
5. The score of four teams in 5 IPL matches is available to you. Write a program to plot these in a bar chart.
6. The score of a team in 5 IPL matches is available to you. Write a program to create a pie chart from this data, showing the last match's performance as a wedge.
7. The prices of a stock for 3 months are given. Write a program to show the variations in prices for each month by 3 lines on same line chart. Make sure to add legends and labels. Show grid also.
8. A distribution data stores about 1000 random number. Write a program to create a scatter chart from this data with varying point sizes.
9. Navya has started an online business. A list stores the number of orders in last 6 months. Write a program to plot this data on a horizontal bar chart.

10. Given the following set of data :

```
Weight measurements for 16 small orders of French-fries (in grams).
78  72  69  81  63  67  65  75
79  74  71  83  71  79  80  69
```

(*a*) Create a simple histogram from the above data
(*b*) Create a horizontal histogram from the above data
(*c*) Create a step type of histogram from the above data
(*d*) Create a cumulative histogram from the above data

11. Create an ndarray containing 16 values and then plot this array along with dataset of previous question in same histogram
(*a*) normal histograms
(*b*) cumulative histograms
(*c*) horizontal histograms

12. Out of above plotted histograms, which ones can be used for creating frequency polygons ? Can you draw frequency polygons from all the above histograms ?

13. Create/draw frequency polygon from the data used in above questions.

14. From the following ordered set of data :

```
63, 65, 67, 69, 69, 71, 71, 72, 74, 75, 78, 79, 79, 80, 81, 83
```

(*a*) Create a horizontal boxplot
(*b*) Create a vertical boxplot
(*c*) Show means in the boxplot
(*d*) Create boxplot without the box

15. Sina has created ordered set of data from the number of new customers registered on his online service centre in last 20 months.
Write a program to plot this data on a filled boxplot with means shown.

4 Importing/Exporting Data between CSV Files/MySQL and Pandas

Outline

4.1 Introduction
4.2 Transferring Data between .csv Files and DataFrames
4.3 Transferring Data Between Dataframes and MySQL
4.4 Exporting a Dataframe's Data as a Table in MySQL Database

4.1 Introduction

Previous chapters introduced Python Pandas library to you and you learnt about two datastructures of Pandas library — *Series* and *DataFrame*. You also learnt that dataframes are capable of storing any types of data in 2D tabular form. Most data files that we use to store data such as spreadsheet files or database tables, also store the data in 2D tabular formats. Since dataframes can also hold data in similar way, you can transfer data from dataframe to such data files (or tables) or from files (or tables) into dataframes. This chapter is dedicated to the discussion of the same. In this chapter, you shall learn how to transfer data to/from a .CSV file (.CSV is a format that stores data in comma separated form – Comma Separated Values) from/into a dataframe and also to/from a database table from/into a dataframe.

4.2 Transferring Data between .csv Files and DataFrames

The acronym **CSV** is short for **Comma-Separated Values**. The CSV format refers to a tabular data that has been saved as plaintext where data is separated by commas. For example, consider a table having some data as shown below. If you store this table's data in CSV format then the data of the table will be stored in CSV format as shown below on the right.

Roll No.	Name	Marks
101	Tia	67.8
102	Radha	78.9

Tabular data

When converted to CSV format

```
Roll No.,Name,Marks
101,Tia,67.8
102,Radha,78.9
```

CSV Data

As you can see that in CSV format :

- Each row of the table is stored in one row *i.e.*, the number of rows in a CSV file are equal to number of rows in the table (or sheet or database table etc.)
- The field-values of a row are stored together with commas after every field value ; but after the last field's value in a line/row, no comma is given, just the end of line.

The CSV format is popular as it offers following advantages :

- A simple, compact and ubiquitous format for data storage.
- A common format for data interchange.
- It can be opened in popular spreadsheet packages like *MS-Excel, Calc* etc.
- Nearly all spreadsheets and databases support import / export to csv format.

CSV

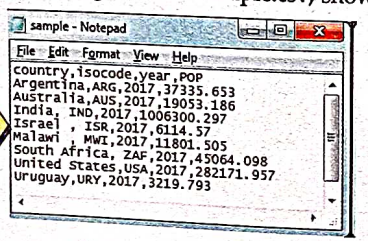
"The acronym CSV is short for Comma-Separated Values, which refers to a tabular data saved as plaintext where data values are separated by commas."

4.2.1 Loading Data from CSV to Dataframes

Python's Pandas library offers two functions **read_csv( )** and **to_csv( )** that help you bring data from a CSV file into a dataframe and write a dataframe's data to a CSV file. Let us learn how to do this.

Suppose we have following csv file – **sample.csv**, shown in notepad as well as in MS Excel.

Sample data in a CSV file shown in *Notepad* as well as in *MS Excel*



	A	B	C	D
1	country	isocode	year	POP
2	Argentina	ARG	2017	37335.65
3	Australia	AUS	2017	19053.19
4	India	IND	2017	1006300
5	Israel	ISR	2017	6114.57
6	Malawi	MWI	2017	11801.51
7	South Afri	ZAF	2017	45064.1
8	United Sta	USA	2017	282172
9	Uruguay	URY	2017	3219.793

Figure 4.1 Sample CSV file (name : sample.csv)

(You can create a CSV file yourself by saving data of an MS-Excel file in CSV format using **Save AS** command from **File** tab/menu and selecting **Save As Type as CSV Format**).

Before doing anything with Pandas, like always, make sure to import the Pandas library by giving command :

```
import pandas as pd
```


4.2.1A Reading From a CSV File to Dataframe

You can use **read_csv( )** function to read data from a CSV file in your dataframe by using the function as per following syntax :

```
<DF> = pandas.read_csv(<filepath>)
```

That is, if you give following command after importing pandas library (note that our sample file is stored in c:\data folder) :

```
df = pd.read_csv("c:\\data\\sample.csv")
```

Read data into dataframe df from the CSV file whose path is given here as a string

Now if you print dataframe **df**, Python will show :

	country	isocode	year	POP
0	Argentina	ARG	2017	37335.653
1	Australia	AUS	2017	19053.186
2	India	IND	2017	1006300.297
3	Israel	ISR	2017	6114.570
4	Malawi	MWI	2017	11801.505
5	South Africa	ZAF	2017	45064.098
6	United States	USA	2017	282171.957
7	Uruguay	URY	2017	3219.793

```
>>> df
          country isocode  year        POP
0       Argentina     ARG  2017  37335.653
1       Australia     AUS  2017  19053.186
2           India     IND  2017  1006300.297
3          Israel     ISR  2017  6114.570
4          Malawi     MWI  2017  11801.505
5    South Africa     ZAF  2017  45064.098
6   United States     USA  2017  282171.957
7         Uruguay     URY  2017  3219.793
```

Compare the dataframe **df**'s data with the sample CSV file's data shown in figure 4.1. As you can see that it has taken the first row from the CSV file as the column names for dataframe **df**.

NOTE

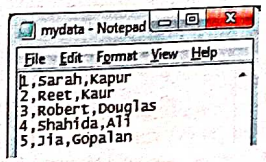
While giving filepaths, sometimes error is reported with path even though you have given the right path. For that you may give file paths with double slashes *i.e.*, in place of c:\data\myfile.csv, write c:\\data\\myfile.csv.

4.2.1B Reading CSV File and Specifying Own Column Names

But you may have a CSV file that does not have top row containing column headers (as shown below)

See, it only has data and no column names.

CSV file (mydata.csv)



	A	B	C
1	1	Sarah	Kapur
2	2	Reet	Kaur
3	3	Robert	Douglas
4	4	Shahida	Ali
5	5	Jia	Gopalan

Now if you read such a file by just giving the filepath, it will take the top row as the column headers , *e.g.*, as shown below :

```
df2 = pd.read_csv("c:\\data\\mydata.csv")
```

Now if you print dataframe **df2**, Python will show :

	1	Sarah	Kapur
0	2	Reet	Kaur
1	3	Robert	Douglas
2	4	Shahida	Ali
3	5	Jia	Gopalan

```
>>> df2 = pd.read_csv ("c:\data\mydata.csv")
>>> df2
   1     Sarah    Kapur
0  2      Reet     Kaur
1  3    Robert  Douglas
2  4   Shahida      Ali
3  5       Jia  Gopalan
```

See, it has taken first row (which is actually data) as column headings

NOT WANTED this way

BUT this is not what we wanted. The top row (*1, Sarah, Kapur*) is data, not column headings. So what to do ? For such a situation, you can specify own column headings in **read_csv( )** using **names** argument as per following syntax :

```
<DF> = pandas.read_csv(<filepath>, names = <sequence containing column names>)
```

That is, you need to modify above statement as :

```
df2 = pd.read_csv("c:\\data\\mydata.csv", names = ["Roll no", "First_Name", "Last_Name"])
```

And now when you print **df2**, Python will show :

```
>>> df2 = pd.read_csv ("c:\data\mydata.csv", \
        name = ["Roll no", First_Name", "Last_Name"])
>>> df2
   Roll no First_Name Last_Name
0        1      Sarah     Kapur
1        2       Reet      Kaur
2        3     Robert   Douglas
3        4    Shahida       Ali
4        5        Jia   Gopalan
```

This time it has not taken first row as headings. First row remains data and the column headings are as per **names** sequence

If you want the first row not to be used as header and at the same time you do not want to specify column headings rather go with default column headings which go like 0, 1, 2, ,3..., then simply give argument as **header = None** in **read_csv( )**, *i.e.*, as :

```
<DF> = pandas.read_csv( <filepath>, header = None)
```

For example, the following statement will not consider row 1 for column headings and default column headings will be taken.

```
df3 = pd.read_csv("c:\\data\\mydata.csv", header = None)
```

The dataframe **df3** will be storing data as shown below:

```
>>> df3 = pd.read_csv ("c:\data\mydata.csv", header = None)
>>> df3
   0        1         2
0  1    Sarah     Kapur
1  2     Reet      Kaur
2  3   Robert   Douglas
3  4  Shahida       Ali
4  5      Jia   Gopalan
```

See, this time first row has not been taken as column headings, rather is taken as data.

Now comes a situation where you have first row of CSV file storing some column headings but you do not want to use them, *e.g.*, as shown below :

One	Two	Three
1	Sarah	Kapur
2	Reet	Kaur
3	Robert	Douglas
4	Shahida	Ali
5	Jia	Gopalan

In that case, in **read_csv( )** if you simply give

(*i*) Argument names = <sequence of column headings> then the first row's data will be taken as data, *e.g.*,

```
>>> df4 = pd.read_csv("c:\\data\\mydata.csv", names = ["Rollno", "Name", "marks"])
```

	Rollno	Name	Marks
0	One	Two	Three
1	1	Sarah	Kapur
2	2	Reet	Kaur
3	3	Robert	Douglas
4	4	Shahida	Ali
5	5	Jia	Gopalan

See, the row one is now taken as data WHICH YOU DID NOT WANT

(*ii*) Or if you give argument as **header = None**, then also it will take default headings as 0, 1, 2, ..., which you do not want

```
>>> df3 = pd.read_csv("c:\\data\\mydata.csv", header = None)
>>> df3
```

	0	1	2
0	1	Sarah	Kapur
1	2	Reet	Kaur
2	3	Robert	Douglas
3	4	Shahida	Ali
4	5	Jia	Gopalan

See, the row one is not taken as column headings, BUT default column headings given, WHICH YOU DID NOT WANT

The solution to this problem is to find a way so that it skips row 1 while fetching data from CSV file. For this, you now need to give two arguments along with file path : one for column headings *i.e.*, **names = <column headings sequence>** and another **skiprows = <n>**, *i.e.*,

```
<DF> = pandas.read_csv(<filepath>, names = <column names' sequence >, skiprows = <n>)
```

The number given with skip rows tells how many rows to be skipped from CSV while reading data.

Skiprows argument can either take a number for number of rows to be skipped from beginning or it take a list of rows numbers to be skipped, *e.g.*, to skip 3rd, 5th, 7th row, write,

```
skiprows = [3, 5, 7]
```

Thus, if you give statement as given below :

```
>>> df5 = pd.read_csv("c:\\data\\mydata.csv", names = ["Rollno", "Name", "Marks"], skiprows = 1)
>>> df5
```

	Rollno	Name	Marks
0	1	Sarah	Kapur
1	2	Reet	Kaur
2	3	Robert	Douglas
3	4	Shahida	Ali
4	5	Jia	Gopalan

See, this time it has skipped 1 row and taken data from 2nd row onwards. Also, the columns have names as per your choice

BUT this is not what we wanted. The top row (*1, Sarah, Kapur*) is data, not column headings. So what to do ? For such a situation, you can specify own column headings in **read_csv( )** using **names** argument as per following syntax :

```
<DF> = pandas.read_csv(<filepath>, names = <sequence containing column names>)
```

That is, you need to modify above statement as :

```
df2 = pd.read_csv("c:\\data\\mydata.csv", names = ["Roll no", "First_Name", "Last_Name"])
```

And now when you print **df2**, Python will show :

```
>>> df2 = pd.read_csv ("c:\data\mydata.csv", \
        name = ["Roll no", First_Name", "Last_Name"])

>>> df2
   Roll no First_Name Last_Name
0        1      Sarah     Kapur
1        2       Reet      Kaur
2        3     Robert   Douglas
3        4    Shahida       Ali
4        5        Jia   Gopalan
```

This time it has not taken first row as headings. First row remains data and the column headings are as per **names** sequence

If you want the first row not to be used as header and at the same time you do not want to specify column headings rather go with default column headings which go like 0, 1, 2, ,3..., then simply give argument as **header = None** in **read_csv( )**, *i.e.*, as :

```
<DF> = pandas.read_csv( <filepath>, header = None)
```

For example, the following statement will not consider row 1 for column headings and default column headings will be taken.

```
df3 = pd.read_csv("c:\\data\\mydata.csv", header = None)
```

The dataframe **df3** will be storing data as shown below:

```
>>> df3 = pd.read_csv ("c:\data\mydata.csv", header = None)

>>> df3
   0        1        2
0  1    Sarah    Kapur
1  2     Reet     Kaur
2  3   Robert  Douglas
3  4  Shahida      Ali
4  5      Jia  Gopalan
```

See, this time first row has not been taken as column headings, rather is taken as data.

Now comes a situation where you have first row of CSV file storing some column headings but you do not want to use them, *e.g.*, as shown below :

One	Two	Three
1	Sarah	Kapur
2	Reet	Kaur
3	Robert	Douglas
4	Shahida	Ali
5	Jia	Gopalan

In that case, in **read_csv( )** if you simply give

(*i*) Argument names = <sequence of column headings> then the first row's data will be taken as data, *e.g.*,

```
>>> df4 = pd.read_csv("c:\\data\\mydata.csv", names = ["Rollno", "Name", "marks"])
```

	Rollno	Name	Marks
0	One	Two	Three
1	1	Sarah	Kapur
2	2	Reet	Kaur
3	3	Robert	Douglas
4	4	Shahida	Ali
5	5	Jia	Gopalan

See, the row one is now taken as data WHICH YOU DID NOT WANT

(*ii*) Or if you give argument as **header = None**, then also it will take default headings as 0, 1, 2, ..., which you do not want

```
>>> df3 = pd.read_csv("c:\\data\\mydata.csv", header = None)
>>> df3
```

	0	1	2
0	1	Sarah	Kapur
1	2	Reet	Kaur
2	3	Robert	Douglas
3	4	Shahida	Ali
4	5	Jia	Gopalan

See, the row one is not taken as column headings, BUT default column headings given, WHICH YOU DID NOT WANT

The solution to this problem is to find a way so that it skips row 1 while fetching data from CSV file. For this, you now need to give two arguments along with file path : one for column headings *i.e.*, **names = <column headings sequence>** and another **skiprows = <n>**, *i.e.*,

```
<DF> = pandas.read_csv(<filepath>, names = <column names' sequence >, skiprows = <n>)
```

The number given with skip rows tells how many rows to be skipped from CSV while reading data.

Skiprows argument can either take a number for number of rows to be skipped from beginning or it take a list of rows numbers to be skipped, *e.g.*, to skip 3rd, 5th, 7th row, write,

```
skiprows = [3, 5, 7]
```

Thus, if you give statement as given below :

```
>>> df5 = pd.read_csv("c:\\data\\mydata.csv", names = ["Rollno", "Name", "Marks"], skiprows = 1)
>>> df5
```

	Rollno	Name	Marks
0	1	Sarah	Kapur
1	2	Reet	Kaur
2	3	Robert	Douglas
3	4	Shahida	Ali
4	5	Jia	Gopalan

See, this time it has skipped 1 row and taken data from 2nd row onwards. Also, the columns have names as per your choice

Following examples illustrate the loading of a csv file in a dataframe, with or without the header row.

EXAMPLE 1 *Write a program to read from a CSV file Employee.csv and create a dataframe from it.*

SOLUTION

```
import pandas as pd
edf = pd.read_csv("c:\\pywork\\Employee.csv")
print(edf)
```

Output

	Empno	Name	Designation	Salary
0	1001	Trupti	Manager	56000
1	1002	Raziya	Manager	55900
2	1003	Simran	Analyst	35000
3	1004	Silviya	Clerk	25000
4	1005	Suji	PR Officer	31000

File *Employee.csv*

```
Empno,Name,Designation,Salary
1001,Trupti,Manager,56000
1002,Raziya,Manager,55900
1003,Simran,Analyst,35000
1004,Silviya,Clerk,25000
1005,Suji,PR Officer,31000
```

EXAMPLE 2 *Write a program to read from a CSV file Employee.csv and create a dataframe from it but dataframe should not use file's column header rather should use own column numbers as 0, 1, 2 and so on.*

SOLUTION

```
import pandas as pd
edf = pd.read_csv("c:\\pywork\\Employee.csv", header = None, skiprows = 1 )
print(edf)
```

Output

	0	1	2	3
0	1001	Trupti	Manager	56000
1	1002	Raziya	Manager	55900
2	1003	Simran	Analyst	35000
3	1004	Silviya	Clerk	25000
4	1005	Suji	PR Officer	31000

See, the columns have numbers this time

Get Dataframe Index Lables from csv

If you want to use a column's coming from a csv file as the index labels for the dataframe, you may specify the column name in **index_col** argument, *e.g.*,

```
pd.read_csv("c:\\pywork\\Employee.csv", index_col = 'Empno')
```

Following example is doing the same. Carefully have a look at its output.

EXAMPLE 3 *Write a program to read from a CSV file Employee.csv and create a dataframe from it. Use Empno column as index labels.*

SOLUTION

```
import pandas as pd
edf = pd.read_csv("c:\\pywork\\Employee.csv", index_col = "Empno")
print(edf)
```

Output

Empno	Name	Designation	Salary
1001	Trupti	Manager	56000
1002	Raziya	Manager	55900
1003	Simran	Analyst	35000
1004	Silviya	Clerk	25000
1005	Suji	PR Officer	31000

The values of column Empno are now used to define index lables of the dataframe.

EXAMPLE 4 *Write a program to read from a CSV file Employee.csv and create a dataframe from it but dataframe should not use file's column header rather should use own column headings as EmpID, EmpName, Designation and Salary. Also print the maximum salary given to an employee.*

SOLUTION

```
import pandas as pd
edf = pd.read_csv("c:\\pywork\\Employee.csv",\
        names = [ 'EmpID', 'EmpName', 'Designation', 'Salary'], skiprows = 1)
print(edf)
print("Maximum salary is", edf.Salary.max( ) )
```

Output

	EmpID	EmpName	Designation	Salary
0	1001	Trupti	Manager	56000
1	1002	Raziya	Manager	55900
2	1003	Simran	Analyst	35000
3	1004	Silviya	Clerk	25000
4	1005	Suji	PR Officer	31000

Maximum salary is 56000

EXAMPLE 5 *Following program code is creating a dataframe from compresult.csv file shown below :*

```
import pandas as pd
cdf = pd.read_csv("c:\\pywork\\compresult.csv", header = None , skiprows = 1 )
print("Maximum Points are", cdf.Points.max( ) )
```

But the program code is giving the following error when it tries to print the maximum number of points from the dataframe.

```
AttributeError: 'DataFrame' object has no attribute 'Points'
```

Find out the error/problem and rectify it.

File *Compresult.csv*

```
Name,Points,Rank
Shradha,4500,23
Nishchay,4800,31
Ali,4500,25
Adi,5100,14
```

SOLUTION

The problem with the above code is when argument **header = None** is used in **read_csv( )** then dataframe is given own column numbers which are 0, 1, 2. Thus the second column is now identified as **cdf[1]** and not as **cdf.Points** as used in the code.

So, if we change the last line of the code as follows, then program will run and give the output as shown below :

```
print("Maximum Points are", cdf[1].max() )
```

Output with corrected code will be :

```
   0          1     2
0  Shradha    4500  23
1  Nishchay   4800  31
2  Ali        4500  25
3  Adi        5100  14
Maximum Points are 5100
```

4.2.1C Reading Specified Number of Rows from CSV File

Giving argument **nrows = <n>** in **read_csv( )**, will read the specified number of rows from the CSV file, *e.g.*, see below :

```
>>> df6 = pd.read_csv("c:\\data\\mydata.csv", names = ["Rollno", "Name", "Surname"], nrows = 3)
>>> df6
```

	Rollno	Name	Surname
0	1	Sarah	Kapur
1	2	Reet	Kaur
2	3	Robert	Douglas

Because of argument **nrows = 3**, only three rows have been read from CSV file whose data is shown on the right

	A	B	C
1	1 Sarah	Kapur	
2	2 Reet	Kaur	
3	3 Robert	Douglas	
4	4 Shahida	Ali	
5	5 Jia	Gopalan	

Using **nrows** argument, you can extract *top rows*, *bottom rows* and *middle rows* too if you use it with head( ) or tail( ) functions, as our following example is doing.

EXAMPLE **6** *Consider the Employee.csv file used in example 1. Write a program to read the records from this file in dataframe and print the middle row stored in the csv file.*

SOLUTION

We shall load the top three rows in a dataframe (as there are only five rows in the dataframe) and then display the bottom row from the dataframe, which will be the *middle row* we require.

```
import pandas as pd
empdf = pd.read_csv("c:\\pywork\\employee.csv", nrows = 3)
print("Middle row of employee.csv is :")
print(empdf.tail(1) )
```

Output

```
Middle row of employee.csv is :
      Empno   Name    Designation  Salary
2     1003    Simran     Analyst   35000
```


4.2.1D Reading from CSV files having Separator Different from Comma

Some CSV files are so created that their separator character is different from comma such as a semicolon(;) or a pipe symbol(|) etc. To read data from such CSV files, you need to specify an additional argument as **sep = <separator character>**. If you skip this argument then default separator character (comma) is considered. For example, following command will read data from a CSV file where data has been separated using semicolons :

```
dfNew = pd.read_csv("c:\\data\\match.csv", sep = ';', names = ["Country1", "Stat", "Country2"])
```

```
>>> dfNew
   Country1   Stat  Country2
0  Denmark    beat      Peru
1  Mexico     beat   Germany
2  Australia  draw   Denmark
```

Data separator is ; in place of comma

match - Notepad

File Edit Format View Help

```
Denmark; beat; Peru
Mexico; beat ; Germany
Australia; draw; Denmark
```

EXAMPLE **7** *Write a program to read from **sport.csv** file (shown below) where the separator is a tab character.*

SOLUTION

```
import pandas as pd
sdf = pd.read_csv("c:\\pywork\\sport.csv", sep = '\t')
print( sdf )
```

File *Sport.csv*

Sport	Competitions	Prizes won
Tennis	14	9
Football	22	16
Chess	25	15

Output

```
   Sport      Competitions  Prizes won
0  Tennis     14            9
1  Football   22            16
2  Chess      25            15
```

Thus, we can summarize **read_csv( )**[1] as follows :

```
<DF> = pandas.read_csv( <filepath>, sep = <separator character>, \
names = <column names sequence>, index_col = <column name> \
header = None, skiprows = <n>, nrows =<n>)
```

4.2.2 Storing Dataframe's Data to CSV File

Sometimes, we have data available in dataframes and we want to save that data in a CSV file. For this purpose, Python Pandas provides **to_csv( )** function that saves the data of a dataframe in a CSV file.

Function **to_csv( )** can be used as per following syntaxes :

```
<DF>.to_csv(<filepath>)
```

Or

```
<DF>.to_csv(<filepath>, sep =<separator_character>)
```

[1] The separator character must be a one character string only.

Repeating again that every function contains lot of functionality, covering all of it is not possible as the aim of this not to be a manual but to explain concepts as per syllabus.

For instance, if you have a dataframe namely **df7** storing following data :

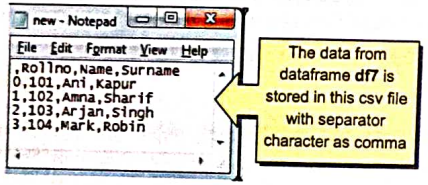
	Rollno	Name	Surname
0	101	Ani	Kapur
1	102	Amna	Sharif
2	103	Arjan	Singh
3	104	Mark	Robin

To store this data of **df7**, you can write :

```
>>> df7.to_csv("c:\\data\\new.csv")
```

As no separator has been mentioned, it will take default separator character, comma, to separate data in the CSV file. Also, even if there exists a file with the same name at the given location, it will simply overwrite the file without any warning.

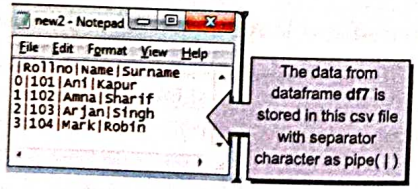
Now you can check at the given path and you will find a file with the **new.csv** there. Open the file and you will find the data of dataframe **df7** there – comma separated.



Let us save the same dataframe's data in another file namely **new2.csv** but with separator as '|' character by giving following command :

```
df7.to_csv("c:\\data\\new2.csv", sep = "|")
```

You can open file **new2.csv** in Notepad and it will look like :



EXAMPLE 8 *Consider the dataframe* ***allDf*** *as shown below.*

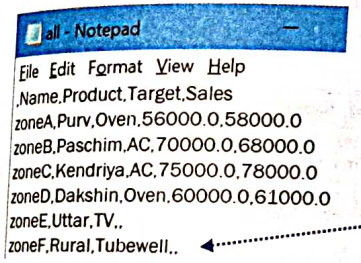
	Name	Product	Target	Sales
zoneA	Purv	Oven	56000.0	58000.0
zoneB	Paschim	AC	70000.0	68000.0
zoneC	Kendriya	AC	75000.0	78000.0
zoneD	Dakshin	Oven	60000.0	61000.0
zoneE	Uttar	TV	NaN	NaN
zoneF	Rural	Tubewell	NaN	NaN

Write a program to export the contents of this dataframe to a CSV file.

SOLUTION

```
import pandas as pd
:       # dataframe allDf created or loaded
# writing to csv file
allDf.to_csv("c:\\pywork\\all.csv")
```

Output (all.csv file)

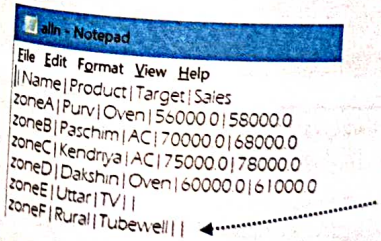


EXAMPLE 9 *Write a program to store the* ***allDf*** *dataframe of previous example in file alln.csv, with a separator character '|'.*

SOLUTION

```
import pandas as pd
:       # dataframe allDf created or loaded
# writing to csv file
allDf.to_csv("c:\\pywork\\alln.csv", sep = '|')
```

Output (alln.csv file)



4.2.2A Handling NaN Values with to_csv( )

Sometimes, your dataframe has some missing values. For instance, execute following two statements to have missing values in your dataframe. (You must have imported **NumPy** library as **np** prior to executing following statements)

```
df7.loc[3, "Name"] = np.NaN
df7.loc[0, "Surname"] = np.NaN
```

Now your dataframe df7 will store data as :

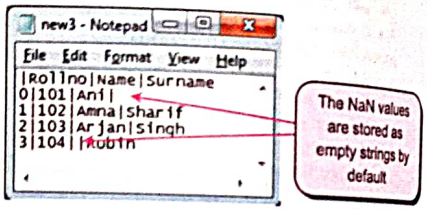
	Rollno	Name	Surname
0	101	Ani	NaN
1	102	Amna	Sharif
2	103	Arjan	Singh
3	104	NaN	Robin

See, the dataframe df7 has two NaN values in rows 0 and 3 respectively

Now, if you store this dataframe in a CSV file by giving following command :

```
df7.to_csv("c:\\data\\new3.csv", sep = "|")
```

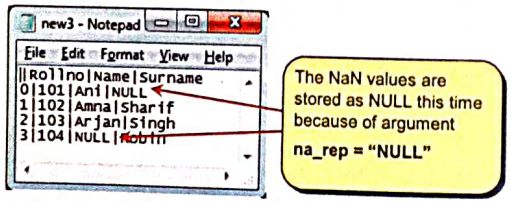
By default, the missing/NaN values are stored as empty strings in CSV file. The file *new3.csv* will look like as shown on the right.



You can specify your own string that can be written for missing/NaN values by giving an argument **na_rep = <string>**, *i.e.*, following statement will write NULL in place of NaN values in the CSV file :

```
df7.to_csv("c:\\data\\new3.csv", sep = "|", na_rep = "NULL")
```

That is, now the file *new3.csv* will look like :



NOTE

Note that the function read_csv( ) is pandas' function (carefully read its syntax) and to_csv( ) is dataframe structure's function (carefully read its syntax).

Thus, we can summarise **to_csv( )** as follows where **sep** and **na_rep** arguments are optional :

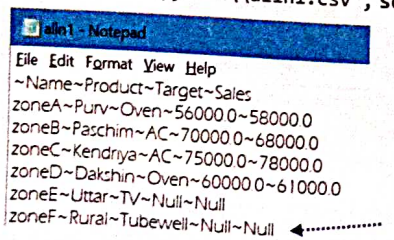
```
<DF>.to_csv(<filepath>, sep = <separator character>, na_rep =<string>)
```

EXAMPLE **10** *Previous examples 8 and 9 created csv files without NaN values. Write a program to store the data of allDf dataframe in a csv file along with NaN values stored as Null and separator as '~'.*

SOLUTION

```
import pandas as pd
:          # dataframe allDf created or loaded
# writing to csv file
allDf.to_csv("c:\\pywork\\alln1.csv", sep = '~', na_rep= 'Null')
```

Output (alln1.csv file)



See, the created CSV file stores values with '~' separator and all NaN values as Null

4.3 Transferring Data Between Dataframes and MySQL

An SQL database is a relational database having data in tables called relations and it uses a special type of query language, *Structured Query Language* (SQL), to query upon manipulate data or to communicate with database. There are many SQL databases available such as *MySQL, SQL Server, SQLite* etc. You have already learnt to work with MySQL databases in your previous class. In this section, you shall learn to import/export data from MySQL database to a dataframe in your Python program and vice versa.

NOTE

Please note that there are multiple ways of connecting with MySQL from Python. We shall cover two of these : using MySQL connector, which is provided by Oracle (the MySQL company) and by using pymysql, which is provided by Python itself.

Installing *mysql connector or pymysql packages*

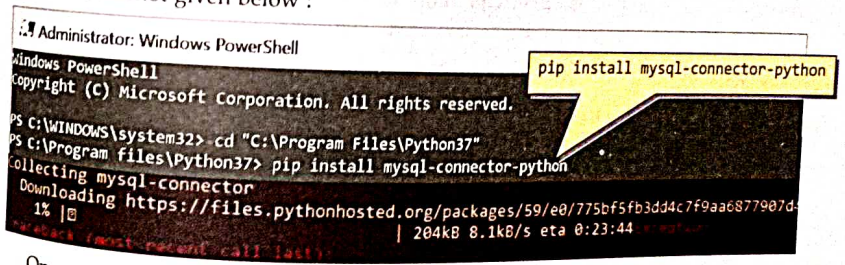
In order to connect with MySQL from with a Python program, you must have the **mysql connector** package (**mysql-connector-python**) or **pymysql** installed. For this, you can open the command shell on your operating system and go to the Python installation folder by giving command :

```
C:\WINDOWS\system32> cd <Python folder path>
```

And then type the following command on the prompt to install **mysql-connector-python** or **pymysql** package :

```
C:\<path> > pip install mysql-connector-python
c:\<path> > pip install pymysql
```

See screenshot given below :



Once you have *mysql-connector-python* or *pymysql* package installed, you can import/export data from a MySQL database into your Python program in a dataframe. Following section will talk about how you can do it programmatically.

In section 4.3, we are using **mysql-connector-python** to connect with MySQL and in section 4.4, we shall use **pymysql** to connect with MySQL.

4.3.1 Bringing Data from MySQL Database into a Dataframe

In order to read data from a MySQL database from with your python program in a dataframe, you need to follow the steps given below. There are mainly seven steps that must be followed in order to create a database connectivity application.

Step 1 Start Python and import the packages required for database programming.

Step 2 Open a connection to database.

Step 3 Execute sql command and fetch rows into a dataframe.

Step 4 Process as desired.

Step 5 Close the connection

Step 1 : Import Required Libraries

Firstly, you should import the **mysql.connector** library along with importing **pandas** library *i.e.*, write the following commands in your program.

```
import pandas as pd
import mysql.connector as sqltor
```

*Like earlier, you can give any alias name ; we have given alias as **sqLtor***

Step 2 : Open a Connection to MySQL Database

Next you need to establish connection to a MySQL database using **connect( )** function of **mysql.connector** package.

The **connect( )** function of **mysql.connector** establishes connection to a MySQL database and requires four parameters, which are :

```
<Connection-Object> = mysql.connector.connect (host = <host-name> , user = <username>,
                                               passwd = <password> [ , database = <database>])
```

- *user* is the username on MySQL
- *password* is the password of the user
- *host-name* the database server hostname or IP address
- *database* is optional which provides the database name of a MySQL database.

Specify a Python identifier as **<connection object>** name. You shall be using this name for executing queries and for other tasks on connected database.

For example :

loginid and password of your MySQL database ; it should be correct password

The connection object

```
import mysql.connector as sqltor
mycon = sqltor.connect(host = "localhost", user = "root", passwd = "MyPass",
                       database = "test")
```

a MySQL database

Imp. *Make sure that this is the name of an existing database*

Since we imported **mysql.connector** package as **sqltor**, we are using the name **sqltor** in place of **mysql.connector**. If you have imported the package as

```
import mysql.connector
```

Then you need to write above function as :

```
mycon = mysql.connector.connect(host = "localhost", user = "root",
                                passwd = "MyPass", database = "test")
```

The above command will establish connection to MySQL database with **user as "root"**, **password as "MyPass"** and to the **MySQL database namely test** which exists on the MySQL.

Here, you must ensure that the **user** and **password** that you specify are valid *user and password* for your local MYSQL installation.

If Python reports no error, it means you have successfully connected to MySQL database. You can also check for successful connection using function **is_connected( )** with connected object (which returns *True*, if connection is successful, *i.e.*, if you may write following additional code to check if the connection is successful or not)

CONNECTION OBJECT

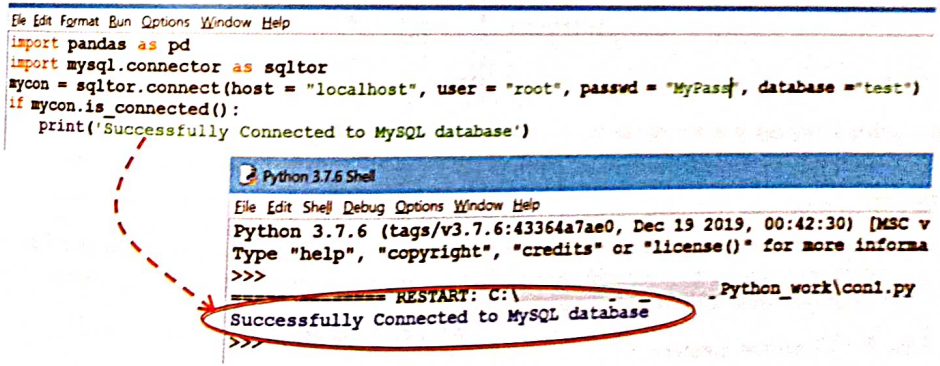
A database **Connection object** controls the connection to the database. It represents a unique session with a database connected from within a script/program.

The same connection object with which we connected to MySQL database

```
if mycon.is_connected():
    print('Successfully Connected to MySQL database')
```

Following screenshot shows you the output produced by the code shown below :

```
import pandas as pd
import mysql.connector as sqltor
mycon = sqltor.connect(host = "localhost", user = "root", passwd = "MyPass", database ="test")
if mycon.is_connected():
    print('Successfully Connected to MySQL database')
```



Step 3 : Execute SQL command and fetch rows in a Dataframe

Once the connection to SQL database is established, read data from a table into a dataframe using **read_sql( )** as per the following syntax :

```
<DF> = pandas.read_sql("<SQL statement>", <connection name>)
```

That is, if you give following command, it will read all the records from table **Student** of database **test** (to which we have established connection with name **mycon**).

This is the connection that we created in step 2

The SQL query statement must and with a semicolon.

```
>>> df = pd.read_sql("SELECT * From Student ;", mycon)
```

Here, make sure that the SQL statement given inside the **read_sql( )** function :

(*i*) must end with a semicolon and

(*ii*) should be enclosed in quotes.

Now the above full code will give output as shown below :

```
*con1.py - C:\Pywork\con1.py (3.7.6)*
File Edit Format Run Options Window Help
import pandas as pd
import mysql.connector as sqltor
mycon = sqltor.connect(host = "localhost",\
        user = "root", passwd = "MyPass", database ="test")
if mycon.is_connected():
    df1 = pd.read_sql("Select * from student ; ", mycon)
    print(df1)
else:
    print("MySQL Connection problem")
```

Result of given SQL statement

```
Python 3.7.6 Shell
File Edit Shell Debug Options Window Help
ywork\con1.py ==========
   Rollno    Name  Marks Grade Section    Project
0     101  Ruhani   76.8     A       A    Pending
1     102  George   71.2     B       A  Submitted
2     103  Simran   81.2     A       B  Evaluated
3     104     Ali   61.2     B       C   Assigned
4     105  Kushal   51.6     C       C  Evaluated
5     106  Arsiya   91.6    A+       B  Submitted
6     107  Raunaq   32.5     F       B  Submitted
>>>
```

See, how easy it is to execute sql query and load its result in a dataframe. You can give SQL queries with WHERE condition as well, which we shall learn in the next sub-section.

Step 4 : Process as Required

Once dataframe is loaded with data from a csv file, now you can process and use dataframe's data as required, *e.g.*, you might need to print top or bottom rows of it or display a specific row or column or apply some function on it etc.

Step 5 : Close the connection

After you are through all the processing, in this final step, you need to close the connection established.

This, you can do as follows :

```
<connection object>.close( )
```

For example, we established the database connection via connection object **mycon** above, so, we shall write :

```
mycon.close( )
```

EXAMPLE 11 *The* ***Student*** *table of* ***test*** *database of MySQL stores student details as (Rollno.. Name, Marks, Grade, Section, Project). Write a program to load only rows of this* ***Student*** *table where marks >70 in a dataframe namely* ***mdf****.*

SOLUTION

```
import pandas as pd
import mysql.connector as sqltor
```


Replace this password with the MySQL root user password on your computer

Database name

```
mycon = sqltor.connect(host = "localhost", user = "root", passwd = "MyPass", database ="test" )
if mycon.is_connected( ) :
    print("Student details with marks >70")
    mdf = pd.read_sql( "Select * from student where marks >70 ; " , mycon)
    print(mdf)
else:
    print("MySQL Connection problem")
```

SQL query

Output

```
Student details with marks > 70
   Rollno    Name  Marks Grade Section Project
0     101  Ruhani   76.8     A       A Pending
1     102  George   71.2     B       A Submitted
2     103  Simran   81.2     A       B Evaluated
3     106  Arsiya   91.6    A+       B Submitted
```

4.3.2 Framing Flexible SQL Queries with User Data

In previous section, you learnt about how you can execute simple SQL queries after establishing connection to MySQL database. But SQL queries that were used, were simple queries. Sometimes, you may need to run queries which are based on some parameters or values that you provide from outside, *e.g.*,

```
inp = 70
SELECT * FROM student WHERE MARKS > inp ;
```

Such queries are called **parameterised queries**. To execute parameterised queries in a *mysql.connector* connection, you need to form SQL query strings that include values of parameters. Let us talk about how you can form such query strings.

There are many ways of framing flexible/parameterised query strings but we shall talk about a very simple way of creating it – **String Templates with % Formatting**, which is known as old style formatting. There is a new style of formatting too, which is given in **Appendix B**.

String Templates with % formatting

In this style, string formatting uses this general form : f % v

where **f** is a template string and **v** specifies the value or values to be formatted using that template. If multiple values are to be formatted, **v** must be a tuple. For this you can write the SQL query in a string but use a **%s** code in place of the value to be provided as a parameter.

For example,

```
"select * from student where marks > %s"
```

This means it is a parameter and its value must be provided

The above string is an incomplete string (only f, no %v here) as it contains a placeholder for parameter as %<code>, but its parameter value is not provided here. To complete it, you must provide a tuple of values with% prefixed to it next to the string (called the value tuple, % v), e.g., if you want to provide value 70 for %s placeholder in above query string, you can form the value tuple as follows :

```
(70,)
```
← *For single value to be stored in a tuple, comma must follow it*

Now you can complete the above SQL query string as follows :

```
"select * from student where marks > %s"   %(70,)
```
f v

Now you can store this query string in a variable and then execute that variable through **read_sql( )** function, *e.g.,*

```
1. sname = input("Which student's record you want to see? Enter Name :")
2. qryStr = "Select * from Student where name = ' %s ' ; " % ( sname, )
3. df1 = pd.read_sql( qryStr , <connection> )
```

If you entered name as *'George'* on execution of line 1, then above query string will be framed as :
Select * from Student where name = 'George' ;

Now this dataframe will have only the records having name as you typed on the execution of line 1

You can also format query strings using newstyle of formatting, given in **Appendix B**.

Let us now do some examples.

EXAMPLE 12 *Write modified program of example 10. Take* **marks** *from user and fetch those records which have marks more than input marks.*

SOLUTION

```
import pandas as pd
import mysql.connector as sqltor
mycon = sqltor.connect(host = "localhost", user = "root", passwd = "MyPass", database ="test")
if mycon.is_connected():
    mks = float(input("Enter marks :"))
    qry = "Select * from student where marks > %s ; " % (mks,)
    mdf = pd.read_sql( qry, mycon)
    print("Student details with marks >", mks)
    print(mdf)
else:
    print("MySQL Connection problem")
```

comma to make a tuple from a single value

Sample run :

```
Enter marks : 75
Student details with marks > 75.0
   Rollno  Name   Marks Grade Section Project
0     101  Ruhani 76.8  A     A       Pending
1     103  Simran 81.2  A     B       Evaluated
2     106  Arsiya 91.6  A+    B       Submitted
```

EXAMPLE 13 *Write a modified program of example 11. Take the* **marks range** *from the user, i.e., lower and upper limit of the marks range and fetch those records having marks in this range.*

SOLUTION

```
import pandas as pd
import mysql.connector as sqltor
mycon = sqltor.connect(host = "localhost", user = "root", passwd = "MyPass", database ="test")
if mycon.is_connected():
    lmks = float(input("Enter lower limit of marks range :") )
    hmks = float(input("Enter higher limit of marks range :") )
    qry = "Select * from student where marks between %s and %s ; " %(lmks, hmks)
    mdf = pd.read_sql( qry, mycon)
    print("Student details with marks in the range (", lmks, ",", hmks, ") : ")
    print(mdf)
else:
    print("MySQL Connection problem")
```

Sample run :

```
Enter lower limit of marks range :50
Enter higher limit of marks range :75
Student details with marks in the range ( 50.0 , 75.0 ) :
   Rollno  Name   Marks Grade Section Project
0     102  George 71.2  B     A       Submitted
1     104  Ali    61.2  B     C       Assigned
2     105  Kushal 51.6  C     C       Evaluated
```

IMPORTANT Queries that use % for pattern matching should use %% to signify the pattern matching operator because single % sign is used for a specific purpose *e.g.* %s. Internally %% will be taken as % pattern matching character of SQL.

Following example will make it clear.

EXAMPLE 14 *Write a program to bring those students' details from* **student** *table of MySQL database* **test**, *whose names begin with a specific letter. Take* **the letter** *as user input.*

SOLUTION

```
import pandas as pd
import mysql.connector as sqltor
```

```
mycon = sqltor.connect(host = "localhost", user = "root", passwd = "MyPass", database = "test")
if mycon.is_connected():
    letter = input("Enter beginning letter for the names : ")
    qry = "Select * from student where name like '%s%%' ; " %(letter,)
    mdf = pd.read_sql( qry, mycon)
    print("Student details with names beginning with", letter, "are:")
    print(mdf)
else:
    print("MySQL Connection problem")
```

Internally this %% will be taken as %.

Letter of %s followed by a % sign

Sample run

```
Enter beginning letter for the names : A
Student details with names beginning with A are :
   Rollno  Name   Marks  Grade  Section  Project
0     104   Ali    61.2      B        C  Assigned
1     106  Arsiya  91.6     A+        B  Submitted
==========
Enter beginning letter for the names : R
Student details with names beginning with R are :
   Rollno  Name   Marks  Grade  Section  Project
0     101  Ruhani  76.8      A        A  Pending
1     107  Raunaq  32.5      F        B  Submitted
```

4.4 Exporting a Dataframe's Data as a Table in MySQL Database

Before we proceed to discussing the procedure of exporting a dataframe's data onto a MySQL table, you should know an important thing about connection to a MySQL database. While writing onto MySQL databases, it requires proper **ORM (Object Relational Mapping)** which will ensure that MySQL database tables are created or manipulated exactly as MySQL database would have done natively.

You have been creating connection to MySQL database using **mysql.connector** library which you have used throughout *section 4.3*. But for writing onto MySQL databases, we shall connect to a MySQL database using **pymysql** library which uses **create_engine( )** of **sqlalchemy** library. I know you are thinking, **"Why?"**. Well, the reasons behind it are many, but in simple words :

(*i*) connection with **mysql.connector** supports many flexible parameterized queries for reading from MySQL database using **read_sql( )** but it does not support writing onto MySQL database using **to_sql( )** (the function used for writing on a database table) for MySQL's latest versions.

(*ii*) For writing onto MySQL's versions 5.5 or higher, a connection with strong ORM support is required, which is created through **sqlalchemy's create_engine( )** function.

(*iii*) **Pymysql** library ensures *Python to MySQL* transition of data smoothly.

Now the next question that may arise in your mind is (*see*, I am a mind reader too, haha 😊) - if connection to a MySQL database through **sqlalchemy** and **pysql** is so MySQL friendly, why

can't we use it for read_sql( ) as well? Well, the answer is that you can use it and it will work with read_sql( ) too. BUT, the only thing is that, it is not very parameterised-SQL-queries friendly, *e.g.*, you can't run Example 13's pattern matching SQL query with *read_sql( )* through a connection established using *sqlalachemy* and *pysql*.

After this brief discussion on why we are creating an ORM based connection with MySQL database, let us now learn to export a dataframe's data onto a MySQL table. Thus, make sure to install these two libraries using **pip install <library name>** command on the command prompt.

For exporting a dataframe onto a mySQL table, follow the steps given below:

(*i*) Import pandas, pymysql and sqlalchemy libraries[2] :

(*ii*) Establish connection toMySQL database

(*iii*) Write dataframe's data onto MySQL table

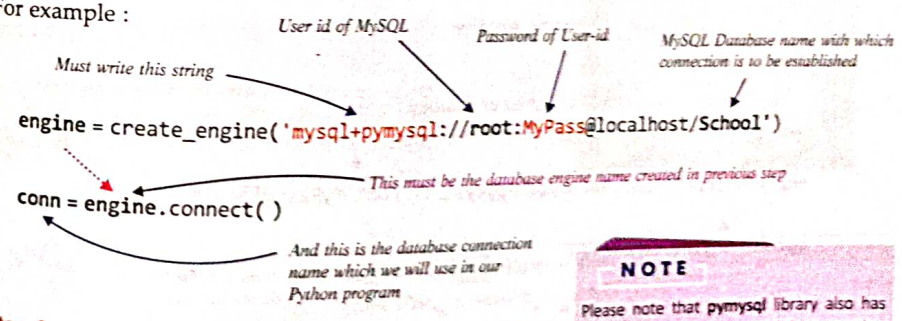
Step 1 : **Import required libraries by issuing commands as :**

```
import pandas as pd
import pymysql
from sqlalchemy import create_engine   # only create_engine is needed from sqlalchemy
```

Step 2 : **Establish connection to database using *create_engine( )*** by giving two commands as explained below :

```
<db engine > = create_engine('mysql+pymysql://<user>:<password>@localhost/<MySQL database>'
<connection name> = <db engine>.connect()
```

For example :



NOTE

Please note that **pymysql** library also has **connect( )** method to establish connection with a MySQL database, but we are using **SqLalchemy's create_engine( )** method as we want to create ORM based connection.

Step 3 : **Write dataframe's data onto MySQL table using to_sql( )**

In this step, now you can write a dataframe in the form of a table by using **to_sql( )** as per the following syntax :

```
<DF>.to_sql(<tablename>, <connection> [index = True] [, if_exists = "append"|"replace"|"fail"])
```

2. Before importing them in your Python program, make sure to install these libraries by issuing following commands on the command prompt (same as you installed *mysql.connector*in section 4.3):
C:\..\Python> pip install **pymysql**
C:\..\Python> pip install **sqlalchemy**

Where,

<DF> is the dataframe whose data is to be exported to MySQL database

<tablename> is the name of table on MySQL database with which connection is established

<connection> is the database connection name established in step 2.

index argument specifies if the a new column for database's indexes is to be created ; by default it is True

if_exists argument specifies what to do if the mentioned table name already exists in the database ; default is "fail" which raises **ValueError** if a table with the mentioned name already exists.

To understand this, consider the statements given below (after importing required libraries). Consider dataframe **tDfas** shown below :

Notice, this dataframe has index numbers , not labels

```
>>> tDf
   Rollno  Name   Marks
0  115     Pavni  97.5
1  236     Rishi  98.0
2  307     Preet  98.5
3  422     Paula  98.0
```

Let us create a table ***Topper1*** in a MySQL database namely ***test*** using the dataframe **tDf**. Carefully look at following statements :

```
import pandas as pd
from sqlalchemy import create_engine
import pymysql
:       # tDf dataframe created or loaded here
engine = create_engine('mysql+pymysql://root:MyPass@localhost/test')
mycon = engine.connect()
tDf.to_sql ('Topper1', mycon, if_exists= 'replace')
```

Table name

User Id and Password at My MySQL client (Replace with your own

Database name

connection name

*What to do if **Topper1** table already exists on the database*

The above code successfully executed and we can check this by running a select query on MySQL client :

```
MySQL 8.0 Command Line Client
mysql> select * from Topper1 ;
+-------+--------+-------+-------+
| index | Rollno | Name  | Marks |
+-------+--------+-------+-------+
|     0 |    115 | Pavni |  97.5 |
|     1 |    236 | Rishi |    98 |
|     2 |    307 | Preet |  98.5 |
|     3 |    422 | Paula |    98 |
+-------+--------+-------+-------+
4 rows in set (0.00 sec)
mysql>
```

Above code has successfully created a table from dataframe tDf's data

Let us consider another dataframe **topDf** which has index labels unlike above shown dataframe *tDf* :

This dataframe has index labels unlike earlier shown dataframe **tDf**, which had index numbers

```
>>> topDf
       Rollno  Name   Marks
Sec A  115     Pavni  97.5
Sec B  236     Rishi  98.0
Sec C  307     Preet  98.5
Sec D  422     Paula  98.0
```

Let us try to write above dataframe using the similar **to_sql( )** function, *i.e.,*

```
:       # code similar to earlier
topDf.to_sql('Topper2', mycon, if_exists = 'replace')
```

But Python showed an error as shown below :

```
sqlalchemy.exc.InvalidRequestError: Could not reflect
```

The reason is that while writing dataframes having index labels , we should give **index = False** argument so that index labels are ignored and MySQL table is created. Thus, if we modify above **to_sql( )** as follows, it will run successfully :

```
topDf.to_sql('Topper2', mycon, index = False, if_exists= 'replace')
```

And now it will run successfully. You can check yourself by running a SELECT query on MySQL client :

```
MySQL 8.0 Command Line Client
mysql> Select * from topper2 ;
+--------+-------+-------+
| Rollno | Name  | Marks |
+--------+-------+-------+
|    115 | Pavni |  97.5 |
|    236 | Rishi |    98 |
|    307 | Preet |  98.5 |
|    422 | Paula |    98 |
+--------+-------+-------+
4 rows in set (0.00 sec)
```

NOTE

Please note that Python may issue a warning if you use any Uppercase letter in table, though it will execute the code successfully; reason being that MySQL is not case sensitive while Python is.

Check Point

4.1

1. What is CSV file ? Is it a text file ?
2. Name the function you would use to read data from a CSV file into a dataframe.
3. Name the function you would use to store data from a dataframe into a CSV file.
4. Which function lets you fetch data from an SQL table into a dataframe ?
5. Which function lets you store data of a dataframe into an SQL table ?
6. Which additional argument would you give to append the data of a dataframe in an existing SQL table ?

EXAMPLE 15 *Write a program to write the dataframe **allDf** given below in **sales** table of **test** database on MySQL. If the table exists, then the records should overwrite the table.*

```
>>>allDf
       Name      Product  Target   Sales
zoneA  Purv      Oven     56000.0  58000.0
zoneB  Paschim   AC       70000.0  68000.0
zoneC  Kendriya  AC       75000.0  78000.0
zoneD  Dakshin   Oven     60000.0  61000.0
zoneE  Uttar     TV       NaN      NaN
zoneF  Rural     Tubewell NaN      NaN
```

SOLUTION

```
import pandas as pd
from sqlalchemy import create_engine
import pymysql
:          # statements to create / load allDf dataframe
# creating connection to database
engine = create_engine('mysql+pymysql://root:MyPass@localhost/test')
mycon = engine.connect( )
allDf.to_sql('sales', mycon, index = False, if_exists = 'replace')
```

Database name

Table name

The output produced by the above code is :

```
MySQL 8.0 Command Line Client
mysql> select * from sales ;
+----------+----------+--------+-------+
| Name     | Product  | Target | Sales |
+----------+----------+--------+-------+
| Purv     | Oven     |  56000 | 58000 |
| Paschim  | AC       |  70000 | 68000 |
| Kendriya | AC       |  75000 | 78000 |
| Dakshin  | Oven     |  60000 | 61000 |
| Uttar    | TV       |   NULL |  NULL |
| Rural    | Tubewell |   NULL |  NULL |
+----------+----------+--------+-------+
6 rows in set (0.00 sec)
```

EXAMPLE 16 *Write a program to write only the top 4 rows of the dataframe allDf used in previous example, in sales2 of test database on MySQL. If the table exists, then the records should get appended to the table.*

SOLUTION

```
import pandas as pd
from sqlalchemy import create_engine
import pymysql
:          # statements to create / load allDf dataframe
# creating connection to database
engine = create_engine('mysql+pymysql://root:MyPass@localhost/test')
mycon = engine.connect( )
allDf.head(4).to_sql('sales2', mycon, index = False, if_exists= 'append')
```

Database name

Top 4 rows extracted from a dataframe to form a dataframe

Output

```
MySQL 8.0 Command Line Client
mysql> select * from sales2;
+----------+---------+--------+-------+
| Name     | Product | Target | Sales |
+----------+---------+--------+-------+
| Purv     | Oven    |  56000 | 58000 |
| Paschim  | AC      |  70000 | 68000 |
| Kendriya | AC      |  75000 | 78000 |
| Dakshin  | Oven    |  60000 | 61000 |
+----------+---------+--------+-------+
4 rows in set (0.00 sec)
```


LET US REVISE

- *The acronym CSV is short for Comma-Separated Values, which refers to a tabular data saved as plaintext where data values are separated by commas.*
- *You can use pandas.read_csv( ) function to read data from a CSV file in your dataframe.*
- *The <DF>.to_csv( ) function saves the data of dataframe in a CSV file.*
- *The function read_csv( ) is pandas' function (carefully read its syntax) and to_csv( ) is dataframe structure's function (carefully read its syntax).*
- *The sqlite3 library can be used to interact with sqlite3 databases from within python.*
- *You can use pandas.read_sql( ) function to read data from an SQL table in your dataframe.*
- *The <DF>.to_sql( ) function saves the data of dataframe in an SQL table.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. A CSV file can take _____ character as separator.
   (a) , (b) -
   (c) | (d) \t
   (e) only (a) (f) all of these
2. In order to work with CSV files from panda, you need to import _____ , other than pandas.
   (a) .csv (b) pandas.io
   (c) newcsv (d) no extra package required
3. The correct statement to read from a CSV file in a dataframe is :
   (a) <DF>.read_csv(<file>) (b) <File>. read_csv( )(<DF>)
   (c) <DF> = pandas.read(<file>) (d) <DF> = pandas.read_csv(<files>)
4. Which argument do you specify with read_csv( ) to specify a separator character ?
   (a) character (b) char
   (c) separator (d) sep
5. To suppress first row as header, which of the following arguments is to be given in read_csv( ) ?
   (a) noheader = True (b) header = None
   (c) skipheader = True (d) header = Null
6. To read specific number of rows from a CSV file, which argument is to be given in read_csv( ) ?
   (a) rows = <n> (b) nrows = <n>
   (c) n_rows = <n> (d) number_rows = <n>
7. To skip first 5 rows of CSV file, which argument will you give in read_csv( ) ?
   (a) skiprows = 5 (b) skip_rows = 5
   (c) skip = 5 (d) noread = 5

8. To skip 1st, 3rd, 5th and 7th row of CSV file, which argument will you give in read_csv( ) ?
   (*a*) skiprows = 1 | 3 | 5 (*b*) skiprows = [1, 5, 1]
   (*c*) skiprows = [1, 3, 5] (*d*) any of these
9. While reading from a CSV file, to use a column's values as index labels, argument given in read_CSV( ) is :
   (*a*) index (*b*) index_col
   (*c*) index_values (*d*) index_label
10. While writing a dataframe onto a CSV file, which argument would you use in to_sql( ) for NaN values' representation as NULL ?
    (*a*) NaN = NULL (*b*) na_rep = NULL
    (*c*) na_value = NULL (*d*) na = NULL
11. Which of the following libraries let you establish a connection with a MySQL database from within Pythan
    (*a*) mysql.connector (*b*) Pymysql, sqlalchemy
    (*c*) pandas (*d*) numpy
12. In pandas.read_sql (<A>, <B>), <A> is
    (*a*) connection name (*b*) table name
    (*c*) SQL query string (*d*) database name
13. In pandas.read-sql (<A>, <B>), <B> is
    (*a*) connection name (*b*) table name
    (*c*) SQL query string (*d*) database name
14. To suppress the creation of a column for index labels of a dataframe, _____ argument is specified in to_sql( ).
    (*a*) if_exists = False (*b*) index = False
    (*c*) index = True (*d*) if_exists = True
15. To append the content of dataframe in a table of MySQL, _____ argument is used in to_sql( ).
    (*a*) if_exists = "Add" (*b*) if_exists = "append"
    (*c*) if_exists = Add (*d*) if_exists = append

Fill in the Blanks

1. Full form of CSV is _____ _____ _____ .
2. Default separator of CSV files is _____ .
3. To load data of a CSV file in a dataframe _____ function is used.
4. To write data of a dataframe in a CSV file, _____ function is used.
5. To specify a separator other than comma in a CSV file, _____ argument is used.
6. To specify the string to represent NaN values in a CSV file, _____ argument in to_sql( ) is used.
7. To load data in a dataframe from mysql table, _____ function is used.
8. To write data of a dataframe in a mysql table, _____ function is used.

True/False Questions

1. CSV files can only store comma separated values.

2. The number of rows in a dataframe are by default equal to number of rows in a CSV file, it created from a CSV file.
3. Pandas can only read from CSV files but can't write CSV files.
4. You need to import CSV package in order to store a dataframe in a CSV file.
5. The read_csv( ) can handle different separator characters but not to_csv( ).
6. In order to read from a MySQL table into a dataframe, the table must exist.
7. The to_sql( ) function can append to a MySQL table.
8. The to_sql( ) function cannot create a MySQL table.
9. The to_sql( ) function cannot work if the named table already exists.
10. The read_sql( ) can also read from CSV files.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *Explain briefly the CSV format of storing files.*

   Solution. The acronym CSV is short for Comma-Separated Values, which refers to a tabular data saved as plaintext where data values are separated by commas.

   In CSV format :

   - Each row of the table is stored in one row *i.e.*, the number of rows in a CSV file are equal to number of rows in the table (or sheet or database table etc.).
   - The field-values of a row are stored together with commas after every field value; but after the last field's value, no comma is given, just the end of line.

2. *Consider the following code and figure out what these are trying to do ? The pandas library has been imported as pd.*

   (*i*) `pd.read_csv("data.csv", nrows = 20 )`
   (*ii*) `pd.read_csv("data.csv", skiprows = [1, 2, 3, 4])`
   (*iii*) `pd.read_csv("data.csv", header = None)`
   (*iv*) `pd.read_csv("data.csv", sep = '@')`
   (*v*) `pd.read_csv("data.csv", index_col = 'Ename')`

   Solution.

   (*i*) The given code is reading a csv file namely ***data.csv*** and it will read only the first 20 rows from it.
   (*ii*) The given code is reading a csv file namely ***data.csv*** and it will skip the mentioned row numbers given in the list for **skiprows** argument.
   (*iii*) The given code is reading a csv file namely ***data.csv*** and it will not read column header row as dataframe column headings.
   (*iv*) The given code is reading a csv file namely ***data.csv*** which has separator character as '@'.
   (*v*) The given code is reading a csv file namely ***data.csv*** and will create dataframe's index labels from column Ename's values.

3. *Following code is reading from employee.csv as shown below and intends to use column Empno's values as the index label. Why is the given code giving error? Suggest solution.*

```
Empno , Name , Designation , Salary
1001, Trupti, Manager, 56000
1002, Raziya, Manager, 55900
1003, Simran, Analyst, 35000
1004, Silviya, Clerk, 25000
1005, Suji, PR Officer, 31000
```

```
import pandas as pd
edf = pd.read_csv("Employee.csv" , index_col = "Empno.")
print(edf)
ValueError: Index Empno. invalid
```

Solution. The above code is giving error because the column name is given as Empno. (a dot in the end) whereas the column name in the csv file is ***Empno*** (without dot), not ***Empno.***

To correct this error, we just need to change the column name as **Empno** for **index_col** argument, *i.e.*, the 2nd line of the code should be :

```
edf = pd.read_csv("Employee.csv" , index_col = "Empno")
```

4. *Following code is reading from file sport.csv. It displays the whole dataframe but when it tries to print the* **Competitions** *column of the dataframe, the code raises error.*

```
import pandas as pd
sdf = pd.read_csv("sport.csv")
print(sdf)
print(sdf.Competitions)
```

Sport.csv

Sport	Competitions	Prizes won
Tennis	14	9
Football	22	16
Chess	25	15

Output generated is :

```
Sport\tCompetitions\tPrizes won
0       Tennis\t14\t9
1       Football\t22\t16
2       Chess\t25\t15
AttributeError: 'DataFrame' object has no attribute 'Competitions'
```

Why is above code giving this error? Suggest a solution.

Solution. The given file sport.csv has a separator as '\t', which is not specified while reading from it. Thus, the code has read the entire row as one column and thus there is no separate column by the name **Competitions**.

To correct the above problem, all we need to do is to specifiy the separator as '\t'. That is, the 2nd line of the code should be :

```
sdf = pd.read_csv("sport.csv" , sep = '\t')
```

Now it will display the column **Competitions'** values as well.

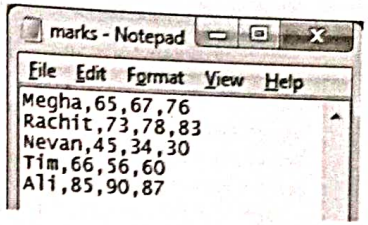
5. *Write a program that reads from a CSV file where the separator character is '$'. Read only first 5 rows in your dataframe.*

- Give column headings as *ItemName, Quanity, Price.*
- Make sure to read first row as data and not as column headers.

Solution.

```
import pandas as pd
df = pd.read_csv("c:\\data\\data.csv", sep = "$", \
     names = ["ItemName", "Quantity", "Price"], header = None, nrows = 5 )
print(df)
```


6. *Write a program that reads from a csv file (marks.csv stored in data folder of C: drive having data as shown below : Name and marks in 3 subjects) in a dataframe. Then the program should add a column 'Total' storing total of marks in three subjects and another column storing average marks. Print the dataframe after adding these columns.*



Solution.

```
import pandas as pd
df = pd.read_csv("c:\\data\\marks.csv", names = ["Name", "Marks1", "Marks2", "Marks3"])
print("Dataframe after fetching data from CSV file")
print(df)
df['Total'] = df['Marks1'] + df['Marks2'] + df['Marks3'] # 'Total' column
df['AvgMarks'] = df['Total'] / 3
print("Dataframe after all the calculations")
print(df)
```

The output produced by above

```
Dataframe after fetching data from CSV file
      Name   Marks1   Marks2   Marks3
0    Megha       65       67       76
1   Rachit       73       78       83
2    Nevan       45       34       30
3      Tim       66       56       60
4      Ali       85       90       87
Dataframe after all the calculations
      Name   Marks1   Marks2   Marks3   Total   AvgMarks
0    Megha       65       67       76     208  69.333333
1   Rachit       73       78       83     234  78.000000
2    Nevan       45       34       30     109  36.333333
3      Tim       66       56       60     182  60.666667
4      Ali       85       90       87     262  87.333333
```

Write a program that lists only those records from SQL table Stud that have marks in the range 50-60.

Note. *Table Stud table is in sqlite 3 data data new.db, stored at c:\sqlite3 folder.*

Solution.

```
import pandas as pd
import sqlite3 as sq
conn = sq.connect("c:\\sqlite3\\new.db")
query = "select * from Stud where marks >=50 and marks <= 60 ;"
df1 = pd.read_sql(query, conn)
print(df1)
```

The output produced by above code is

```
   rollno   name   marks
0    1002  Gauhar    59.5
1    1007   Amal    50.0
```

8. *The* ***Student*** *table of* ***test*** *database of MySQL stores student details as* ***(Rollno., Name, Marks, Grade, Section, Project)****. Write a program to load the data of this* ***Student*** *table in a dataframe. Display the dataframe and also display the details of the topper student (the student having the maximum marks).*

Solution.

```
import pandas as pd
import mysql.connector as sqltor
mycon = sqltor.connect(host = "localhost", \
  user = "root", passwd = "MyPass", database ="test")
if mycon.is_connected():
  qry = " Select * from student ;"
  sdf = pd.read_sql( qry, mycon )
  print( "Dataframe contains:" )
  print( sdf )
  #index of maximum marks
  topper_index = sdf.Marks.idxmax()
  print("Topper student:")
  print(sdf.iloc[ topper_index, :])
else:
  print("MySQL Connection problem")
```

Output :

```
Dataframe contains:
   Rollno  Name    Marks  Grade  Section  Project
0     101  Ruhani   76.8      A        A  Pending
1     102  George   71.2      B        A  Submitted
2     103  Simran   81.2      A        B  Evaluated
3     104  Ali      61.2      B        C  Assigned
4     105  Kushal   51.6      C        C  Evaluated
5     106  Arsiya   91.6     A+        B  Submitted
6     107  Raunaq   32.5      F        B  Submitted
Topper student:
Rollno      106
Name        Arsiya
Marks       91.6
Grade       A+
Section     B
Project     Submitted
```

9. *The* ***Student*** *table of* ***test*** *database of MySQL stores student details as* ***(Rollno., Name, Marks, Grade, Section, Project)****. Write a program to load the student records of this* ***Student*** *table where grade is 'A' or 'A+' in a dataframe. Display the dataframe.*

Solution.

```
import pandas as pd
import mysql.connector as sqltor
mycon = sqltor.connect(host = "localhost",\
  user = "root", passwd = "MyPass", database ="test")
```



```
if mycon.is_connected():
  qry = "Select * from student where grade in ('A', 'A+'); "
  mdf = pd.read_sql( qry, mycon)
  print("Student details with grades A or A+ are :")
  print(mdf)
else:
  print("MySQL Connection problem")
```

Output :

```
Student details with grades A or A+ are :
   Rollno  Name    Marks  Grade  Section  Project
0     101  Ruhani   76.8      A        A  Pending
1     103  Simran   81.2      A        B  Evaluated
2     106  Arsiya   91.6     A+        B  Submitted
```

10. *Dataframe* ***saleDf*** *stores about 50 rows in it. Write a program to store its rows from 10 to 15 in a table* ***random*** *on MySQL database namely 'world'. Append the rows if the table already exists.*

Solution

```
import pandas as pd
from sqlalchemy import create_engine
import pymysql
engine = create_engine('mysql+pymysql://root:MyPass@localhost/world')
mycon = engine.connect()
:        # statements to create or load saleDf
saleDf.iloc[10:15, : ].to_sql('random', mycon, index = False, if_exists='replace')
```

GLOSSARY

CSV	Acronym for Comma Separated Values
CSV File Format	A text file format storing data values separated by commas (or any other character as a separator)

Assignments

Type A : Short Answer Questions/Conceptual Questions

1. What are advantages of CSV file formats ?
2. What all libraries do you require in order to bring data from a CSV file into a dataframe ?
3. You want to read data from a CSV file in a dataframe but you want to provide your own column names to dataframe. What additional argument would you specify in read_csv( ) ?
4. By default, read_csv( ) uses the values of first row as column headers in dataframes. Which argument will you give to ensure that the top/first row's data is used as data and not as column headers ?
5. Which argument would you give to **read.csv( )** if you only want to read top 10 rows of data ?
6. Write command to store data of dataframe **mdf** into a CSV file Mydata.csv, with separator character as '@'.

7. Why do you need connection to an SQL database in order to get data from a table ?
8. What is pymysql library of Python ?
9. What all libraries do you require in order to interact with MySQL databases (and dataframe) from within Python ?
10. What additional argument do you need to specify in to_sql( ) so that old data of MySQL table is retained ?
11. If **query** is a string storing an SQL statement. Write statements so that the data is fetched based on query from SQL database Mydata.

Type B : Application Based Questions

1. Predict the output of following code fragments one by one. For every next code fragment, consider that the changes by previous code fragment are in place. That is, for code fragment (*b*), changes made by code fragment (*a*) are persisting ; for (*c*), changes by (*a*) and (*b*) are persisting and so on.

(*a*)
```
import pandas as pd
columns=['2015','2016', '2017', '2018']
index=['Messi','Ronaldo','Neymar','Hazard']
df = pd.DataFrame(columns=columns,index=index)
print (df)
df.to_csv("c:\one.csv")
```

(*b*)
```
df['2015']['Messi'] = 12
df['2016']['Ronaldo'] = 11
df['2017']['Neymar'] = 8
df['2018']['Hazard'] = 16
print (df)
df.to_csv("c:\two.csv", sep = '@')
```

(*c*)
```
new_df = pd.read_csv('c:\one.csv',index_col= 0)
print(new_df)
```

(*d*)
```
new_df = pd.read_csv('c:\one.csv')
print(new_df)
```

(*e*)
```
new_df = pd.read_csv('c:\two.csv')
print(new_df)
```

(*f*)
```
new_df = pd.read_csv('c:\two.csv', sep = '@')
print(new_df)
```

2. Are the following two statements same? Why/Why not?
   (*i*) `pd.read_csv('zoo.csv', sep = ',')`
   (*ii*) `pd.read_csv('zoo.csv')`
3. How are following two codes similar of different ? What output will they produce ?

(*i*)
```
df = pd.read_csv("data.csv", nrows = 5)
print(df)
```

(*ii*)
```
df = pd.read_csv("data.csv")
print(df)
```


4. What is the difference between following two statements ?
   (*i*) `df.to_sql('houses', con = conn, if_exists = 'replace')`
   (*ii*) `df.to_sql('houses', con = conn, if_exists = 'replace', index = False)`
5. Consider following code when **conn** is the name of established connection to MySQL database.

```
Cars = { 'Brand': ['Alto', 'Zen', 'City', 'Kia'],
         'Price': [22000, 25000, 27000, 35000]
       }
df = DataFrame(Cars, columns= ['Brand', 'Price'])
df.to_sql('CARS', conn, if_exists = 'replace', index = False)
```

What will be the output of following query if executed on MySQL :

```
SELECT * from CARS ;
```

6. Consider following code when **conn** is the name of established connection to MySQL database.

```
sql = SELECT * from Sales where zone = "central"
df = pandas.read_sql(sql, conn)
df.head()
```

What will be stored in df ?

Type C : Programming Practice/Knowledge based Questions

1. Write a program to read details such as **Item, Sales made** in a dataframe and then store this data in a CSV file.
2. Write a program to read data from a CSV file where separator character is '@'. Make sure that :
   - the top row is used as data, not as column headers.
   - only 20 rows are read into dataframe.
3. Write a prograin to get following data in two dataframes :

df1

Roll no	Name
1	ABC
2	DEF
⋮	

df2

Roll no	Marks 1	Marks 2	Marks 3
1	70	80	75
2	60	65	70
⋮			

Store these dataframes as two separate tables in the same database.

4. You have a database on MySQL namely school having three tables in it – *Student, Subject, Teacher*. Write a program to store these tables in three dataframes.
5. The Dataframe SDF stores the sales records of 100 salesmen. Write a program to store this as a table in database namely "company" on MySQL.
6. Consider the SDF dataframe storing the sales records of 100 salesmen, write a program that stores only the first 25 rows of the dataframe in a table on MySQL database.
7. The sales table of company databse of MySQL stores the sales records of 100 salesmen. Write a program to load only those records in datafrem which have made sales more than of Rs 50000/- .

5 MySQL SQL Revision Tour

Outline

5.1 Introduction
5.2 Relational Data Model
5.3 MySQL–A Popular Relational Database Management System
5.4 Accessing Database in MySQL
5.5 Creating Tables in MySQL
5.6 Inserting Data into Table
5.7 Making Simple Queries Through Select Command
5.8 Creating Tables with SQL Constraints
5.9 Viewing a Table Structure
5.10 Inserting Data into Another Table
5.11 Modifying Data in Tables
5.12 Deleting Data from Tables
5.13 Altering Tables
5.14 Dropping Tables

5.1 Introduction

A database system is basically a computer based record keeping system. The collection of data, usually referred to as the *database*, contains information about one particular enterprise. In a typical file-processing system, permanent records are stored in various files. A number of different application programs are written to extract records from files and add records to the appropriate files. But this scheme has a number of major limitations and disadvantages, such as data redundancy (duplication of data), data inconsistency, unsharable data, unstandardized data, insecure data, incorrect data etc. A *database management system* is answer to all these problems as it provides a centralized control of the data.

Various advantages of database systems are :

- Database systems reduce data redundancy (*data duplication*) to a large extent.
- Database systems control data inconsistency to a large extent.
- Databases facilitate sharing of data.
- Databases enforce standards.
- Centralized databases can ensure data security.
- Integrity can be maintained through databases.

5.2 Relational Data Model

A data model refers to a set of concepts to describe the structure of a database, and certain *constraints* (restrictions) that the database should obey.

Most commonly used data model for databases is the **relational data model.**

In relational data model, the data is organized into tables (*i.e.*, rows and columns). These tables are called *relations*. A row in a table represents a *relationship* among a set of values. Since a table is a collection of such relationships, it is generally referred to using the mathematical term *relation*, from which the relational data model derives its name.

Let us now revise different terms used in relational model.

- *Relation* — A table storing logically related data ; data must be atomic in a cell ; all rows of this table are distinct ; ordering of rows and columns is immaterial.
- *Domain* — This is a pool of values from which the actual values appearing in a given column are drawn.
- *Tuple* — A row of a relation is generally referred to as a *tuple*.
- *Attribute* — A column of a relation is generally referred to as an *attribute*.
- *Degree* — This refers to the number of attributes in a relation.
- *Cardinality* — This refers to the number of tuples in a relation.
- *View* — It is a virtual table that does not really exist in its own right but is instead derived from one or more underlying base table(s).
- *Primary Key* — This refers to a set of one or more attributes that can uniquely identify tuples within the relation.
- *Candidate Key* — All attribute combinations inside a relation that can serve as primary key are *candidate keys* as these are candidates for primary key position.
- *Alternate Key* — A candidate key that is not primary key, is called an *alternate key*.
- *Foreign Key* — A non-key attribute, whose values are derived from the primary key of some other table, is known as *foreign key* in its current table.

Referential Integrity

A referential integrity is a system of rules that a DBMS uses to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data.

You can set referential integrity when all of the following conditions are met :

- The matching field from the primary table is a primary key or has a unique index.
- The related fields have the same data type.
- Both tables belong to the same database. Referential integrity can't be enforced for linked table from databases in other formats.

When referential integrity is enforced, you must observe the following rules :

- You can't enter a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table. However, you can enter a Null value in the foreign key, specifying that the records are unrelated.
- You can't delete a record from a primary table if matching records exist in a related table.
- You can't change a primary key value in the primary table, if that record has related records.

5.3 MySQL – A Popular Relational DataBase Management System

MySQL is a freely available open source *Relational Database Management System* (RDBMS) that uses Structured Query Language (SQL). It is downloadable from site www.mysql.org. In a MySQL database, information is stored in *Tables*. A single MySQL database can contain many tables at once and store thousands of individual records. MySQL provides you with a rich set of features that support a secure environment for storing, maintaining, and accessing data. MySQL is a fast, reliable, scalable alternative to many of the commercial RDBMSs available today.

MySQL Database System

MySQL database system refers to the combination of a MySQL server instance and a MySQL database. MySQL operates using client/server architecture in which the server runs on the machine containing the databases and clients connect to the server over a network.

- **The server** (MySQL server) listens for client requests coming in over the network and accesses database contents according to those requests and provides that to the clients.
- **Clients** are programs that connect to the database server and issue queries in a pre-specified format. MySQL is compatible with the standards based SQL (Structured Query Language).

5.3.1 MySQL and SQL

In order to access data within the MySQL database, all programs and users must use, *Structured Query Language* (SQL). SQL is the set of commands that is recognised by nearly all RDBMSs.

The Structured Query Language (SQL) is a language that enables you to create and operate on relational databases, which are sets of related information stored in tables.

Classification of SQL Statements

SQL provides many different types of commands used for different purposes. SQL commands can be mainly divided into following categories :

(*i*) **Data Definition Language (DDL) Commands.** Commands that allow you to perform tasks related to data definition *e.g.*,

- creating, altering and dropping.
- granting and revoking privileges and roles.
- maintenance commands

(*ii*) **Data Manipulation Language (DML) Commands.** Commands that allow you to perform data manipulation *e.g.*, retrieval, insertion, deletion and modification of data stored in a database.

(*iii*) **Transaction Control Language (TCL) Commands.** Commands that allow you to manage and control the transactions (a transaction is one complete unit of work involving many steps), *e.g.*,

- making changes to database, permanent
- undoing changes to database, permanent
- creating savepoints
- setting properties for current transactions

There are other categories of SQL commands also but above three categories of commands are mainly used by learners.

5.3.2 Common MySQL Data Types

Data types are means to identify the type of data and associated operations for handling it. A value's *datatype* associates a fixed set of properties with the value. Some commonly used data types of MySQL are briefly listed below.

MySQL Data Types

Data Type	Spec	Data Type	Spec
CHAR	String (0 – 255)	INT	Integer (– 2147483648 to 2147483647)
VARCHAR	String (0 – 255)	BIGINT	Integer (– 9223372036854775808 to 9223372036854775807)
TINYTEXT	String (0 – 255)	FLOAT	Decimal (precise to 23 digits)
TEXT	String (0 – 65535)	DOUBLE	Decimal (24 to 53 digits)
BLOB	String (0 – 65535)	DECIMAL	"DOUBLE" stored as string
MEDIUMTEXT	String (0 – 16777215)	DATE	YYYY-MM-DD
MEDIUMBLOB	String (0 – 16777215)	DATETIME	YYYY-MM-DDHH:MM:SS
LONGTEXT	String (0 – 4294967295)	TIMESTAMP	YYYYMMDDHHMMSS
LONGBLOB	String (0 – 4294967295)	TIME	HH:MM:SS
TINYINT	Integer (– 128 to 127)	ENUM	One of preset options
SMALLINT	Integer (– 32768 to 32767)	SET	Selection of preset options
MEDIUMINT	Integer (– 8388608 to 8388607)	BOOLEAN	TINYINT(1)

Difference between Char and Varchar

The difference between CHAR and VARCHAR is that of *fixed length* and *variable length*. The CHAR datatype specifies a *fixed length* character string. When a column is given datatype as CHAR(*n*), then MySQL ensures that all values stored in that column have this length *i.e.*, *n* bytes. If a value is shorter than this length *n* then blanks are added, but the size of value remains *n* bytes.

VARCHAR, on the other hand, specifies a variable length string. When a column is given datatype as VARCHAR(*n*), then the maximum size a value in this column can have is *n* bytes. Each value that is stored in this column stores exactly as you specify it *i.e.*, no blanks are added if the length is shorter than maximum length *n*. However, if you exceed the maximum length *n*, then an error message is displayed.

5.4 Accessing Database in MySQL

only command and example

Before you start writing SQL commands or making queries upon the data in tables of a database, you need to open the database for use. For this, after logging into MySQL, you need to issue a command

```
Use <databasename> ;
```

For example, we want to work on our sample database namely *menagerie*, so we shall write the following command after logging in *MySQL*.

```
mysql> USE menagerie ;
Database changed
mysql>
```


5.5 Creating Tables in MySQL

Tables are defined with the CREATE TABLE command. When a table is created, its columns are named, data types and sizes are supplied for each column. Each table must have at least one column.

The syntax of CREATE TABLE command is :

```
CREATE TABLE <table-name>
      (<column name> <data type> [ (<size>)],
      <columnname> <data type> [ ( <size> ) ... ]) ;
```

NOTE

Before issuing a CREATE TABLE command, make sure that its parent database has been opened using USE <database name> command.

To create an *employee* table whose schema is as follows :

```
employee (ecode, ename, sex, grade, gross)
```

the *SQL* command will be

```
CREATE TABLE employee
(   ecode    integer,
    ename    char(20),
    sex      char(1),
    grade    char(2),
    gross    decimal ) ;
```

5.6 Inserting Data into Table

The rows (tuples) are added to relations using INSERT command of *SQL*. In its simplest form, INSERT takes the following syntax :

```
INSERT INTO <tablename> [ <column list> ]
VALUES ( <value>, <value> ... ) ;
```

For example, to enter a row into *employee* table (defined earlier), you could use the following statement :

```
INSERT INTO employee
VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;
```

See the order of values matches the order of columns in the CREATE TABLE command of *employee*. The same can be done with an alternate command as shown below :

```
INSERT INTO employee (ecode, ename, sex, grade, gross)
VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;
```

The INSERT statement adds a new row to *employee* giving a value for every column in the row. Note that the data values are in the same order as the column names in the table. Data can be added only to some columns in a row by specifying the columns and their data.

For instance, if you want to insert only *ecode*, *ename* and *sex* columns, you use the command :

```
INSERT INTO employee (ecode, ename, sex)
VALUES (2014, 'Manju', 'F') ;
```

The columns that are not listed in the INSERT command will have their default value, if it is defined for them, otherwise, NULL value.

If any other column (that does not have a **default value** and is defined NOT NULL) is skipped or omitted, an error message is generated and the row is not added.

NOTE

In an INSERT statement, only those columns can be omitted that have either default value defined or they allow NULL values.

Inserting NULL values

To insert value NULL in a specific column, you can type NULL without quotes and NULL will be inserted in that column.

Consider the following statement :

```
INSERT INTO EMPL (Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno)
VALUES (8100, 'YASH', 'ANALYST', NULL, '10-MAY-03', 6000, NULL, 20) ;
```

See, for **Mgr** and **Comm** columns, NULL values have been inserted.

Inserting Dates

Dates are by default entered in 'YYYY-MM-DD' format *i.e.*, first four digits depicting *year*, followed by a hyphen, followed by 2 digits of *month*, followed by a hyphen and a two digit day. All this is enclosed in single quotes.

5.7 Making Simple Queries Through Select Command

The SELECT statement is used to pull information from a table. The general form of the statement is :

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy ;
```

5.7.1 Selecting All Data

The simplest form of SELECT retrieves everything from a table. You just need to specify asterisk in the *select-list*(what_to_select), *e.g.*, :

```
mysql> SELECT * FROM pet ;
```

5.7.2 Selecting Particular Rows

You can select particular rows from a table by specify filtering condition through WHERE clause of the SELECT statement, *e.g.*,

1. Select all pets with gender(sex) as ***male***("m").

```
mysql> SELECT * FROM pet
       WHERE sex = 'm';
```

2. Select all pets that were born on or after Jan 1, 2019.

```
mysql> SELECT * FROM pet
       WHERE birth >= '2019-1-1';
```
← *date in yyyy-mm-dd format*

3. Select all female-dogs.

```
mysql> SELECT * FROM pet
       WHERE species = 'dog' AND sex = 'f';
```

4. Select all snakes or birds.

```
mysql> SELECT * FROM pet
       WHERE species = 'snake' OR species = 'bird';
```

5. Select all male cats.

```
mysql> SELECT * FROM pet
       WHERE (species = 'cat' AND sex = 'm') ;
```

5.7.3 Selecting Particular Columns

You can select particular columns by specifying column-names (*i.e.*, attributes) in the select-list of the SELECT command, *e.g.*,

1. Display names and birth-dates of all pets.

```
mysql> SELECT name, birth FROM pet ;
```

2. Display owners of pets born after Dec 2018.

```
mysql> SELECT owner FROM pet
       WHERE birth >'2018-12-31' ;
```
← *date in yyyy-mm-dd format*

5.7.4 Eliminating Redundant Data (with Keyword DISTINCT)

By default, data is selected from all the rows of the table, even if the data appearing in the result gets duplicated. The DISTINCT keyword eliminates duplicate rows from the results of a SELECT statement. *For example,*

1. Display names of all pet-owners (non-redundant).

```
mysql> SELECT DISTINCT owner FROM pet;
```

2. Display distinct species of pets from table pet.

```
mysql> SELECT DISTINCT(species) FROM pet.
```

5.7.5 Selecting From All the Rows – ALL Keyword

If in place of keyword DISTINCT, you give keyword ALL then the result retains the duplicate output rows. It is just the same as when you specify neither DISTINCT nor ALL ; ALL is essentially a clarifier rather than a functional argument. Thus if you give

```
SELECT ALL city FROM suppliers ;
```

it will give values of *city* column from every row of the table without considering the duplicate entries.

5.7.6 Viewing Structure of a Table

If you want to know the structure of a table, you can use *Describe* or *Desc* command as per following syntax :

```
DESCRIBE | DESC <table name> ;
```

For instance, the commands : DESCRIBE pet ; or DESC pet ; will display the structure of table ***pet***.

```
mysql> DESC pet ;
```

5.7.7 Performing Simple Calculations

To perform simple calculations, you can write the expression/formula to be calculated next to keyword SELECT, *e.g.*,

1. To calculate 3.14159*6*6

```
mysql> SELECT 3.14159*6*6 ;
```

2. To obtain current system date

```
mysql> SELECT curdate();
```

5.7.8 Using Column Aliases

The columns that you select in a query can be given a different name *i.e.*, **column alias name** for output purposes. As per following syntax :

```
Select <columnname> AS [columnalias] [, <columnname> AS [columnalias]]
    .....
From <tablename> ;
```

For example,

```
mysql> SELECT date, type AS "Event Type"
    -> FROM event ;
```

5.7.9 Condition Based on a Range

The BETWEEN operator defines a range of values that the column values must fall in to make the condition true. The range includes both lower value and the upper value. For example, to list the items whose *QOH* falls between 30 to 50 (both inclusive), the command would be :

```
SELECT icode, descp, QOH
FROM items
WHERE QOH BETWEEN 30 AND 50 ;
```

5.7.10 Condition Based on a List

To specify a list of values, IN operator is used. The IN operator selects values that match any value in a given list of values. For example, to display a list of members from 'DELHI', 'MUMBAI', 'CHENNAI' or 'BANGALORE' cities, you may give

```
SELECT * FROM members
WHERE city IN ('DELHI', 'MUMBAI', 'CHENNAI', 'BANGALORE') ;
```

The NOT IN operator finds rows that do not match in the list. So if you write

```
SELECT * FROM members
WHERE city NOT IN ('DELHI', 'MUMBAI', 'CHENNAI') ;
```

it will list members not from the cities mentioned in the list.

5.7.11 Condition Based on Pattern Matches

SQL also includes a string-matching operator, LIKE, for comparisons on character strings using patterns. Patterns are described using two special wildcard characters :

- **percent (%)**. The % character matches any substring.
- **underscore (_)**. The _ character matches any character.

The LIKE keyword is used to select rows containing columns that match a wildcard pattern.

Examples :

1. To list members which are in areas with pin codes starting with 13, the command is :

```
SELECT firstname, lastname, city
FROM members
WHERE pin LIKE '13%' ;
```

2. To list names of pets who have names ending with '*y*', the command would be :

```
SELECT name
FROM emp
WHERE name LIKE '%y' ;
```

5.7.12 Searching for NULL

The NULL value in a column can be searched for in a table using IS NULL in the WHERE clause. (Relational operators like =, <> etc. can't be used with NULL). For example, to list details of all employees whose departments contain NULL (*i.e.*, novalue), you use the command :

```
SELECT empno, empname, job
FROM emp WHERE DeptNo IS NULL ;
```

Non-NULL values in a table can be listed using IS NOT NULL.

5.8 Creating Tables with SQL Constraints

You already know that a table is created using CREATE TABLE command. The SQL syntax for **CREATE TABLE** is

```
CREATE TABLE "table_name"
("column 1" "data_type_for_column_1",
"column 2" "data_type_for_column_2", ... ) ;
```

So, if we are to create the customer table specified as above, we would type in

```
CREATE TABLE customer
(   First_Name char(50),
    Last_Name char(50),
    Address char(50),
    City char(50),
    Country char(25),
    Birth_Date date   ) ;
```

But while creating tables, we may need to apply certain conditions on columns *e.g.*, this column must not be blank or NULL for any record. To apply conditions on columns, SQL constraints are used.

8.1 SQL Constraints

Common types of constraints that are applied on table columns include the following :

S.No.	Constraints	Description
1.	*NOT NULL*	❖ Ensures that a column cannot have NULL value.
2.	*DEFAULT*	❖ Provides a default value for a column when none is specified.
3.	*UNIQUE*	❖ Ensures that all values in a column are different.
4.	*CHECK*	❖ Makes sure that all values in a column satisfy certain criteria.
5.	*Primary Key*	❖ Used to uniquely identify a row in the table.
6.	*Foreign Key*	❖ Used to ensure referential integrity of the data.

5.8.1A SQL NOT NULL Constraint

By default, a column can hold NULL. If you do not want to allow NULL value in a column, you will want to place a constraint on this column specifying that NULL is now not an allowable value.

For example, in the following statement,

```
CREATE TABLE Customer
(   SID integer NOT NULL,
    Last_Name varchar (30) NOT NULL,
    First_Name varchar(30) );
```

CONSTRAINT

> "A **Constraint** is a condition or check applicable on a field or set of fields."

Columns *SID* and *Last_Name* cannot include NULL, while *First_Name* can include NULL.

An attempt to execute the following SQL statement,

```
INSERT INTO Customer (Last_Name, First_Name)
VALUES ('Wang','Pedro') ;
```

will result in an error because this will lead to column *SID* being NULL, which violates the NOT NULL constraint on that column.

5.8.1B SQL DEFAULT Constraint

The DEFAULT constraint provides a *default value* to a column when the INSERT INTO statement does not provide a specific value.

For example, if we create a table as below :

```
CREATE TABLE Student
(   Student_ID integer Unique,
    Last_Name varchar (30),
    First_Name varchar (30),
    Score DEFAULT 80 );
```

and execute the following SQL statement,

```
INSERT INTO Student (Student_ID, Last_Name, First_Name)
VALUES ('10','Qureshi','Zeeshan') ;
```

See, no value has been provided for Score field

After this SQL query, the table will look like the following :

Student_ID	Last_Name	First_Name	Score
10	Qureshi	Zeeshan	80

See, Score field has got the default value

Even though we didn't specify a value for the *Score* column in the INSERT INTO statement, it does get assigned the *default value* of *80* since we had already set *80* as the *default value* for this column.

5.8.1C SQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are distinct. In other words, no two rows can hold the same value for a column with UNIQUE constraint.

For example, in the following CREATE TABLE statement,

```
CREATE TABLE Customer
(   SID integer Unique,
    Last_Name varchar (30),
    First_Name varchar(30) );
```

column *SID* has a **unique** constraint, and hence cannot include duplicate values. Such constraint does not hold for columns *Last_Name* and *First_Name*. So, if the table already contains the following rows :

SID	Last_Name	First_Name
1	Wang	Pedro
2	Qureshi	Zeeshan
3	Rastogi	Rajiv

Executing the following SQL statement,

```
INSERT INTO Customer
VALUES ('3','Cyrus','Grace');
```

will result in an error because the value *3* already exists in the *SID* column, thus trying to insert another row with that value violates the UNIQUE constraint.

NOTE

Please note that a column that is specified as a primary key must also be unique. At the same time, a column that's unique may or may not be a primary key. In addition, multiple UNIQUE constraints can be defined on a table but PRIMARY KEY constraint is defined only once in the table.

5.8.1D SQL CHECK Constraint

The CHECK constraint ensures that all values in a column satisfy certain conditions. Once defined, the database will only insert a new row or update an existing row if the new value satisfies the CHECK constraint. The CHECK constraint is used to ensure data quality For example, in the following CREATE TABLE statement,

```
CREATE TABLE Customer
(   SID integer CHECK (SID > 0),
    Last_Name varchar (30),
    First_Name varchar(30) );
```

NOTE

Please note that MySQL does not enforce the CHECK constraint although MySQL syntax allows the definition of CHECK constraint. As per MySQL documentation, 'Check constraint gets parsed but IGNORED'.

5.8.1E PRIMARY KEY Constraint

A primary key is used to uniquely identify each row in a table. It can either be part of the actual record itself, or it can be an artificial field (one that has nothing to do with the actual record). A primary key can consist of one or more fields on a table. When multiple fields are used as a primary key, they are called a **composite key**.

Primary keys can be specified either when the table is created (using **CREATE TABLE**) or by changing the existing table structure (using **ALTER TABLE**).

Defining Primary Key through Create Table Command

You can define a primary key in CREATE TABLE command through keywords PRIMARY KEY.

Below are examples for specifying a primary key when creating a table :

```
CREATE TABLE Customer
(   SID integer not null PRIMARY KEY,
    Last_Name varchar(30),
    First_Name varchar(30)  );
```

Or

```
CREATE TABLE Customer
(   SID integer not null,
    Last_Name varchar(30),
    First_Name varchar(30),
    PRIMARY KEY (SID)  );
```

The latter way is useful if you want to specify a *composite primary key* (*i.e.*, having a group of fields) *e.g.*,

```
CREATE TABLE Customer
(   Branch integer not null,
    SID integer not null,
    Last_Name varchar(30),
    First_Name varchar(30),
    PRIMARY KEY (Branch, SID) );
```

*Combination of fields **Branch and SID** is the composite primary key of the table.*

Defining Primary Key through Alter Table Command

You can define a primary key in ALTER TABLE command through keywords

```
ADD PRIMARY KEY (<key-field>)
```

Below are examples for specifying a primary key by altering a table :

```
ALTER TABLE Customer
ADD PRIMARY KEY (SID);
```

NOTE

Before using the ALTER TABLE command to add a primary key, you'll need to make sure that the field is defined as 'NOT NULL' – in other words, NULL cannot be an accepted value for that field.

5.8.1F Foreign Key Constraint

In an RDBMS, tables reference one another through common fields and to ensure validity of references, *referential integrity* is enforced. *Referential integrity is a system of rules that a DBMS uses*

to ensure that relationships between records in related tables are valid, and that users don't accidentally delete or change related data. Referential integrity is ensured through FOREIGN KEY constraint. This is implemented as explained in the following paragraph.

Whenever two tables are related by a common column (or set of columns), then the related column(s) in the **parent table** (or **primary table**) should be either declared a PRIMARY KEY or UNIQUE key and the related column(s) in the **child table** (or **related table**) should have FOREIGN KEY constraint. For instance, if we have two tables having structures as given below :

Table : CUSTOMER

column name	characteristic
SID	Primary Key
Last_Name	
First_Name	

Table : ORDERS

column name	characteristic
Order_ID	Primary Key
Order_Date	
Customer_SID	Foreign Key
Amount	

In the above example, the **Customer_SID** column in the *ORDERS* table is a foreign key pointing to the **SID** column in the *CUSTOMER* table.

Just like primary key, Foreign key can also be created in *two* ways : through CREATE TABLE and ALTER TABLE commands.

Defining Foreign key through Create Table

In Create Table command, you can add foreign key's definition through following syntax :

```
Foreign Key (<column-to-be-designated-as-foreign-key>) references
Master-Table(<primary-key-of master-table>);
```

Following example shows how to specify the foreign key when creating the ORDERS table :

```
CREATE TABLE ORDERS
(   Order_ID integer,
    Order_Date date,
    Customer_SID integer,
    Amount double,
    Primary Key (Order_ID),
    Foreign Key (Customer_SID) references CUSTOMER(SID) );
```

The above code will designate **Customer_SID** field of *ORDERS* table as foreign key referencing **SID** field of *CUSTOMER* table.

Defining Foreign key through Alter Table

In Alter Table command, you can add foreign key's definition through following syntax :

```
ALTER TABLE <table-name>
ADD FOREIGN KEY (<column-to-be-designated-as-foreign-key>)
references Master-Table(<primary-key-of master-table>) ;
```

Following example specifies a foreign key by altering a table. This assumes that the ORDERS table has been created, and the foreign key has not yet been put in :

```
ALTER TABLE ORDERS
ADD FOREIGN KEY (customer_sid) REFERENCES CUSTOMER(SID);
```

IMPORTANT

Foreign Key and Storage Engine

MySQL is capable of creating databases in many different storage engines that offer different features. The default storage engine is ISAM but ironically it does not support foreign keys. So, if you find that your foreign key constraint are being ignored or not implemented, you need to change the storage engine of your database tables to InnoDB storage engine.

To do so you simply need to write following command for your connecting tables :

```
ALTER TABLE <tablename> ENGINE = innodb ;
```

e.g.,

```
ALTER TABLE table1 ENGINE = innodb ;
```

Once done, you can create and implement foreign keys for your tables.

5.8.2 Applying Table Constraints

When a constraint is to be applied on a group of columns of the table, it is called ***table constraint***. The table constraints appear in the end of table definition. For instance, if you want combination of *icode* and *descp* of table *items* to be unique, you may write it as follows :

```
CREATE TABLE items
(   icode    char (5)     NOT NULL,
    descp    char (20)    NOT NULL,
    ROL      integer,
    QOH      integer,
    CHECK    (ROL < QOH),
    UNIQUE   (icode, descp) ) ;
```

these are a table constraints

The above statement ensures that the combination of *icode* and *desc* in each row must be unique.

A constraint applied on one column (*e.g.,* as you define **not null** with a column definition) is known as **column constraints**.

5.9 Viewing a Table Structure

Once you have created a table, you may want to view its structure. Anytime, if you want to view an already created/existing table's structure, you may use DESC[RIBE] command of MySQL.

You can use DESC[RIBE] command as follows :

```
DESC[RIBE] <tablename> ;
```

For example,

```
DESC empl ;
```

or

```
DESCRIBE empl ;
```


5.10 Inserting Data Into another Table

You already know that data is added to tables using INSERT INTO command *e.g.,*

```
INSERT INTO employee
VALUES (1001, 'Rahman', 'M', 'E4', 14700.00) ;
```

Or

```
INSERT INTO employee (ecode, ename, sex)
VALUES (3005, 'Meera', 'F') ;
```

INSERT command can also be used to take or derive values from one table and place them in another by using it with a query. To do this, simply replace the VALUES clause with an appropriate query as shown in the following example :

```
INSERT INTO branch1
SELECT * FROM branch2
WHERE gross > 7000.00 ;
```

This query will generate data to be inserted into table

It will extract all those rows from *branch2* that have *gross* more than 7000.00 and insert this produced result into the table *branch1*. But for above command to work, table namely **branch1** must be an existing table of the database.

5.11 Modifying Data in Tables

You can modify data in tables using UPDATE command of SQL. The UPDATE command specifies the rows to be changed using the WHERE clause, and the new data using the SET keyword. The new data can be a specified constant, an expression or data from other tables. For example, to change the reorder level *ROL* of all items to 250, you would write

```
UPDATE items
SET ROL = 250 ;
```

To update the *ROL* and *QOH* for items having icode less than 'I040', we shall write

```
UPDATE items
SET ROL = 400, QOH = 700
WHERE icode < 'I040' ;
```

To double the gross pay of employees of grade 'E3' and 'E4', you use the command :

```
UPDATE employee
SET gross = gross * 2
WHERE (grade = 'E3' OR grade = 'E4') ;
```

5.12 Deleting Data from Tables

To delete some data from tables, you can use SQL DELETE commands. The DELETE command removes rows from a table. This removes the entire rows, not individual field values, so no field argument is needed or accepted.

The DELETE statement takes the following general form :

```
DELETE FROM <tablename>
[WHERE <predicate> ] ;
```

For instance, to remove the tuples from *employee* table that have *gross* (salary) less than 2200.00, the following command is used :

```
DELETE FROM employee
WHERE gross < 2200.00 ;
```

If you specify no condition with WHERE, then all the rows of the table will be deleted, *e.g.*, To remove all the contents of *items* table, you use the command :

```
DELETE FROM items ;
```

5.13 Altering Tables

The ALTER TABLE command is used to change definitions of existing tables. Usually, it is used to add columns to a table. Sometimes it is used to delete columns (depending on privileges) or change their sizes.

In general, in MySQL SQL, ALTER TABLE command is used :

- to add a column
- to add an integrity constraint
- to redefine a column (datatype, size, default value).

You can use ALTER TABLE **for adding columns** as follows :

```
ALTER TABLE <table name> ADD <column name> <data type><size> [<constraint name>] ;
```

For instance, to add a new column *tel_number* of type *integer* in table *Empl* you may give :

```
ALTER TABLE Empl
ADD (tel_number integer) ;
```

To *modify existing columns of table*, ALTER TABLE command can be used according to following syntax :

```
ALTER TABLE <tablename>
MODIFY (columname newdatatype (newsize) ) [FIRST|AFTER column] ;
```

To modify column *Job* of table *Empl* (given just before the *Assignment* at the end of this chapter) to have new width of 30 characters, you may give :

```
ALTER TABLE Empl
MODIFY (Job char(30) ) ;
```

Sometimes you may need to **change the name of one of your columns**. For this you can use CHANGE clause of ALTER TABLE command as per following syntax :

```
ALTER TABLE
CHANGE [COLUMN]old_col_name new_col_name column_definition ;
```

For instance, to change the existing column namely *First_Name* of table *Student*, to *FirstName* you may write :

```
ALTER TABLE Customers
CHANGE First_Name FirstName VARCHAR(20);
```

See, complete column description is given along with the new name.

5.14 Dropping Tables

The DROP TABLE command of *SQL* lets you drop a table from the database. The syntax for using a DROP TABLE command is :

```
DROP TABLE [IF EXISTS] <tablename>
```

That is, to drop a table *items*, you need to write :

```
DROP TABLE items ;
```

Once this command is given, the table name is no longer recognized and no more commands can be given on that object.

The IF EXISTS clause of DROP TABLE first checks whether the given table exists in the database or not. If it does, then it drops the mentioned table from the database. For instance, consider the following query :

```
DROP TABLE IF EXISTS players ;
```

The above query will first check for existence of *players* table in current database. If it exists, then it (table *players*) will be dropped from the database.

Check Point 5.1

1. What is SQL ? What are the different categories of SQL commands ?
2. What is a datatype ? Name some datatypes available in MySQL.
3. How would you calculate 13 * 15 in SQL ?
4. Is NULL value the same as 0 (zero) ? Write the reason for your answer.
5. Write the UPDATE command to increase the commission (Column name : COMM) by 500 of all the Salesmen who have achieved Sales (Column name : SALES) more than 200000. The table's name is COMPANY.
6. While using SQL pattern matching, what is the difference between '_' (underscore) and '%' wildcard symbols ?
7. Write one similarity and one difference between CHAR and VARCHAR data types. **[CBSE OD 15]**
8. Write SQL statement to display
   `Today, the date is <current date>`
9. Write a command to add a NOT NULL constraint on FEES column of a student table.
10. What is a constraint ? Name some constraints that you can apply to enhance database integrity.
11. What is primary key ? What is PRIMARY KEY constraint ?
12. What is NOT NULL constraint ? What is DEFAULT constraint ?
13. When a column's value is skipped in an INSERT command, which value is inserted in the database ?
14. Write MySql command to display the list of existing databases. **[CBSE D 12]**
15. Write MySql command will be used to open an already existing database "CONTACTS". **[CBSE OD 11]**

LET US REVISE

- *The basic elements of MySQL SQL are : literals, datatypes, nulls and comments.*
- *Literals are fixed data values.*
- *Data types of MySQL SQL include : NUMERIC (INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT, FLOAT(M, D), DOUBLE(M, D) AND DECIMAL(M, D), DATE and TIME (DATE, DATETIME, TIMESTAMP, TIME and YEAR(M) and STRING (CHAR(M), VARCHAR(M), BLOB or TEXT, TINYBLOB or TINYTEXT, MEDIUMBLOB or MEDIUMTEXT and ENUM).*
- *Column having no value is said to have NULL value.*
- *The SELECT command of SQL lets you make queries on the database.*
- *The DISTINCT keyword eliminates redundant data.*
- *To perform calculations, the expressions can be written along with SELECT without specifying any table name.*
- *A small work table* **Dummy** *can be used for making simple calculations.*
- *The* **curdate** *pseudo-column returns the current system date.*
- *The WHERE clause is used to select specific rows.*
- *The logical operators OR (\|\|) , AND (&&) and NOT (!) are used to connect search conditions in the WHERE clause.*
- *The BETWEEN operator is used for making range checks in queries.*
- *The IN operator is used for selecting values from a list of values.*
- *The LIKE operator is used for making character comparisons using strings.*

- The null value in a column can be searched for in a table using IS NULL in the WHERE clause.
- CREATE TABLE command is used to create tables in database.
- INSERT INTO command is used to insert data in the table.
- To insert data from other tables, subquery can be used inside INSERT INTO command.
- Existing data in tables can be changed with UPDATE command.
- Tuples in a table can be deleted using DELETE command.
- ALTER TABLE command is used to alter the definition of already created tables.
- With ALTER TABLE, new columns can be added, existing columns can be redefined.
- DROP TABLE command drops a table from a database.

Objective Type Questions — OTQs

Multiple Choice Questions

1. What is the full form of SQL ?
   (a) Structured Query Language (b) Structured Query List
   (c) Simple Query Language (d) None of these
2. What is the full form of DDL ?
   (a) Dynamic Data Language (b) Detailed Data Language
   (c) Data Definition Language (d) Data Derivation Language
3. What does DML stand for ?
   (a) Different Mode Level (b) Data Model Language
   (c) Data Mode Lane (d) Data Manipulation language
4. Which is the subset of SQL commands used to manipulate database structures, including tables ?
   (a) Data Definition Language (DDL)
   (b) Data Manipulation Language (DML)
   (c) Both (a) and (b) (d) None
5. Which of the following sublanguages of SQL is used to define the structure of the relation, deleting relations and relating schemas?
   (a) DML (Data Manipulation Language)
   (b) DDL (Data Definition Language)
   (c) Query (d) Relational Schema
6. Which of the following sublanguages of SQL is used to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database ?
   (a) DML (Data Manipulation Language)
   (b) DDL (Data Definition Language)
   (c) Query (d) Relational Schema
7. Which of the following keywords will you use in the following query to display the unique values of the column dept_name?

```
SELECT _____ dept_name FROM Company;
```

   (a) All (b) From (c) Distinct (d) Name

8. Which of the following keywords will you use in the following query to display all the values of the column dept_name ?

```
SELECT _____ dept_name FROM Company;
```

   (a) All (b) From (c) Distinct (d) Name
9. The _____ clause of SELECT query allows us to select only those rows in the result that satisfy a specified condition.
   (a) Where (b) from (c) having (d) like
10. Consider the following query :

```
SELECT name FROM class WHERE subject _____ NULL;
```

   Which comparison operator may be used to fill the blank space in above query ?
   (a) = (b) LIKE (c) IS (d) IS NOT
11. In SQL, which command is used to SELECT only one copy of each set of duplicable rows ?
   (a) SELECT DISTINCT (b) SELECT UNIQUE
   (c) SELECT DIFFERENT (d) All of the above

Fill in the Blanks

1. The SQL keyword _____ is used to specify the table(s) that contains the data to be retrieved.
2. To remove duplicate rows from the result of a query, specify the SQL qualifier _____ in select list.
3. To obtain all columns, use a(n) _____ instead of listing all the column names in the select list.
4. The SQL _____ clause contains the condition that specifies which rows are to be selected.
5. When two conditions must both be true for the rows to be selected, the conditions are separated by the SQL keyword _____ .
6. To refer to a set of values needed for a condition, we can use the SQL operator _____ .
7. To exclude one or more values (a list of values) using a condition, the SQL keyword _____ _____ should be used.
8. The SQL keyword _____ is used in SQL expressions to select based on patterns.

True/False Questions

1. The condition in a WHERE clause in a SELECT query can refer to only one value.
2. SQL provides the AS keyword, which can be used to assign meaningful column names to the results of queries using the SQL built-in functions.
3. SQL is a programming language.
4. SELECT DISTINCT is used if a user wishes to see duplicate columns in a query.
5. Data manipulation language (DML) commands are used to define a database, including creating, altering, and dropping tables and establishing constraints.
6. The keyword LIKE can be used in a WHERE clause to refer to a range of values.
7. The keyword BETWEEN can be used in a WHERE clause to refer to a range of values.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *What is SQL ? What are different categories of commands available in SQL ?*

Solution. In order to access data within the Oracle database, all programs and users must use, Structured Query Language (SQL). SQL is the set of commands that is recognised by nearly all RDBMSs.

SQL commands can be divided into following categories :

1. Data Definition Language (DDL) Commands.
2. Data Manipulation Language (DML) Commands.
3. Transaction Control Language (TCL) Commands.
4. Session Control Commands.
5. System Control Commands.

2. *Differentiate between DDL and DML commands.*

Solution. The Data Definition Language (DDL) commands, as the name suggests, allow you to perform tasks related to data definition. That is, through these commands, you can perform tasks like, create, alter and drop schema objects, grant and revoke privileges etc.

The Data Manipulation Language (DML) commands, as the name suggests, are used to manipulate data. That is, DML commands query and manipulate data in existing schema objects.

3. *Differentiate between CHAR and VARCHAR datatypes.*

Solution. The difference between CHAR and VARCHAR is that of *fixed length* and *variable length*. The CHAR datatype specifies a *fixed length* character string. When a column is given datatype as CHAR(*n*), then MySQL ensures that all values stored in that column have this length *i.e.*, *n* bytes. If a value is shorter than this length *n* then blanks are added, but the size of value remains *n* bytes.

VARCHAR, on the other hand, specifies a variable length string. When a column is given datatype as VARCHAR(*n*), then the maximum size a value in this column can have is *n* bytes. Each value that is stored in this column stores exactly as we specify it *i.e.*, no blanks are added if the length is shorter than maximum length *n*. However, if we exceed the maximum length *n*, then an error message is displayed.

4. *What is the "data type"? What are main objectives of datatypes ?*

Solution. Data type is defined as a set of values along with the operations that can be performed on those values.

Some common data types are *Integer, Float, Varchar, Char, String*, etc.

Main objectives of datatypes are :

- Optimum usage of storage space
- Represent all possible values
- Improve data integrity

5. (*a*) *Write two examples of DBMS software.* [CBSE D 2014]
(*b*) *What is meant by NULL value in MySQL ?*
(*c*) *Table 'Club' has 4 rows and 3 columns. Table 'Member' has 2 rows and 5 columns. What will be the cardinality of the Cartesian product of them ?*

Solution.

(*a*) (*i*) Oracle (*ii*) SQL Server (*iii*) MySQL

(*b*) Null value signifies a legal empty value.

(*c*) 8

6. *Consider the following table named "GARMENT". Write command of SQL for (i) to (iv).*

Table : GARMENT

GCODE	GNAME	SIZE	COLOUR	PRICE
111	TShirt	XL	Red	1400.00
112	Jeans	L	Blue	1600.00
113	Skirt	M	Black	1100.00
114	Ladies Jacket	XL	Blue	4000.00
115	Trousers	L	Brown	1500.00
116	Ladies Top	L	Pink	1200.00

(*i*) *To display names of those garments that are available in 'XL' size.*

(*ii*) *To display codes and names of those garments that have their names starting with 'Ladies'.*

(*iii*) *To display garment names, codes and prices of those garments that have price in the range 1000.00 to 1500.00 (both 1000.00 and 1500.00 included).*

(*iv*) *To change the colour of garment with code as 116 to "Orange".*

Solution.

(*i*)
```
SELECT GNAME
FROM GARMENT
WHERE SIZE = 'XL'
```

(*ii*)
```
SELECT GCODE, GNAME
FROM GARMENT
WHERE GNAME LIKE 'Ladies%' ;
```

(*iii*)
```
SELECT GNAME, GCODE, PRICE
FROM GARMENT
WHERE PRICE BETWEEN 1000.00 AND 1500.00 ;
```

(*iv*)
```
UPDATE garment
SET COLOUR = 'orange'
WHERE GCODE = 116 ;
```

7. *Consider the following table named "SOFTDRINK". Write commands of SQL for (i) to (iv).*

Table : SOFTDRINK

DRINKCODE	DNAME	PRICE	CALORIES
101	Lime and Lemon	20.00	120
102	Apple Drink	18.00	120
103	Nature Nectar	15.00	115
104	Green Mango	15.00	140
105	Aam Panna	20.00	135
106	Mango Juice Bahaar	12.00	150

(*i*) *To display names and drink codes of those drinks that have more than 120 calories.*

(ii) *To display drink codes, names and calories of all drinks, in descending order of calories.*

(iii) *To display names and price of drinks that have price in the range 12 to 18 (both 12 and 18 included).*

(iv) *Increase the price of all drinks in the given table by 10%.*

Solution.

(i)
```
SELECT DNAME, DRINKCODE
FROM SOFTDRINK
WHERE CALORIES > 120 ;
```

(ii)
```
SELECT DRINKCODE, DNAME, CALORIES
FROM SOFTDRINK
ORDER BY CALORIES DESC ;
```

(iii)
```
SELECT DNAME, PRICE
FROM SOFTDRINK
WHERE PRICE BETWEEN 12 and 18 ;
```

(iv)
```
UPDATE SOFTDRINK
SET PRICE = PRICE + 0.10 * PRICE ;
```

GLOSSARY

DataType	Means to identify type of data and associated operations for handling it.
Query	A command given to produce certain specified information from the database table(s).
DDL	Data Description Language Subset of SQL commands that are used to describe various entities of database.
DML	Data Manipulation Language Subset of SQL commands that are used to manipulate data in tables.

Consider a table Empl as given below. It will be required for many assignment questions.

empno	ename	job	mgr	hiredate	sal	comm	deptno
8369	SMITH	CLERK	8902	1990-12-18	800.00	NULL	20
8499	ANYA	SALESMAN	8698	1991-02-20	1600.00	300.00	30
8521	SETH	SALESMAN	8698	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	20
8654	MOMIN	SALESMAN	8698	1991-09-28	1250.00	1400.00	30
8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	30
8882	SHIAVNSH	MANAGER	8839	1991-06-09	2450.00	NULL	10
8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	20
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	10
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500.00	0.00	30
8886	ANOOP	CLERK	8888	1993-01-12	1100.00	NULL	20
8900	JATIN	CLERK	8698	1991-12-03	950.00	NULL	30
8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	20
8934	MITA	CLERK	8882	1992-01-23	1300.00	NULL	10

Assignment

1. How are SQL commands classified ?
2. Differentiate between DDL and DML commands.
3. (a) What is the use of UPDATE statement in SQL ? How is it different from ALTER statement ?
   (b) Mr. Shankar created a table VEHICLE with 3 rows and 4 columns. He added 1 more row to it and deleted one column. What is the Cardinality and Degree of the Table VEHICLE ?
   (c) Consider the following table named "GYM" with details about fitness items being sold in the store. Write command of SQL for (i) to (iv).

Table : GYM

ICODE	INAME	PRICE	BRANDNAME
G101	Power Fit Exerciser	20000	Power Gymea
G102	Aquafit Hand Grip	1800	Reliable
G103	Cycle Bike	14000	Ecobike
G104	Protoner Extreme Gym	30000	Coscore
G105	Message Belt	5000	Message Expert
G106	Cross Trainer	13000	GTC Fitness

(i) To display the names of all the items whose name starts with "A".

(ii) To display ICODEs and INAMEs of all items, whose Brandname is Reliable or Coscore.

(iii) To change the Brandname to "Fit Trend India" of the item, whose ICODE as "G101".

Add a new row for new item in GYM with the details :
"G107", "Vibro exerciser", 21000, "GTCFitness" [CBSE D 2013]

4. (a) Mr. James created a table **CLIENT** with 2 rows and 4 columns. He added 2 more rows to it and deleted one column. What is the Cardinality and Degree of the Table **CLIENT** ?

   (b) Consider the following table **FITNESS** with details about fitness products being sold in the store. Write command of SQL for (i) to (iv).

Table : FITNESS

PCODE	PNAME	PRICE	Manufacturer
P1	Treadmill	21000	Coscore
P2	Bike	20000	Aone
P3	Cross Trainer	14000	Reliable
P4	Multi Gym	34000	Coscore
P5	Massage Chair	5500	Regrosene
P6	Belly Vibrator Belt	6500	Ambawya

(i) To display the names of all the products with price more than 20000.

(ii) To display the names of all products by the manufacturer "Aone".

(*iii*) To change the price data of all the products by applying 25% discount reduction.

(*iv*) To add a new row for product with the details :

"P7", "Vibro Exerciser", 28000, "Aone".

5. Write SQL commands for the following on the basis of given table CLUB. [CBSE OD 2013]

Table : CLUB

COACH_ID	COACHNAME	AGE	SPORTS	DATOFAPP	PAY	SEX
1.	KUKREJA	35	KARATE	27/03/1996	1000	M
2.	RAVINA	34	KARATE	20/01/1998	1200	F
3.	KARAN	34	SQUASH	19/02/1998	2000	M
4.	TARUN	33	BASKETBALL	01/01/1998	1500	M
5.	ZUBIN	36	SWIMMING	12/01/1998	750	M
6.	KETAKI	36	SWIMMING	24/02/1998	800	F
7.	ANKITA	39	SQUASH	20/02/1998	2200	F
8.	ZAREEN	37	KARATE	22/02/1998	1100	F
9.	KUSH	41	SWIMMING	13/01/1998	900	M
10.	SHAILYA	37	BASKETBALL	19/02/1998	1700	M

(*a*) To show all information about the swimming coaches in the club.

(*b*) To list names of all coaches with their date of appointment (DATOFAPP) in descending order.

(*c*) To display a report, showing coachname, pay, age and bonus (15% of pay) for all the coaches.

6. Write SQL commands for the following on the basis of given table STUDENT.

Table : STUDENT1

No.	Name	Stipend	Stream	AvgMark	Grade	Class
1	Karan	400.00	Medical	78.5	B	12B
2	Divakar	450.00	Commerce	89.2	A	11C
3	Divya	300.00	Commerce	68.6	C	12C
4	Arun	350.00	Humanities	73.1	B	12C
5	Sabina	500.00	Nonmedical	90.6	A	11A
6	John	400.00	Medical	75.4	B	12B
7	Robert	250.00	Humanities	64.4	C	11A
8	Rubina	450.00	Nonmedical	88.5	A	12A
9	Vikas	500.00	Nonmedical	92.0	A	12A
10	Mohan	300.00	Commerce	67.5	C	12C

(*a*) Select all the Nonmedical stream students from STUDENT1.

(*b*) List the names of those students who are in class 12 sorted by Stipend.

(*c*) List all students sorted by AvgMark in descending order.

7. What is foreign key ? How do you define a foreign key in your table ?
8. How is FOREIGN KEY commands different from PRIMARY KEY command ?
9. How is FOREIGN KEY commands related to the PRIMARY KEY ?
10. What are table constraints ? What are column constraints ? How are these *two* different ?
11. Insert all those records of table Accounts into table *Pending* where amt_outstanding is more than 10000.
12. Increase salary of employee recrods by 10% (*table employee*).
13. Add a constraint (NN-Grade) in table Empl (given before assignment) that declares column Grade not null.
14. Drop the table Empl.
15. Differentiate between :

(*i*) DROP TABLE, DROP DATABASE

(*ii*) DROP TABLE, DROP clause of ALTER TABLE.

16. Mr. Mittal is using a table with following columns :

Name, Class, Stream_Id, Stream_name

He needs to display names of students who have not been assigned any stream or have been assigned stream_name that ends with "computers".

He wrote the following command, which did not give the desired result.

```
SELECT Name, Class FROM Students
WHERE Stream_name = NULL OR Stream_name = "%computers" ;
```

Help Mr. Mittal to run the query by removing the error and write correct query. [CBSE D 12]

17. The Doc_name Column of a table Hospital is given below : [CBSE OD 11]

Doc_name
Avinash
Hariharan
Vinayak
Deepak
Sanjeev

Based on the information, find the output of the following queries :

(*i*) SELECT doc_name FROM HOSPITAL WHERE Doc_name like "%v";

(*ii*) SELECT doc_name FROM HOSPITAL WHERE doc_name like "%e%";

18. Sarthak, a student of class XII, created a table "Class". Grade is one of the columns of this table. To find the details of students whose Grades have not been entered, he wrote the following MySql query, which did not give the desired result : [CBSE OD 11]

```
SELECT * FROM Class WHERE Grade = "Null";
```

Help Sarthak to run the query by removing the errors from the query and write the correct query.

19. What is the purpose of DROP TABLE command in MySql ? How is it different from DELETE command ? [CBSE D 11]

6 MySQL Functions

Outline

6.1 Introduction
6.2 Text Functions
6.3 Numeric Functions
6.4 Date/Time Functions

6.1 Introduction

Data can be manipulated in queries through functions supported by SQL. A **function** is a special type of predefined command set that performs some operation and returns a single value. Functions operate on zero, one, two or more values that are provided to them. The values that are provided to functions are called **parameters** or **arguments**.

This chapter is dedicated to the discussion of various MySQL functions.

The MySQL functions have been categorised into various categories, such as *Text functions, Mathematical functions, Date and Time functions* and so on. Although MySQL offers a big bouquet of functions, yet in this discussion we shall remain limited to the functions as mentioned in your syllabus.

FUNCTION

"A **Function** is a special type of predefined command set that performs some operation and returns a single value."

6.2 Text/String Functions

The string functions of MySQL can manipulate the text string in many ways. Some commonly used string functions are being discussed below.

1. CHAR( )

This function returns the character for each integer passed.

Syntax

```
CHAR(value1 [, value2 ...])
```

- returns a string made up of the ASCII representation of the decimal value list.
- Strings in numeric format are converted to a decimal value.
- Null values are ignored.

Argument types : Integers *Return Value* : String

EXAMPLE 1 *Write a query to create a string from the ASCII values 70, 65, 67, 69.*

SOLUTION

```
mysql> SELECT CHAR (70, 65, 67, 69 ) ;
+----------------------------+
| char (70, 65, 67, 69 )     |
+----------------------------+
| FACE                       |
+----------------------------+
1 row in set (0.00 sec)
```

Even if an argument contains a fractional value or a string type number, they are considered as integers. Consider the following example.

EXAMPLE 2 *Write a query to create a string from the ASCII values 65, 67.3, '68.3'.*

SOLUTION

```
mysql> SELECT CHAR (65, 67.3, '69.3' ) ;
+----------------------------+
| char (65, 67.3, '69.3' )   |
+----------------------------+
| ACE                        |
+----------------------------+
1 row in set, 1 warning (0.00 sec)
```

See, the fractional value 67.3 is treated as 67 (character C) and the string type number '69.3' is treated as 69

2. CONCAT

This function concatenates two strings.

Syntax

```
CONCAT (str1, str2, ...)
```

- Returns argument ***str1*** concatenated with argument ***str2***.

Argument types : String, String *Return value* : String

EXAMPLE 3 *Concatenate name and aggregate for students having age as 14 or 16.*

Table : *Student*

name	age	sex	matric	aggregate
Abu Bakar	15	M	9531185	456
Aanya	16	F	9531186	340
Gurvinder	14	F	9531187	480
Ali	16	M	9531188	260
Michelle	15	F	9531188	321
Zubin	15	M	9531184	412
Simran	15	F	9531183	378
Fatimah	14	F	9531182	400
Anup	15	M	9531181	302
Mita	16	F	9531180	150

SOLUTION

```
mysql> SELECT CONCAT(name, aggregate) AS "Name Marks" FROM student
    -> WHERE age = 14 OR age = 16 ;
+----------------+
| Name Marks     |
+----------------+
| Aanya340       |
| Gurvinder480   |
| Ali260         |
| Fatimah400     |
| Mita150        |
+----------------+
5 rows in set (0.00 sec)
```

Example 4 uses nesting (*i.e.*, function inside other function) to concatenate three character strings.

EXAMPLE 4 *Consider table Empl given in before Assignment of chapter 5. Display the job for employee 8900 in the format <Name> is a <Job>.*

SOLUTION

```
mysql> SELECT CONCAT ( CONCAT ( ename, "is a"), job )
    -> FROM empl
    -> WHERE empno = 8900;
+-----------------------------------------+
| concat ( concat ( ename, "is a"), job ) |
+-----------------------------------------+
| JATIN is a CLERK                        |
+-----------------------------------------+
1 row in set (0.00 sec)
```

3. LOWER / LCASE

This function converts a string into lowercase.

Syntax

```
LOWER(str)
LCASE(str)
```

- Returns the argument ***str***, with all letters in lowercase.
- The return value has the same data type as the argument *char* (CHAR or VARCHAR).

Argument type : String *Return value* : String

EXAMPLE 5 *Display names 'MR. OBAMA' and 'MS. Gandhi' into lowercase.*

SOLUTION

```
mysql> SELECT LOWER('MR. OBAMA') AS "LowerName1" ,
    -> LOWER('Ms. Gandhi') AS "LowerName2" ;
```

Or

```
mysql> SELECT LCASE('MR. OBAMA') AS "LowerName1" ,
    -> LCASE ('Ms. Gandhi') AS "LowerName2" ;
```

```
+------------+--------------+
| LowerName1 | LowerName2   |
+------------+--------------+
| mr. obama  | ms. gandhi   |
+------------+--------------+
1 row in set (0.00 sec)
```

EXAMPLE 6 *Display names of all employees in lowercase. (Consider table Empl of Chapter 5)*

SOLUTION

```
mysql> SELECT LOWER(ename) "employeeName"
    -> FROM empl ;
```

Or

```
mysql> SELECT LCASE(ename) "employeeName"
    -> FROM empl ;
```

```
+--------------------+
| employeeName       |
+--------------------+
| smith              |
| anya               |
| seth               |
| mahadevan          |
| momin              |
| bina               |
| shiavnsh           |
| scott              |
| amir               |
| kuldeep            |
| anoop              |
| jatin              |
| fakir              |
| mita               |
+--------------------+
14 rows in set (0.00 sec)
```


4. SUBSTR

This function extracts a substring from a given string.

Syntax

```
(i) SUBSTR(str, m [, n]) OR (ii) SUBSTR(str FROM m [FOR n])
```

- Returns a portion of *str*, beginning at character *m*, *n* characters long. If *m* is 0, it is treated as 1. If *m* is positive, MySQL counts from the beginning of *str* to find the first character. If *m* is negative, MySQL counts backwards from the end of *str*. If *n* is omitted, MySQL returns all characters to the end of *str*. If *n* is less than 1, a null is returned.
- Floating point numbers passed as arguments to *substr* are automatically converted to integers.

Argument type: String, Numeric [, Numeric] *Return value* : String

You may also write SUBSTRING in place of SUBSTR, both work identically.

EXAMPLE 7 *Display 4 characters extracted from 3rd left character onwards from string 'ABCDEFG'.*

SOLUTION

```
mysql> SELECT SUBSTR('ABCDEFG', 3, 4) "Subs" ;
```

```
+------------+
| Subs       |
+------------+
| CDEF       |
+------------+
```

EXAMPLE 8 *Display 4 characters extracted from 5th right character onwards from string 'ABCDEFG'.*

SOLUTION

```
mysql> SELECT SUBSTR('ABCDEFG', -5, 4) "Subs" ;
```

```
+------------+
| Subs       |
+------------+
| CDEF       |
+------------+
```

EXAMPLE 9 *Display first three characters extracted from jobs of employees 8888 and 8900.*

SOLUTION

```
mysql> SELECT SUBSTR(job, 1, 3)
    -> FROM empl
    -> WHERE empno = 8888 or empno = 8900 ;
```

```
+--------------------+
| SUBSTR(job, 1, 3)  |
+--------------------+
| ANA                |
| CLE                |
+--------------------+
```

5. UPPER / UCASE

This function converts the given string into upper case.

Syntax

```
-UPPER(str)        -UCASE(str)
```

- Returns argument ***str***, with all letters uppercase.
- The return value has the same data type as the argument *str*.

Argument type : String *Return value* : String

EXAMPLE 10 *Convert and display string 'Large' into uppercase.*

SOLUTION

```
mysql> SELECT UPPER('Large') "Uppercase" ;
```

Or

```
mysql> SELECT UCASE('Large') "Uppercase" ;
```

```
+-----------------+
| Uppercase       |
+-----------------+
| LARGE           |
+-----------------+
```

6. LTRIM

This function removes leading spaces *i.e.*, spaces from the left of given string.

Syntax

```
LTRIM(str)
```

- Removes spaces from the left of argument ***str*** with initial characters removed.

Argument type : String *Return value* : String

EXAMPLE 11 *Write a query to remove leading spaces of string ' RDBMS MySQL'.*

SOLUTION

```
mysql> SELECT LTRIM('  RDBMS MySQL') ;
```

```
+----------------------------+
| ltrim('  RDBMS MySQL')     |
+----------------------------+
| RDBMS MySQL                |
+----------------------------+
1 row in set (0.05 sec)
```

Notice that LTRIM( ) removes only the leading spaces. Spaces in the middle or trailing spaces remain untouched with LTRIM( ).

7. RTRIM

This function removes trailing spaces *i.e.*, spaces from the right of given string.

Syntax

```
RTRIM(str)
```

- Returns ***str***, with trailing spaces removed after the last character.

Argument type : String *Return value* : String

EXAMPLE 12 *Write a query to remove leading spaces of string ' RDBMS MySQL '.*

SOLUTION

```
mysql> SELECT RTRIM('  RDBMS MySQL  );
```

```
+-----------------------------+
| rtrim('  RDBMS MySQL  ')    |
+-----------------------------+
| RDBMS MySQL                 |
+-----------------------------+
1 row in set (0.00 sec)
```

Notice that RTRIM( ) removes only the trailing spaces. Spaces in the middle or leading spaces remain untouched with RTRIM( ).

8. TRIM

This function removes leading and trailing spaces from a given string. It performs combined functions of *LTRIM( )* and *RTRIM( )*.

Syntax

```
TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRM([remstr FROM]str)
```

- Returns the string ***str*** with all ***remstr*** prefixes or suffixes removed.
- If none of the specifiers BOTH, LEADING, or TRAILING is given, BOTH is assumed.
- ***remstr*** is optional and, if not specified, spaces are removed.

Argument types : String *Return Value* : String

EXAMPLE 13 *Write a query to remove leading and trailing spaces from string ' Bar One '.*

SOLUTION

```
mysql> SELECT TRIM('  Bar One  ');
```

```
+-----------------------+
| TRIM('  Bar One  ')   |
+-----------------------+
| Bar One               |
+-----------------------+
1 row in set (0.00 sec)
```

EXAMPLE 14 Write a query to remove leading spaces from string ' Bar One '. Make use of TRIM() function only.

SOLUTION

```
mysql> SELECT TRIM(leading ' ' FROM '  Bar One  ');
```

```
+-----------------------------------+
| TRIM(leading ' ' from '  Bar One  ') |
+-----------------------------------+
| Bar One                           |
+-----------------------------------+
1 row in set (0.00 sec)
```

EXAMPLE 15 *Write a query to remove leading x characters from string 'xxxxBar One xxxxx'.*

SOLUTION.
```
mysql> SELECT TRIM(LEADING 'x' FROM 'xxxxBar One xxxxx') ;
+-----------------------------------------+
| TRIM(LEADING 'x' FROM 'xxxxBar One xxxxx') |
+-----------------------------------------+
| Bar One xxxxx                           |
+-----------------------------------------+
1 row in set (0.00 sec)
```

EXAMPLE 16 *Write a query to remove both leading and trailing x characters from string 'xxxxBar One xxxxx'.*

SOLUTION
```
mysql> SELECT TRIM(BOTH 'x' FROM 'xxxxBar One xxxxx') ;
+-----------------------------------------+
| TRIM(both 'x' FROM 'xxxxBar One xxxxx') |
+-----------------------------------------+
| Bar One                                 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

EXAMPLE 17 *Write a query to remove trailing x characters from string 'xxxxBar One xxxxx'.*

SOLUTION.
```
mysql> SELECT TRIM(TRAILING 'x' FROM 'xxxxBar One xxxxx') ;
+-----------------------------------------+
| TRIM(TRAILING 'x' FROM 'xxxxBar One xxxxx') |
+-----------------------------------------+
| xxxxBar One                             |
+-----------------------------------------+
1 row in set (0.00 sec)
```

9. INSTR

This function searches for given second string into the given first string.

Syntax

```
INSTR(str1, str2)
```

- Searches ***str1*** for ***str2*** and returns the position

Argument type : String, String *Return value* : Number

EXAMPLE 18 *Display the position of occurrence of string 'OR' in string 'CORPORATE FLOOR'.*

SOLUTION.
```
mysql> SELECT INSTR('CORPORATE FLOOR', 'OR') AS Instring ;
+----------+
| Instring |
+----------+
| 2        |
+----------+
1 row in set (0.03 sec)
```


EXAMPLE 19 *Display the position of string 'LE' in field Job of table Empl.*

SOLUTION
```
mysql> SELECT job, INSTR(job, 'LE') AS "LE in Job" FROM empl;
+-----------+------------+
| job       | LE in Job  |
+-----------+------------+
| CLERK     | 2          |
| SALESMAN  | 3          |
| SALESMAN  | 3          |
| MANAGER   | 0          |
| SALESMAN  | 3          |
| MANAGER   | 0          |
| MANAGER   | 0          |
| ANALYST   | 0          |
| PRESIDENT | 0          |
| SALESMAN  | 3          |
| CLERK     | 2          |
| CLERK     | 2          |
| ANALYST   | 0          |
| CLERK     | 2          |
+-----------+------------+
14 rows in set (0.02 sec)
```

10. LENGTH

This function returns the length of a given string in bytes.

Syntax

```
LENGTH(str)
```

- Returns the length of parameter specified by argument ***str*** in characters.
- If argument ***str*** has datatype CHAR, the length includes all trailing blanks.
- If argument ***str*** is null, his function returns null.

Argument type : String *Return value* : Number

EXAMPLE 20 *How many characters are there in string 'CANDIDE'.*

SOLUTION
```
mysql> SELECT LENGTH('CANDIDE') "Length in characters" ;
+----------------------+
| Length in characters |
+----------------------+
| 7                    |
+----------------------+
```

EXAMPLE 21 *Display the employee names and the name-lengths for employees 8888 and 8900.*

SOLUTION

```
mysql> SELECT ename, LENGTH(ename) AS "Name Length" FROM empl
    -> WHERE empno = 8888 OR empno = 8900 ;

+------------+---------------+
| ename      | Name Length   |
+------------+---------------+
| SCOTT      | 5             |
| JATIN      | 5             |
+------------+---------------+
2 rows in set (0.00 sec)
```

11. LEFT

This function returns the leftmost number of characters as specified.

Syntax

```
LEFT(str,len)
```

- Returns the leftmost ***len*** characters from the string ***str***.
- Returns NULL if any argument is NULL.

Argument types : String, integer *Return Value* : String

EXAMPLE 22 *Write a query to extract institute code from a string 'USS/23/67/09'. The first three characters tell the institute code.*

SOLUTION

```
mysql> SELECT LEFT('USS/23/67/09', 3) ;

+--------------------------+
| left('USS/23/67/09', 3)  |
+--------------------------+
| USS                      |
+--------------------------+
1 row in set (0.00 sec)
```

12. RIGHT

This function returns the rightmost number of characters as specified.

Syntax

```
RIGHT(str,len)
```

- Returns the rightmost ***len*** characters from the string ***str***.
- Returns NULL if any argument is NULL.

Argument types : String, integer *Return Value* : String

EXAMPLE 23 *Write a query to extract 2 digit year from a string 'USS/23/67/09'. The last two characters tell the year.*

SOLUTION

```
mysql> SELECT RIGHT('USS/23/67/09', 2) ;

+--------------------------+
| right('USS/23/67/09', 2) |
+--------------------------+
| 09                       |
+--------------------------+
1 row in set (0.00 sec)
```

13. MID

This function returns a substring starting from the specified position.

Syntax

```
MID(str, pos, len)
```

- returns a substring from ***str*** starting from ***pos*** and having number of characters as ***len***.
- *MID(str,pos,len)* is a synonym for *SUBSTRING(str,pos,len)*.

Argument types : integers *Return Value* : String

EXAMPLE 24 *Write a query to extract a sub string from string **'Quadratically'** which should be 6 characters long and start from 5th character of the given string.*

SOLUTION

```
mysql> SELECT SUBSTRING('Quadratically',5,6);
```

Or

```
mysql> SELECT MID('Quadratically',5,6);

+-----------------------------------+
| SUBSTRING('Quadratically',5,6)    |
+-----------------------------------+
| ratica                            |
+-----------------------------------+
1 row in set (0.02 sec)
```

6.3 Numeric Functions

The number functions are those functions that accept numeric values and after performing the required operation, return numeric values.

Some useful numeric functions are being discussed below :

1. MOD

This function returns modulus (*i.e.*, remainder) of given two numbers.

Syntax

```
MOD(m, n), M % N , M MOD N
```

- Returns remainder of argument ***m*** divided by argument ***n***.
- Returns ***m*** if ***n*** is 0 *i.e.*, if denominator is 0.

Argument type : Numeric, Numeric *Return value* : Numeric

EXAMPLE 25 *Find out the remainder of 11 divided by 4.*

SOLUTION

```
mysql> SELECT MOD(11, 4) "Modulus" ;
```

```
+-----------+
| Modulus   |
+-----------+
| 3         |
+-----------+
```

Remainder of 11/4

2. POWER / POW

This function returns m^n *i.e.,* a number *m* raised to the *n*th power.

Syntax

```
POWER(m, n)  or  POW(m, n)
```

- Returns value of argument ***m*** raised to the ***n***th power. The base ***m*** and the exponent ***n*** can be any numbers, but if ***m*** is negative, ***n*** must be an integer.

Argument type : Numeric, Numeric *Return value* : Numeric

EXAMPLE 26 *Display the result of* 3^2.

SOLUTION

```
mysql> SELECT POWER(3, 2) "Raised" ;
```

```
+-----------+
| Raised    |
+-----------+
| 9         |
+-----------+
```

result of 3^2

3. ROUND

This function returns a number rounded off as per given specifications.

Syntax

```
ROUND(n[,m])
```

- Returns value of argument **n** rounded to **m** places right of the decimal point ; if **m** is omitted to 0 place, **m** can be negative to round off digits left of the decimal point. **m** must be an integer.

Argument type : Numeric, Numeric *Return value* : Numeric

EXAMPLE 27 *Round off value 15. 193 to one decimal place.*

SOLUTION

```
mysql> SELECT ROUND(15.193, 1) "Round" ;
```

```
+-----------+
| Round     |
+-----------+
| 15.2      |
+-----------+
```

rounded to 1 decimal place

EXAMPLE 28 *Round off value 15.193 to nearest ten's.*

SOLUTION

```
mysql> SELECT ROUND(15.193, -1) "Round" ;
```

```
+-----------+
| Round     |
+-----------+
| 20        |
+-----------+
```

rounded to one place left to the decimal i.e., to nearest 10s

4. SIGN

This function returns sign of a given number.

Syntax

```
SIGN(n)
```

- If argument **n** < 0, the function returns –1 ;
- if argument **n** = 0, the function returns 0 ;
- if argument **n** > 0, the function returns 1.

Argument type : Numeric *Return value* : Numeric

EXAMPLE 29 *Find out the sign of value –15.*

SOLUTION

```
mysql> SELECT SIGN(-15) "Sign" ;
```

```
+-----------+
| Sign      |
+-----------+
| -1        |
+-----------+
```

Since the value – 15 is less than 0, it has returned – 1

EXAMPLE 30 *Find out the sign of value 15.*

SOLUTION

```
mysql> SELECT SIGN(15) "Sign" ;
```

```
+-----------+
| Sign      |
+-----------+
| 1         |
+-----------+
```

Since the value 15 is more than 0, it has returned 1

5. SQRT

This function returns the square root of given number.

Syntax

```
SQRT(n)
```

- Returns square root of argument ***n***. The value ***n*** cannot be negative.
- SQRT returns a "real" result.

Argument type : Numeric *Return value* : Numeric

EXAMPLE 31 *Find out the square root of value 26.*

SOLUTION

```
mysql> SELECT SQRT(26) "Square root" ;

+---------------+
| Square root   |
+---------------+
| 5.09901951    |
+---------------+
```

this is square root of 26

6. TRUNCATE

This function returns a number with some digits truncated.

Syntax

```
TRUNCATE(n, m)
```

- Returns value of argument ***n*** truncated to argument ***m*** decimal places ;
- if argument ***m*** is given as 0 places, ***m*** can be negative to truncate (*i.e.*, make zero) *m* digits left of the decimal point.

Argument type : Numeric, Numeric *Return value* : Numeric

EXAMPLE 32 *Truncate value 15.79 to 1 decimal place.*

SOLUTION

```
mysql> SELECT TRUNCATE(15.79, 1) "Truncate" ;

+------------+
| Truncate   |
+------------+
| 15.7       |
+------------+
```

EXAMPLE 33 *Truncate value 15.79 to ten's place.*

SOLUTION

```
mysql> SELECT TRUNCATE(15.79, -1) "Truncate" ;

+------------+
| Truncate   |
+------------+
| 10         |
+------------+
```

NOTE

Please note there is difference between round( ) and truncate( ) functions. Truncate function just truncates the digits without rounding them off.

6.4 Date/Time Functions

Date functions operate on values of the DATE datatype.

1. CURDATE( ) / CURRENT_DATE( )

This function returns the current date.

Syntax

```
CURDATE( )
```

Or `CURRENT_DATE( )`

Or `CURRENT_DATE`

- Returns the current date as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on whether the function is used in a string or numeric context.
- CURRENT_DATE and CURRENT_DATE( ) is the synonym for CURDATE( ).

Argument Type : None *Return Type* : Date

EXAMPLE 34 *Write a query to display current date on your system.*

SOLUTION

```
mysql> SELECT CURDATE( ) ;

+---------------+
| CURDATE()     |
+---------------+
| 2020-04-03    |
+---------------+
1 row in set (0.00 sec)
```

EXAMPLE 35 *Write a query to display date after 10 days of current date on your system.*

SOLUTION

```
mysql> SELECT CURDATE( ) + 10 ;

+---------------------+
| CURDATE() + 10      |
+---------------------+
| 20200413            |
+---------------------+
1 row in set (0.00 sec)
```

when you add a number to a date, then returned value is not formatted as date.

2. DATE( )

This function extracts the date part of a date or datetime expression

Syntax

```
DATE(expr)
```

- Extracts the date part of the date or datetime expression *expr*

Argument types : date or datetime *Return Value* : date

EXAMPLE 36 *Write a query to extract date from a given datetime value '2008-12-31 01:02:03'.*

SOLUTION

```
mysql> SELECT DATE('2008-12-31 01:02:03') ;
+-----------------------------+
| DATE('2008-12-31 01:02:03') |
+-----------------------------+
| 2008-12-31                  |
+-----------------------------+
1 row in set (0.03 sec)
```

3. MONTH( )

This function returns the month from the date passed.

Syntax

```
MONTH(date)
```

- Returns the month for *date*, in the range 1 to 12 for January to December.
- Returns 0 for dates such as '0000-00-00' or '2008-00-00' that have a zero month part.

Argument types : date *Return Value* : integer

EXAMPLE 37 *Write a query to extract month part from date 3rd Feb 2009.*

SOLUTION

```
mysql> SELECT MONTH('2009-02-03') ;
+----------------------+
| MONTH('2009-02-03')  |
+----------------------+
| 2                    |
+----------------------+
1 row in set (0.03 sec)
```

4. MONTHNAME( )

This function returns the name of the month for a date.

Syntax

```
MONTHNAME(date)
```

- Returns the name of the month of the given date.

Argument types : date *Return Value* : string

EXAMPLE 38 *Write a query to display name of the month for date 01-05-2020.*

SOLUTION

```
mysql> SELECT MONTHNAME('2020-05-01') ;
+-------------------------+
| MONTHNAME('2020-05-01') |
+-------------------------+
| May                     |
+-------------------------+
1 row in set (0.02 sec)
```


5. DAY( )

This function returns the day part of a date.

Syntax

```
DAY (date)
```

- Returns the day part of the given date.

Argument types : date *Return Value* : integer

EXAMPLE 39 *Write a query to display day from date 13-Apr-2020.*

SOLUTION

```
mysql> SELECT DAY('2020-04-13') ;
+--------------------------+
| DAY ('2020-04-13')       |
+--------------------------+
| 13                       |
+--------------------------+
1 row in set (0.00 sec)
```

6. YEAR( )

This function returns the year part of a date.

Syntax

```
YEAR(date)
```

- Returns the year for *date*, in the range 1000 to 9999.
- Returns 0 for the "zero" date.

Argument types : date *Return Value* : integer

EXAMPLE 40 *Write a query to extract year part from date 3rd Feb 2009.*

SOLUTION

```
mysql> SELECT YEAR('2009-02-03') ;
+-----------------------+
| year('2009-02-03')    |
+-----------------------+
| 2009                  |
+-----------------------+
1 row in set (0.02 sec)
```

7. DAYNAME( )

This function returns the name of weekday.

Syntax

```
DAYNAME(date)
```

- Returns the name of the weekday for *date*.

Argument types : date *Return Value* : String

EXAMPLE 41 *Write a query to display name of weekday for date 3rd Feb 2009.*

SOLUTION

```
mysql> SELECT DAYNAME('2009-02-03') ;
+------------------------+
| DAYNAME('2009-02-03')  |
+------------------------+
| Tuesday                |
+------------------------+
1 row in set (0.05 sec)
```

8. DAYOFMONTH ( )

This function returns the day of month.

Syntax

```
DAYOFMONTH (date)
```

- Returns the day of the month for *date* in the range 1 to 31.
- Returns or 0 for dates such as '0000-00-00' or '2008-00-00' that have a zero day part.

Argument types : date *Return Value :* integer

EXAMPLE 42 Write a query to display day of the month for date 3rd Feb 2009.

SOLUTION

```
mysql> SELECT DAYOFMONTH('2009-02-03') ;
+---------------------------+
| DAYOFMONTH('2009-02-03')  |
+---------------------------+
| 3                         |
+---------------------------+
1 row in set (0.00 sec)
```

9. DAYOFWEEK( )

This function returns the day of week.

Syntax

```
DAYOFWEEK(date)
```

- Returns the weekday index for *date* (1 = Sunday, 2 = Monday, ..., 7 = Saturday).

Argument types : date *Return Value :* integer

EXAMPLE 43 *Write a query to display day of the week for date 13th Feb 2009.*

SOLUTION

```
mysql> SELECT DAYOFWEEK('2009-02-13') ;
+--------------------------+
| DAYOFWEEK('2009-02-13)   |
+--------------------------+
| 6                        |
+--------------------------+
1 row in set (0.01 sec)
```


10. DAYOFYEAR( )

This function returns the day of the year.

Syntax

```
DAYOFYEAR(date)
```

- Returns the day of the year for *date*, in the range 1 to 366.

Argument types : date *Return Value :* integer

EXAMPLE 44 *Write a query to display day of the year for date 13th Feb 2009.*

SOLUTION

```
mysql> SELECT DAYOFYEAR('2009-02-13') ;
+----------------------------+
| DAYOFYEAR('2009-02-13')    |
+----------------------------+
| 44                         |
+----------------------------+
1 row in set (0.00 sec)
```

11. NOW( )

This function returns the current date and time.

Syntax

```
NOW( )
```

- Returns the current date and time as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uuuuuu format, depending on whether the function is used in a string or numeric context.
- The value is expressed in the current time zone.
- **Now( )** returns a constant time that indicates the time at which the statement began to execute.

Argument types : none *Return Value :* datetime

EXAMPLE 45 *Write a query to display current date and time.*

SOLUTION

```
mysql> SELECT NOW( ) ;
+---------------------+
| NOW()               |
+---------------------+
| 2009-06-03 15:27:20 |
+---------------------+
1 row in set (0.00 sec)
```

Check Point 6.1

1. Which function can be used to concatenate two strings ?
2. Which function(s) can be used for extracting a substrings ?
3. What is the difference between Trim( ) and Rtrim( ) ?
4. What is the difference between round( ) and truncate( ) ?
5. Which functions returns the current date and current time ?
6. What is the difference between sysdate( ) and now( ) functions ?

12. SYSDATE( )

This function returns the time at which the function executes.

Syntax

```
SYSDATE( )
```

- Returns the current date and time as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS.uuuuuu format, depending on whether the function is used in a string or numeric context.
- SYSDATE( ) returns the time at which it executes. This differs from the behavior for NOW( ), which returns a constant time that indicates the time at which the statement began to execute.

Argument types : none *Return Value :* datetime

Consider the following example that illustrates the difference between *now( )* and *sysdate( ).*

EXAMPLE 46 *Write a query to illustrate the difference between now( ) and sysdate( ).*

SOLUTION

```
mysql> SELECT NOW(), SLEEP(2), NOW();
+---------------------+----------+---------------------+
| NOW()               | SLEEP(2) | NOW()               |
+---------------------+----------+---------------------+
| 2009-06-03 15:32:34 | 0        | 2009-06-03 15:32:34 |
+---------------------+----------+---------------------+
1 row in set (2.01 sec)
```

Notice that in first query the now() gives the same value even after 2 seconds (sleep(2)), but sysdate() values have a difference of 2 seconds because of sleep(2)

```
mysql> SELECT SYSDATE(), SLEEP(2), SYSDATE();
+---------------------+----------+---------------------+
| SYSDATE()           | SLEEP(2) | SYSDATE()           |
+---------------------+----------+---------------------+
| 2009-06-03 15:33:15 | 0        | 2009-06-03 15:33:17 |
+---------------------+----------+---------------------+
1 row in set (2.01 sec)
```

Compare the above *two* results. The *now( )* function will return the same result even if you use it multiple times in same statement, even after creating delays (such as through *sleep( )* that creates delay of specified seconds) because the *now( )* function returns the begin-time of statement, whereas *sysdate( )* function returns the execution time of its own *i.e.*, at what time it started to execute. After reading these lines, compare the above results once again. Now you will clearly understand the difference between the functionality of the two functions *now( )* and *sysdate( ).*

LET US REVISE

- *A function is a special type of predefined command set that performs some operation and returns a single value.*
- *SQL functions can be categorized into – Single Row functions and Multiple Row functions.*
- ***Single row functions*** *work with single row at a time and* ***Multiple row functions*** *work with data of multiple rows at a time and return aggregated value.*
- *Major Character functions are – Lower/LCASE( ), Upper/UCASE( ), Concat( ), Instr( ), Length( ), RTrim( ), LTrim( ), Substr( ).*
- *Major Number functions are – Round( ), Truncate( ), Mod( ), Sign( ).*
- *Major Date functions are – Curdate( ), Date( ), Month( ), Monthname( ), Day( ), Year( ), DayName( ), DayofMonth( ), DayofWeek( ), DayofYear( ), Now( ), SysDate( ).*

Objective Type Questions — OTQs

Multiple Choice Questions

1. Which of the following is not a text function ?
   (a) TRIM( ) (b) TRUNCATE( )
   (c) LEFT( ) (d) MID( )
2. Which of the following is not a numeric function ?
   (a) MOD (b) SIGN (c) MID (d) POW
3. Which of the following is not a date function ?
   (a) Month (b) Year (c) NOW (d) POW
4. Which of the following functions returns the substring from a given string ?
   (a) MID (b) INSTR
   (c) SUBSTR (d) CHAR
5. Which of the following functions returns the position of a substring in a given string ?
   (a) MID (b) INSTR
   (c) SUBSTR (d) CHAR
6. What will be printed by the given query ?
   ```
   SELECT LENGTH("WINNER");
   ```
   (a) 7 (b) 6 (c) 8 (d) 9
7. What will be returned by the given query ?
   ```
   SELECT INSTR("INDIA", "DI");
   ```
   (a) 2 (b) 3 (c) –2 (d) –3
8. What will be returned by the given query ?
   ```
   SELECT concat("It", "was", "ok");
   ```
   (a) "It was ok" (b) "It wasok"
   (c) "Itwasok" (d) "Itwas ok"

9. What will be returned by the given query ?

```
SELECT Round(153.669, 2);
```

(a) 153.6 (b) 153.66 (c) 153.67 (d) 153.7

10. What will be returned by the given query ?

```
SELECT Sign(26);
```

(a) 1 (b) –1 (c) 0 (d) none of these

11. What will returned by the given query ?

```
SELECT Truncate(15.79, -1), Truncate(15.79, 0), Truncate(15.79, 1);
```

(a) 15 15 15.7 (b) 10 15.7 15.9
(c) 10 15 15.7 (d) 10 10 15.9

12. What will be returned by the given query ?

```
SELECT month('2020-05-11');
```

(a) 5 (b) 11 (c) May (d) November

Fill in the Blanks

1. The functions that work with one row at a time are called _____ _____ functions.
2. The functions that work with multiple rows and return aggregated result, are called _____ _____ functions.
3. To get a substring of a string, other than Substr( ), function _____ is also used.
4. To get the day part of a date, _____ function is used.
5. To get the day name from a date, _____ function is used.
6. To remove a character from the right side of a string, _____ function is used.
7. To get the current date, _____ _____ function is used.

True/False Questions

1. Truncate( ) is a text function.
2. Length( ) is a numeric function.
3. Functions MID( ) and SUBSTR( ) do the same thing.
4. Date( ) and Day( ) return the same thing.
5. Instr( ) and SUBSTR( ) work identically.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *Consider the sample menagerie database and answer the following queries.*

(i) `SELECT MONTH(birth), name FROM pet WHERE sex = 'm' ;`

(ii) `SELECT CONCAT ( CONCAT (species, '-'), sex ) , name FROM pet;`

(iii) `SELECT SUBSTR(name,1,3), species FROM pet WHERE sex = 'f' ;`

(iv) `SELECT YEAR( CURDATE() ) + YEAR(birth) FROM pet WHERE sex = 'f' ;`

Solution. *(i)*

```
+---------------+--------------+
| month(birth)  | name         |
+---------------+--------------+
| 3             | Claws        |
| 8             | Fang         |
| 8             | Bowser       |
| 4             | Slim         |
+---------------+--------------+
```

(ii)

```
+-------------------------------------+----------+
| concat ( concat (species, '-'), sex ) | name     |
+-------------------------------------+----------+
| cat-f                               | Fluffy   |
| cat-m                               | Claws    |
| dog-f                               | Buffy    |
| dog-m                               | Fang     |
| dog-m                               | Bowser   |
| bird-f                              | Chirpy   |
| NULL                                | Whistler |
| snake-m                             | Slim     |
| hamster-f                           | Puffball |
+-------------------------------------+----------+
```

(iii)

```
+--------------------+-----------+
| substr(name,1,3)   | species   |
+--------------------+-----------+
| Flu                | cat       |
| Buf                | dog       |
| Chi                | bird      |
| Puf                | hamster   |
+--------------------+-----------+
```

(iv)

```
+-----------------------------------+
| year( curdate() ) + year(birth)   |
+-----------------------------------+
| 4002                              |
| 3998                              |
| 4007                              |
| 4008                              |
+-----------------------------------+
```

2. *Predict the output of following queries*

(i) `SELECT ROUND(29.21), ROUND(32.76);`

(ii) `SELECT ROUND(29.21, 1), ROUND(32.76, 1);`

Ans. (i) 29 33 (ii) 29.2 32.8

3. *Predict the output of following queries*

   (*i*) `SELECT TRIM(TRAILING '.' FROM '  To be continued...');`

   (*ii*) `SELECT TRIM('  To be continued...');`

   Ans.

   ```
   To be continued
   To be continued...
   ```

   Contains leading spaces Please note there are leading spaces before the word 'To'

4. *Predict the output of following queries*

   (*i*) `SELECT 9 mod 2;`

   (*ii*) `SELECT CONCAT('Catch', 'a', 'falling', 'star');`

   Ans. (*i*) `1` (*ii*) `Catchafallingstar`

5. *What is a function ?*

   Solution. A **Function** is a special type of predefined command set that performs some operation and returns a single value.

6. *Write a query to display name, job, salary, and HireDate of employees who are hired between May 20, 1990, and December 31, 1991. Order the query in ascending order of HireDate. (Table Empl)*

   Solution.

```
mysql> SELECT ename, job, sal, hiredate FROM empl
    -> WHERE hiredate BETWEEN '1990-05-20' AND '1991-12-31'
    -> ORDER BY hiredate;
```

7. *Write a query to display the employee's name, salary and current date.*

   Solution.

```
mysql> SELECT ename, sal, CURDATE()
       FROM empl ;
```

8. *For each employee display employee name and total number of years lapsed between HireDate and Today.*

   Solution.

```
mysql> SELECT ename, YEAR(CURDATE() ) - YEAR(hiredate) FROM empl;
```

9. *Write a query which displays the employee name with all other letters in lower case and length of there name string.*

   Solution.

```
mysql> SELECT LOWER(ename), LENGTH(ename) FROM empl;
```

GLOSSARY

Date and Time functions	Functions to manipulate dates and time based expressions.
Function	Predefined command set that carries out an operation and returns a single value.
Numeric Functions	Functions to manipulate numbers.
String Functions	Functions to manipulate text.

Assignments

TYPE A : Short Answer Questions/Conceptual Questions

1. Define a function.
2. What will be the output of following code ?

```
mysql> SELECT CONCAT (CONCAT ('Inform', 'atics'), 'Practices');
```

3. What will be the output of following code ?

```
mysql> SELECT LCASE ('INFORMATICS PRACTICES CLASS 11TH') ;
```

4. What will be the output of following code ?

```
mysql> SELECT UCASE ('Computer studies') ;
```

5. What will be the output of following code ?

```
mysql> SELECT CONCAT (LOWER ('Class'), UPPER ('xii') ) ;
```

6. If Str = "INFORMATICS PRACTICES ...." and Str1 = "... FOR CLASS XI"
   Write commands to print the output as 'informatics practices for class xi'
7. Write commands to display the system date.
8. Write a command to display the name of current month.
9. Write SQL statement to display

```
Today, the date is <current date>
```

10. Write command to print the day of the week of your Birthday in the year 1999.
11. Write a command to display the current time.
12. Consider two fields *B_date*, which stores the birth date and *J_date*, which stores the joining date of an employee. Write commands to find out and display the approximate age of an employee as on today.

TYPE B : Short Answer Questions

1. Given the following table :

Table : CLUB

COACH_ID	COACHNAME	AGE	SPORTS	DATOFAPP	PAY	SEX
1.	KUKREJA	35	KARATE	1996-03-27	1000	M
2.	RAVINA	34	KARATE	1998-01-20	1200	F
3.	KARAN	34	SQUASH	1998-02-19	2000	M
4.	TARUN	33	BASKETBALL	1998-01-01	1500	M
5.	ZUBIN	36	SWIMMING	1998-01-12	750	M
6.	KETAKI	36	SWIMMING	1998-02-24	800	F
7.	ANKITA	39	SQUASH	1998-02-20	2200	F
8.	ZAREEN	37	KARATE	1998-02-22	1100	F
9.	KUSH	41	SWIMMING	1998-01-13	900	M
10.	SHAILYA	37	BASKETBALL	1998-02-19	1700	M

Give the output of following SQL statements :

(*i*) SELECT LCASE (SPORTS) FROM Club ;

(*ii*) SELECT MOD(Age, 5) FROM CLUB WHERE Sex = 'F' ;

(*iii*) SELECT POWER(3, 2) FROM CLUB WHERE Sports = 'KARATE' ;

(*iv*) SELECT SubStr(CoachName, 1, 2) FROM CLUB WHERE Datofapp > '1998-01-31' ;

2. Given the following table :

Table : STUDENT1

No.	Name	Stipend	Stream	AvgMark	Grade	Class
1	Karan	400.00	Medical	78.5	B	12B
2	Divakar	450.00	Commerce	89.2	A	11C
3	Divya	300.00	Commerce	68.6	C	12C
4	Arun	350.00	Humanities	73.1	B	12C
5	Sabina	500.00	Nonmedical	90.6	A	11A
6	John	400.00	Medical	75.4	B	12B
7	Robert	250.00	Humanities	64.4	C	11A
8	Rubina	450.00	Nonmedical	88.5	A	12A
9	Vikas	500.00	Nonmedical	92.0	A	12A
10	Mohan	300.00	Commerce	67.5	C	12C

Give the output of followint SQL statement :

(*i*) SELECT TRUNCATE(AvgMark) FROM Student1 WHERE AvgMark < 75 ;

(*ii*) SELECT ROUND(AvgMark) FROM Student1 WHERE Grade = *'B'* ;

(*iii*) SELECT CONCAT(Name, Stream) FROM Student1 WHERE Class = '12A' ;

(*iv*) SELECT RIGHT(Stream, 2) FROM Student1 ;

3. Given the following table :

TABLE : LIBRARY

No	Title	Author	Type	Pub	Qty	Price
1.	Data Structure	Lipschutz	DS	McGraw	4	217
2.	Computer Studies	French	FND	Galgotia	2	75
3.	Advanced Pascal	Schildt	PROG	McGraw	4	350
4.	Dbase dummies	Palmer	DBMS	PustakM	5	130
5.	Mastering C++	Gurewich	PROG	BPB	3	295
6.	Guide Network	Freed	NET	ZPress	3	200
7.	Mastering Foxpro	Seigal	DBMS	BPB	2	135
8.	DOS guide	Norton	OS	PHI	3	175
9.	Basic for beginners	Morton	PROG	BPB	3	40
10.	Mastering Window	Cowart	OS	BPB	1	225

Give the output of following SQL commands on the basis of table Library.

(*i*) SELECT UPPER(Title) FROM Library WHERE Price < 150 ;

(*ii*) SELECT CONCAT(Author, Type) FROM Library WHERE Qty < 3 ;

(*iii*) SELECT MOD(Qty, 4) FROM Library ;

Perform following questions based on these tables :

```
table PAYDAY (contains one column only)

    CycleDate    DATE

table ADDRESS ( contains following eight columns)

    LastName        VARCHAR(25),
    FirstName       VARCHAR(25),
    Street          VARCHAR(50),
    City            VARCHAR(25),
    State           CHAR(2),
    Zip             NUMBER,
    Phone           VARCHAR(12),
    Ext             VARCHAR(5)
```

4. Write a query to show the current date and time.
5. Write a query to show the city of user with first name as 'MARK'.
6. Show via query how many days remain until Christmas. Round fractional days up using the numeric function ROUND.
7. Write a query against table PAYDAY that will show the number of days between the payday (CYCLEDATE column) and the last day of the month (LAST_DAY function).

   Format your results as below :

```
Month          Days between payday and first days of next month
--------       -----------
January        16
February       13
March          16
```

8. Write a query against the EMPL table to show the names of all employees concatenated with their jobtypes.
9. Write a query against the ADDRESS table to show the names (first name, last name) and phones of all persons concatenated in following form :

```
TinaSeth23456789
MoradK.22211890
```

10. Write a query against the ADDRESS table to select a list of names and phone numbers. The output should match these requirements :
    - The name column should contain both the first and last names with a blank space between them. Use the string concatenation.
    - The second column will contain the phone number.
    - Phone number should be in the format (999) 999-9999. Use the SUBSTR function and CONCAT.
    - Order the query by last name then first name.

7 Querying Using SQL

Outline

7.1 Introduction
7.2 Ordering Records in Result–Order by Clause
7.3 Aggregate Functions
7.4 Types of SQL Functions
7.5 Grouping Result – GROUP BY

7.1 Introduction

You have worked with basic SQL commands in your previous class and used database MySQL. This chapter will talk about some SQL commands in details. After this chapter, you shall be able to display groups of records, summary of groups of records and display selected groups of records.

Let us begin with our discussion on how you can order records of a table using ORDER BY clause.

7.2 Ordering Records in Result – Order by Clause

Although you have read about ORDER BY clause of SQL SELECT statement in your previous class, here we are covering it again in a detailed manner.

7.2.1 Recalling SQL SELECT ORDER BY Clause

The result set generated by the SQL SELECT statement is not ordered in any form by default. However, if you want to sort or order the result set, you can use the ORDER BY clause of SQL SELECT statement as per following format :

```
SELECT <comma separated select list> FROM <table>
[WHERE <condition>]
ORDER BY <fieldname> [ASC|DESC] [,<fieldname> [ASC|DESC], ... ];
```

Keywords ASC and DESC denote the order – **ASC** stands for ***ascending*** and the DESC stands for ***descending***. If you do not specify any order keyword ASC or DESC, then by default, the ORDER BY clause sorts the result set in ascending order.

For example, consider the table **Data** having records as shown below :

```
+---------+----------+----------+-------+-----------+
| rollno  | name     | marks    | grade | section   |
+---------+----------+----------+-------+-----------+
| 101     | Ruhani   | 76.80    | A     | A         |
| 102     | George   | 71.20    | B     | A         |
| 103     | Simran   | 81.20    | A     | B         |
| 104     | Ali      | 61.20    | B     | C         |
| 105     | Kushal   | 51.60    | C     | C         |
| 106     | Arsiya   | 91.60    | A+    | B         |
| 107     | Raunaq   | 32.50    | F     | B         |
| 108     | Meera    | 97.20    | A+    | B         |
| 109     | Amaal    | 57.20    | C     | B         |
| 111     | Simran   | 66.00    | B     | A         |
| 112     | Adam     | 74.20    | B     | C         |
| 113     | Gurnoor  | 93.50    | A+    | B         |
| 115     | Rabiya   | 72.50    | B     | B         |
| 117     | Rahil    | 32.00    | F     | C         |
| 118     | Neha     | 59.50    | C     | A         |
+---------+----------+----------+-------+-----------+
```

Now the statement

```
mysql> SELECT * FROM data
    -> ORDER BY marks ;
```

will produce result set with marks arranged in ascending order as we did not specify ASC keyword explicitly, *i.e.*,

```
+---------+----------+----------+-------+-----------+
| rollno  | name     | marks    | grade | section   |
+---------+----------+----------+-------+-----------+
| 117     | Rahil    | 32.00    | F     | C         |
| 107     | Raunaq   | 32.50    | F     | B         |
| 105     | Kushal   | 51.60    | C     | C         |
| 109     | Amaal    | 57.20    | C     | B         |
| 118     | Neha     | 59.50    | C     | A         |
| 104     | Ali      | 61.20    | B     | C         |
| 111     | Simran   | 66.00    | B     | A         |
| 102     | George   | 71.20    | B     | A         |
| 115     | Rabiya   | 72.50    | B     | B         |
| 112     | Adam     | 74.20    | B     | C         |
| 101     | Ruhani   | 76.80    | A     | A         |
| 103     | Simran   | 81.20    | A     | B         |
| 106     | Arsiya   | 91.60    | A+    | B         |
| 113     | Gurnoor  | 93.50    | A+    | B         |
| 108     | Meera    | 97.20    | A+    | B         |
+---------+----------+----------+-------+-----------+
15 ROWS IN SET (0.01 SEC)
```


The above SQL statement is equivalent to statement shown below where ASC word is explicitly specified.

```
mysql> SELECT * FROM data
    -> ORDER BY marks ASC ;
```

This statement will also produce the same result as above because ASC order is taken by default.

7.2.2 Ordering Data on Multiple Columns

To order the result set on multiple columns, you can specify the multiple column names in ORDER by clause along with the desired sort order, *i.e.*, as :

```
SELECT
:
ORDER BY <fieldname1> [ASC|DESC] [,<fieldname1> [ASC|DESC], … ] ;
```

For example, the following statement will sort the records firstly on the column name **Section** and then on the basis of descending order of column **marks**.

```
mysql> SELECT * FROM data
    -> ORDER BY section ASC , marks DESC ;
```

*First sort field is **section** in ascending order (as **section ASC**) and for all the records of same section, the sort field is **marks** with descending order (**marks DESC**)*

```
+---------+----------+----------+-------+-----------+
| rollno  | name     | marks    | grade | section   |
+---------+----------+----------+-------+-----------+
| 101     | Ruhani   | 76.80    | A     | A         |
| 102     | George   | 71.20    | B     | A         |
| 111     | Simran   | 66.00    | B     | A         |
| 118     | Neha     | 59.50    | C     | A         |
| 108     | Meera    | 97.20    | A+    | B         |
| 113     | Gurnoor  | 93.50    | A+    | B         |
| 106     | Arsiya   | 91.60    | A+    | B         |
| 103     | Simran   | 81.20    | A     | B         |
| 115     | Rabiya   | 72.50    | B     | B         |
| 109     | Amaal    | 57.20    | C     | B         |
| 107     | Raunaq   | 32.50    | F     | B         |
| 112     | Adam     | 74.20    | B     | C         |
| 104     | Ali      | 61.20    | B     | C         |
| 105     | Kushal   | 51.60    | C     | C         |
| 117     | Rahil    | 32.00    | F     | C         |
+---------+----------+----------+-------+-----------+
15 rows in set (0.00 sec)
```

See all records of same section are listed together

*All records with same section are arranged in descending order of **marks** column*

7.2.3 Ordering Data on the Basis of an Expression

Sometimes, you need to display the result of a calculation or a mathematical expression in the result set. In such cases, you may want or need to arrange your result set in the order of the calculated expression. The ORDER BY clause allows you to include the mathematical

expression to order the result set by it. However, to arrange a result set on the basis of a mathematical expression, you should preferably (though not a necessity but preferably) include the mathematical expression in the select list so that it becomes easy to comprehend the result.

Consider the following example statement that arranges the result set on the basis of a calculated result :

```
mysql> SELECT rollno, name, grade, section, marks*.35 FROM data
    -> WHERE marks > 70
    -> ORDER BY section ASC, marks*0.35 DESC ;
```

rollno	name	grade	section	marks*.35
101	Ruhani	A	A	26.8800
102	George	B	A	24.9200
108	Meera	A+	B	34.0200
113	Gurnoor	A+	B	32.7250
106	Arsiya	A+	B	32.0600
103	Simran	A	B	28.4200
115	Rabiya	B	B	25.3750
112	Adam	B	C	25.9700

8 rows in set (0.02 sec)

If you want, you can provide a column alias name to the mathematical expression in the select list, *e.g.*, following statement will also produce the same result as above but it will name the column **marks*.35** as **Term1**:

```
mysql> SELECT rollno, name, grade, section, marks*.35 as Term1 FROM data
    -> WHERE marks > 70
    -> ORDER BY section ASC, Term1 DESC ;
```

rollno	name	grade	section	Term1
101	Ruhani	A	A	26.8800
102	George	B	A	24.9200
108	Meera	A+	B	34.0200
113	Gurnoor	A+	B	32.7250
106	Arsiya	A+	B	32.0600
103	Simran	A	B	28.4200
115	Rabiya	B	B	25.3750
112	Adam	B	C	25.9700

8 rows in set (0.00 sec)

7.3 Aggregate Functions

Till now you have learnt to work with functions that operate on individual rows in a table *e.g.*, if you use **Round( )** function then it will round off values from each row of the table.

MySQL also supports and provides **group functions** or **aggregate functions**. As you can make out that the group functions or aggregate functions work upon groups of rows, rather than on single rows. That is why, these functions are sometimes also called **multiple row functions**.

Many group functions accept the following options :

DISTINCT This option causes a group function to consider only distinct values of the argument expression.

ALL This option causes a group function to consider all values including all duplicates.

The usage of these options will become clear with the coverage of examples in this section.

All the examples that we'll be using here, shall be based upon following table **empl**.

Table 7.1 *Database table empl*

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
8369	SMITH	CLERK	8902	1990-12-18	800.00	NULL	20
8499	ANYA	SALESMAN	8698	1991-02-20	1600.00	300.00	30
8521	SETH	SALESMAN	8698	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	20
8654	MOMIN	SALESMAN	8698	1991-09-28	1250.00	1400.00	30
8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	30
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	10
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500.00	0.00	30
8882	SHIAVNSH	MANAGER	8839	1991-06-09	2450.00	NULL	10
8886	ANOOP	CLERK	8888	1993-01-12	1100.00	NULL	20
8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	20
8900	JATIN	CLERK	8698	1991-12-03	950.00	NULL	30
8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	20
8934	MITA	CLERK	8882	1992-01-23	1300.00	NULL	10

1. AVG

This function computes the average of given data.

Syntax

```
AVG([DISTINCT | ALL] n)
```

▲ Returns average value of parameter(s) *n*.

Argument type : Numeric *Return value* : Numeric

EXAMPLE 1 *Calculate average salary of all employees listed in table empl.*

SOLUTION

```
mysql> SELECT AVG(sal) "Average"
    -> FROM empl ;
```

Average
2073.928571

1 row in set (0.01 sec)

2. COUNT

This function counts the number of rows in a given column or expression.

Syntax

```
COUNT({ * [DISTINCT | ALL] expr})
```

- Returns the number of rows in the query.
- If you specify argument *expr*, this function returns rows where *expr* is not null. You can count either all rows, or only distinct values of *expr*.
- If you specify the asterisk (*), this function returns all rows, including duplicates and nulls.

Argument type : Numeric *Return value* : Numeric

EXAMPLE 2 *Count number of records in table empl.*

SOLUTION

```
mysql> SELECT COUNT(*) "Total"
       FROM empl ;
```

```
+----------+
| Total    |
+----------+
|   14     |
+----------+
1 row in set (0.00 sec)
```

EXAMPLE 3 3 *Count number of jobs in table empl.*

SOLUTION

```
mysql> SELECT COUNT(job) "Job Count"
       FROM empl ;
```

```
+-------------+
| Job Count   |
+-------------+
|     14      |
+-------------+
1 row in set (0.01 sec)
```

EXAMPLE 4 *How many distinct jobs are listed in table empl ?*

SOLUTION

```
mysql> SELECT COUNT(DISTINCT job) "Distinct Jobs"
       FROM empl ;
```

```
+----------------+
| Distinct Jobs  |
+----------------+
|       15       |
+----------------+
1 row in set (0.04 sec)
```

3. MAX

This function returns the maximum value from a given column or expression.

Syntax

```
MAX([DISTINCT|ALL] expr)
```

- Returns maximum value of argument *expr*.

Argument type : Numeric *Return value* : Numeric

EXAMPLE 5 *Display maximum salary from table empl.*

SOLUTION

```
mysql> SELECT MAX(sal) "Maximum Salary"
       FROM empl ;
```

```
+----------------+
| Maximum Salary |
+----------------+
|    5000.00     |
+----------------+
1 row in set (0.01 sec)
```


4. MIN

This function returns the minimum value from a given column or expression.

Syntax

```
MIN([DISTINCT | ALL] expr)
```

- Returns minimum value of *expr*.

Argument type : Numeric *Return value* : Numeric

EXAMPLE 6 *Display the Joining date of seniormost employee.*

SOLUTION

```
mysql> SELECT MIN(hiredate) "Minimum Hire Date"
       FROM empl ;
```

```
+--------------------+
| Minimum Hire Date  |
+--------------------+
|     1990-12-18     |
+--------------------+
1 row in set (0.06 sec)
```

5. SUM

This function returns the sum of values in given column or expression.

Syntax

```
SUM([DISTINCT | ALL] n)
```

- Returns sum of values of *n*.

Argument type : Numeric *Return value* : Numeric

EXAMPLE 7 *Display total salary of all employees listed in table empl.*

SOLUTION

```
mysql> SELECT SUM(sal) "Total Salary"
       FROM empl ;
```

```
+-----------------+
| Total Salary    |
+-----------------+
|    29035.00     |
+-----------------+
1 row in set (0.01 sec)
```

NOTE

These functions are called *aggregate functions* because they operate on aggregates of tuples. The result of an aggregate function is a single value.

Some more examples of group functions are being given below :

Examples :

1. To calculate the total gross for employees of grade 'E2', the command is :

```
SELECT SUM(gross) FROM employee
WHERE grade = 'E2' ;
```

2. To display the average gross of employees with grades 'E1' or 'E2', the command used is :

```
SELECT AVG(gross) FROM employee
WHERE (grade = 'E1' OR grade = 'E2') ;
```

3. To count the number of employees in *employee* table, the *SQL* command is :

```
SELECT COUNT(*)
FROM employee ;
```

4. To count the number of cities, the different members belong to, you use the following command :

```
SELECT COUNT(DISTINCT city)   FROM members ;
```

Here the DISTINCT keyword ensures that multiple entries of the same *city* are ignored. The * is the only argument that includes NULLs when it is used only with COUNT, functions other than COUNT disregard NULLs in any case.

If you want to count the entries including repeats, the keyword ALL is used. The following command will COUNT the number of non NULL *city* fields in the *members* table :

```
SELECT COUNT(ALL city)
FROM members ;
```

In general, GROUP functions

- Return a single value for a set of rows.
- Can be applied to any numeric values, and some Text types and DATE values.

7.4 Types of SQL Functions

Now that you have learnt different types of functions, let us talk about their broad categories. SQL supports many and many functions. All these functions can be generally categorized into following *two* types :

- Single Row (or Scalar) functions.
- Multiple Row (or Group or Aggregate) functions.

(*i*) ***Single Row functions*** work with a single row at a time. A single row function returns a result for every row of a queried table.

(*ii*) ***Multiple Row or Group functions*** work with data of multiple rows at a time and return aggregated value.

Examples of multiple row functions are the group functions that you have learnt in previous section *i.e.*, sum( ), count( ), max( ), min( ), Avg( ) etc.

The difference between these two types of functions is in the number of rows they act upon. A **single row function** works with the data of a single row at a time and returns a single result for each row queried upon ; a **multiple row function** works with a group of rows and returns a single result for that group.

7.5 Grouping Result – GROUP BY

The GROUP BY clause combines all those records that have identical values in a particular field or a group of fields. This grouping results into one summary record per group if group-functions are used with it. In other words, the GROUP BY clause is used in SELECT statements to divide the table into groups. Grouping can be done by a column name, or with aggregate functions in which case the aggregate produces a value for each group.

For example, to calculate the *number of employees in each grade*, you use the command

```
SELECT job, COUNT(*)
FROM empl
GROUP BY job ;
```

```
+-----------+----------+
| job       | COUNT(*) |
+-----------+----------+
| ANALYST   |        2 |
| CLERK     |        4 |
| MANAGER   |        3 |
| PRESIDENT |        1 |
| SALESMAN  |        4 |
+-----------+----------+
```

GROUP BY applies the aggregate functions independently to a series of groups that are defined by having a field value in common.

Now consider the following query, which is also grouping records based on *deptno*.

```
mysql> SELECT deptno, COUNT(*), SUM(sal)
       FROM empl
       GROUP BY deptno ;
```

```
+------------+----------+------------+
| deptno     | COUNT(*) | SUM(sal)   |
+------------+----------+------------+
|     10     |        3 |  8750.00   |
|     20     |        5 | 10885.00   |
|     30     |        6 |  9400.00   |
+------------+----------+------------+
```

As you can make out that the above query is displaying count of records and sum of salaries in each group and the groups are formed on the basis of *deptno*. Thus, from the above output, you can make out that in department number 10, there are 3 employees (records) and total of all salaries is 8750.00 ; in department number 20, there are 5 employees and total of salaries is 10885.00 ; and so on.

7.5.1 Nested Groups – Grouping on Multiple Columns

With GROUP BY clause, you can create groups within groups. Such type of grouping is called ***Nested grouping***. This can be done by specifying in GROUP BY expression, where the *first field* determines the *highest group level*, the *second field* determines the *second group level*, and so on. The *last field* determines the *lowest level of grouping*.

In order to fully understand this concept, consider Table **7.1 empl**.

See there are multiple records having same value for field **Deptno**, we can group records on the basis of field *Deptno*. For instance, if you want to count the number of employees in each group, you need to issue a query statement as given below :

```
SELECT COUNT(empno) FROM empl
GROUP BY Deptno ;
```

And the result produced by this query is

```
+--------------+
|count(empno)  |
+--------------+
|      3       |
|      5       |
|      6       |
+--------------+
```

But can you make out, these are employee-counts for which departments ? To get this information, you may modify the SELECT list as :

```
SELECT Deptno, COUNT(empno)
FROM empl
GROUP BY Deptno ;
```

Now the result will be :

```
+---------+--------------+
| deptno  | count(empno) |
+---------+--------------+
|   10    |      3       |
|   20    |      5       |
|   30    |      6       |
+---------+--------------+
```

See, now it is more clear. But one thing that you should keep in mind is that while grouping, you should include only those values in the *select list* that either have the same value for a group or contain a group (aggregate) function *i.e.*, a **group-expression**. Like in the above query, the first expression **Deptno** field has one (same) value for a group and the other expression **COUNT(empno)** contains a group function.

MySQL as such would not create any error even if you include a **non-group expression** in the select-list. A **non-group field** (or expression) is the field that has different values in the rows belonging to the group.

In this case, it will return **the value from first record** of the group for that non-group field *e.g.*, if you issue command like :

```
SELECT deptno, count(empno), mgr
FROM empl
GROUP BY deptno ;
```

This is non-group field as it has multiple values for a group.

The output returned will be :

```
+---------+--------------+------+
| deptno  | count(empno) | mgr  |
+---------+--------------+------+
|   10    |      3       | NULL |
|   20    |      5       | 8902 |
|   30    |      6       | 8698 |
+---------+--------------+------+
```

See, first record of group with deptno 10 has value NULL in mgr field ; first record of group with deptno 20 has 8902 in mgr field ; and first record of group with deptno 30 has value 8698 in mgr field. Hence this output.

To create a group within a group *i.e.*, nested group, you need to specify multiple fields in the GROUP BY expression. If you have a look at the records of *Empl* table, you can make out that there exists a *group of jobs* within the *department group* as there are same values for **Job** field, in one department group's records.

NOTE

In the select list of a group, only those fields or expressions state be included that either return single value for a group or are constants. Otherwise you may not get authentic results.

To group records *job wise* within *Deptno* wise, you need to issue a query statement like :

```
SELECT Deptno, Job, COUNT(empno)
FROM empl
GROUP BY Deptno, Job ;
```


And the result produced is :

```
+---------+-----------+--------------+
| deptno  |    job    | count(empno) |
+---------+-----------+--------------+
|   10    | CLERK     |      1       |
|   10    | MANAGER   |      1       |
|   10    | PRESIDENT |      1       |
|   20    | ANALYST   |      2       |
|   20    | CLERK     |      2       |
|   20    | MANAGER   |      1       |
|   30    | CLERK     |      1       |
|   30    | MANAGER   |      1       |
|   30    | SALESMAN  |      4       |
+---------+-----------+--------------+
```

7.5.2 Placing Conditions on Groups – HAVING Clause

The HAVING clause places conditions on groups in contrast to WHERE clause that places conditions on individual rows. While WHERE conditions cannot include aggregate functions, HAVING conditions can do so.

For example, to calculate the average gross and total gross for employees belonging to 'E4' grade, the command would be :

```
SELECT AVG(gross), SUM(gross)
FROM employee
GROUP BY grade
HAVING grade = 'E4' ;
```

This condition would be applicable on group and not on individual rows

To display the jobs where the number of employees is less than 3, you use the command :

```
SELECT JOB, COUNT(*)
FROM empl
GROUP BY job
HAVING count(*) < 3 ;
```

This will produce the following output :

```
+-----------+----------+
|    JOB    | COUNT(*) |
+-----------+----------+
| ANALYST   |     2    |
| PRESIDENT |     1    |
+-----------+----------+
2 rows in set (0.11 sec)
```

The HAVING clause can contain either a simple boolean expression (*i.e.*, an expression or condition that results into *true* or *false*) or use aggregate function in the having condition.

You can include more than one condition in HAVING clause, of course, by using logical operators.

Consider this :

```
SELECT Deptno, AVG(Comm), AVG(Sal)
FROM empl
GROUP BY Deptno
HAVING AVG(Comm) > 750 AND
AVG(Sal) > 2000 ;
```

Check Point 7.1

1. Can you arrange the result set of an SQL query on multiple columns ?
2. What is the significance of "ORDER BY" in the given query?

```
SELECT emp_id, fname, lname
FROM person
ORDER BY emp_id;
```

(a) Data of table **person** on the basis of column **emp_id** will be sorted in descending order
(b) Data of table **person** on the basis of column **emp_id** will be sorted in ascending order
(c) Only data of column emp_id will be sorted in descending order
(d) Only data of column emp_id will be sorted in ascending order

3. What will be the order of sorting in the given query?

```
SELECT emp_id, emp_name
FROM person
ORDER BY emp_id, emp_name;
```

(a) Firstly on ***emp_id*** and then on ***emp_name***
(b) Firstly on ***emp_name*** and then on ***emp_id***
(c) Firstly on ***emp_id*** but not on ***emp_name***
(d) None of the mentioned

4. If column ***emp_id*** contains the following set {9, 7, 6, 4, 3, 1, 2}, what will be the output on execution of the given query?

```
SELECT emp_id FROM person
ORDER BY emp_id;
```

(a) {9, 7, 6, 4, 3, 1, 2}
(b) {1, 2, , 3, 4, 6, 7, 9}
(c) {2, 1, 4, 3, 7, 9, 6}
(d) None of these

5. Which function can you use with ORDER BY clause to specify custom sort order ?
(a) SORT( )
(b) CUSTOM( )
(c) FIELD( )
(d) All of these

Also, you can use an aggregate function in the HAVING clause even if it is not in the SELECT list. Consider the following query to understand this :

```
SELECT Deptno, AVG(Sal)
FROM empl
GROUP BY Deptno
HAVING COUNT(*) <= 3 ;
```

You can also use IN or BETWEEN operators with HAVING clause.

Following two queries illustrate this :

```
SELECT Deptno, Job, AVG(Sal)
FROM empl
GROUP BY Deptno, Job
HAVING JOB IN ('CLERK', 'SALESMAN') ;
SELECT Deptno, Job, SUM(sal)
FROM empl
GROUP BY Deptno, Job
HAVING SUM(sal) BETWEEN 3000 AND 7000 ;
```

7.5.3 Non-Group Expressions with GROUP BY

As mentioned before, if you include a non-group expression in the select-list of a query with GROUP BY, MySQL will not produce any error. Rather it will pick value of the specified non-group field from the first row of the group. But we do not recommend this practice because it will produce ambiguous results. For instance, consider the following query and its output.

```
mysql> SELECT ename, sum(sal)
       FROM empl
       GROUP BY deptno ;
```

This will produce the following output :

```
+--------+----------+
| ename  | sum(sal) |
+--------+----------+
| AMIR   |  8750.00 |
| SMITH  | 10885.00 |
| ANYA   |  9400.00 |
+--------+----------+
```

See, isn't it conveying that AMIR's salary-sum is 8750.00, SMITH'S 10885.00 and ANYA'S 9400.00 ?

Thus, we recommend not to use non-group expressions in GROUP BY query unless otherwise necessary.

LET US REVISE

- *The ORDER BY clause lets you arrange the result set in the order of single column, multiple columns, on the basis of an expression etc. .*
- *The GROUP BY clause combines all those records that have identical value in a particular field or a group of fields.*
- *GROUP BY clause is used to divide the result in groups.*
- *A group within another group is called* **Nested Group.**
- *Nested grouping can be done by providing multiple fields in the GROUP BY expression.*
- *All fields containing a NULL value are considered to have a value and are grouped with the fields containing non-NULL values.*
- *The SELECT list of a group can include expressions returning single value per group or constants.*
- *The HAVING clause is used to specify filtering condition for groups.*
- *The difference between WHERE and HAVING clause is that WHERE conditions are applicable on individual rows whereas HAVING conditions are applicable on groups as formed by GROUP BY clause.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. By default, ORDER BY clause lists the results in _____ order.
(a) Descending (b) Any (c) Same (d) Ascending

2. Consider the following query

```
SELECT * FROM employee ORDER BY salary _____ , name _____ ;
```

To display the *salary* from greater to smaller and *name* in alphabetical order which of the following options should be used ?
(a) Ascending, Descending (b) Asc, Desc
(c) Desc, Asc (d) Descending, Ascending

3. Select correct SQL query from below to find the temperature in increasing order of all cities.
(a) SELECT city FROM weather ORDER BY temperature ;
(b) SELECT city, temperature FROM weather ;
(c) SELECT city, temperature FROM weather ORDER BY temperature ;
(d) SELECT city, temperature FROM weather ORDER BY city ;

4. Which SQL function is used to count the number of rows in a SQL query ?
(a) COUNT( ) (b) NUMBER( ) (c) SUM( ) (d) COUNT(*)

5. With SQL, how can you return the number of not null records in the **Project** field of "Students" table ?
(a) SELECT COUNT(Project) FROM Students
(b) SELECT COLUMNS(Project) FROM Students
(c) SELECT COLUMNS(*) FROM Students
(d) SELECT COUNT(*) FROM Students

6. Which of the following is not an aggregate function ?
   (*a*) Avg (*b*) Sum (*c*) With (*d*) Min
7. All aggregate functions except _____ ignore null values in their input collection.
   (*a*) Count(attribute) (*b*) Count(*) (*c*) Avg (*d*) Sum
8. Which of the following group functions ignore NULL values ?
   (*a*) MAX (*b*) COUNT (*c*) SUM (*d*) All of these
9. The HAVING clause does which of the following ?
   (*a*) Acts EXACTLY like a WHERE clause.
   (*b*) Acts like a WHERE clause but is used for columns rather than groups.
   (*c*) Acts like a WHERE clause but is used for groups rather than rows.
   (*d*) Acts like a WHERE clause but is used for rows rather than columns.
10. Aggregate functions can be used in the select list or the_____clause of a select statement. They cannot be used in a _____ clause.
    (*a*) Where, having (*b*) Having, where
    (*c*) Group by, having (*c*) Group by, where
11. SQL applies conditions on the groups through _____ clause after groups have been formed,
    (*a*) Group by (*b*) With (*c*) Where (*d*) Having
12. Which clause is used with "aggregate functions" ?
    (*a*) GROUP BY (*b*) SELECT (*c*) WHERE (*d*) Both (*a*) and (*c*)
13. What is the meaning of "HAVING" clause in SELECT query ?
    (*a*) To filter out the summary groups (*b*) To filter out the column groups
    (*c*) To filter out the row and column values
    (*d*) None of the mentioned
14. Where and Having clauses can be used interchangeably in SELECT queries ?
    (*a*) True (*b*) False (*c*) Only in views (*c*) With order by

Fill in the Blanks

1. To sort the rows of the result table, the _____ _____ clause is specified.
2. Columns can be sorted in descending sequence by using the SQL keyword _____ .
3. The SQL built-in function _____ totals values in numeric columns.
4. The SQL built-in function _____ computes the average of values in numeric columns.
5. The SQL built-in function _____ obtains the largest value in a numeric column.
6. The SQL built-in function _____ obtains the smallest value in a numeric column.
7. The SQL built-in function _____ computes the number of rows in a table.
8. The SELECT clause _____ is used to collect those rows that have the same value in a specified column.

True/False Questions

1. The rows of the result relation produced by a SELECT statement can be sorted, but only by one column.
2. The HAVING clause acts like a WHERE clause, but it identifies groups that meet a criterion, rather than rows.

3. The qualifier DISTINCT must be used in an SQL statement when we want to eliminate duplicate rows.
4. DISTINCT and its counterpart, ALL, can be used more than once in a SELECT statement.
5. DISTINCT and its counterpart, ALL, can be used together on single field in a SELECT statement.
6. COUNT(field_name) tallies only those rows that contain a value; it ignores all null values.
7. SUM, AVG, MIN, and MAX can only be used with numeric columns.
8. The SQL statement : *SELECT Salary + Comm AS Total FROM Emp;* adds two fields salary and comm from each row together and lists the results in a column named Total.
9. ORDER BY can be combined with the SELECT statements.
10. The SQL keyword GROUP BY instructs the DBMS to group together those rows that have the same value in a column.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *The SQL SELECT provides clauses for sorting data and for summarizing results. Write the names of clauses for these.*

   Solution. The ORDER BY clause of SQL SELECT statement allows to sort the data of result set.

   The GROUP BY clause of SQL SELECT statement allows to create summarized results of grouped data from table.

2. *You want to group the result set based on some column's value. Also, you want that the grouped result should appear in a sorted order. In which order will you write the two clauses (for sorting and for grouping). Give example to support your answer.*

   Solution. When we use GROUP BY clause (for grouping of data) and ORDER BY clause (for sorting data) together, the ORDER BY clause always follows other clauses. That is, the GROUP BY clause will come before ORDER BY clause.

   For example,

```
SELECT userid, SUM(score) AS total_score
FROM user_score
GROUP BY userid
ORDER BY userid ASC;
```

3. *In a table* **Apply**, *there is a column namely* **Experience** *that can store only one of these values.: 'Fresher', 'Private-sector-experience', 'Public-sector-experience', 'Govt.-sector experience'.*

   You want to sort the data of table based on column **experience** *as per this order : 'Govt-sector-experience', 'Public-sector-experience', 'Private-sector-experience', 'Fresher'.*

   Write an SQL query to achieve this.

   Solution.

```
SELECT * FROM Apply
ORDER BY FIELD (Experience, 'Govt.-sector-experience', 'Public-sector-experience',
                'Private-sector-experience', 'Fresher') ;
```

4. *What are different types of SQL functions ?*

Solution. (*i*) Single Row (or Scalar) functions.

(*ii*) Multiple Row (or Group or Aggregate) functions.

(*i*) ***Single Row Functions*** work with a single row at a time. A single row function returns a result for every row of a queried table.

(*ii*) ***Multiple Row or Group Functions*** work with data of multiple rows at a time and return aggregated value.

5. *What is the significance of GROUP BY clause in a SQL query ?*

Solution. The GROUP BY clause combines all those records that have identical values in a particular field or a group of fields. This grouping results into one summary record per group if group-functions are used with it.

6. *What is the difference between a WHERE clause and a HAVING clause of SQL SELECT statement ?*

Solution. The difference between WHERE and HAVING clause is that WHERE conditions are applicable on individual rows whereas HAVING conditions are applicable on groups as formed by GROUP BY clause.

7. *Write a query to display the Sum, Average, Highest and Lowest salary of the employees.*

Solution.
```
mysql> SELECT SUM (sal), AVG (sal), MAX (sal), MIN (sal)
       FROM empl ;
```

8. *Write a query to display the Sum, Average, Highest and Lowest salary of the employees grouped by department number.*

Solution.
```
mysql> SELECT SUM (sal), AVG (sal), MAX (sal), MIN (sal)
       FROM empl GROUP BY deptno ;
```

9. *Write a query to display the Sum, Average, Highest and Lowest salary of the employees grouped by department number and sub-grouped by job.*

Solution.
```
mysql> SELECT SUM (sal), AVG (sal), MAX (sal), MIN (sal)
       FROM empl
       GROUP BY deptno, job ;
```

10. *Write a query to display the number of employees with same job.*

Solution.
```
mysql> SELECT COUNT(*) "No_of_Emps", job
       FROM emp
       GROUP BY job ;
```

11. *Write a query to display the difference of highest and lowest salary of each department having maximum salary > 4000.*

Solution.
```
mysql> SELECT MAX (sal) – MIN (sal) "Difference" FROM empl
       GROUP BY deptno
       HAVING MAX (sal) > 4000 ;
```

12. *Gopi Krishna is using a table Employee. It has the following columns :* [CBSE D 2014]

```
Code, Name, Salary, Deptcode
```

He wants to display maximum salary departmentwise. He wrote the following command :

```
SELECT Deptcode, Max(Salary) FROM Employee ;
```

But he did not get the desired result.

Rewrite the above query with necessary changes to help him get the desired output.

Solution.
```
SELECT Deptcode, Max(Salary)
FROM Employee
GROUP BY Deptcode ;
```

13. *Shanya Khanna is using a table Employee. It has the following columns :* [CBSE OD 2014]

```
Admno, Name, Agg, Stream
```

[column Agg contains Aggregate marks]

She wants to display highest Agg obtained in each Stream.

She wrote the following statement :

```
SELECT Stream, MAX(Agg) FROM Employee ;
```

But she did not get the desired result. Rewrite the above query with necessary changes to help her get the desired output.

Solution.
```
SELECT Stream, MAX(Agg)
FROM Employee
GROUP BY Stream ;
```

14. *Write a query that counts the number of salespeople registering orders for each day. (If a salesperson has more than one order on a given day, he or she should be counted only once.).*

Solution.
```
SELECT ord_date, count (DISTINCT salesman_code) FROM orders
GROUP BY ord_date ;
```

15. *Write a query on the customers table that will find the highest rating in each city. Put the output in this form : For the city (city), the highest rating is : (rating).*

Solution.
```
SELECT 'For the city', city, 'the highest rating is :', MAX (rating)
FROM customers
GROUP BY city ;
```

16. *Consider the following tables ACTIVITY and COACH. Write SQL commands for the following statements.* [CBSE Sample Paper I. 12]

Table : **ACTIVITY**

Acode	ActivityName	Stadium	ParticipantsNum	PrizeMoney	ScheduleDate
1001	Relay 100 × 4	Star Annex	16	10000	23-Jan-04
1002	High jump	Star Annex	10	12000	12-Dec-03
1003	Shot Put	Super Power	12	8000	14-Feb-04
1005	Long Jump	Star Annex	12	9000	01-Jan-04
1008	Discuss Throw	Super Power	10	15000	19-Mar-04

Table : **COACH**

Pcode	Name	Acode
1	Ahmad Hussain	1001
2	Ravinder	1008
3	Janila	1001
4	Naaz	1003

(*i*) *To display the names of all activities with their Acodes in descending order.*

(*ii*) *To display sum of PrizeMoney for the Activities played in each of the Stadium separately.*

(iii) *To display the coach's names and Acodes in ascending order of Acode from the table COACH.*

(iv) *To display the content of all activities for which ScheduleDate is earlier than 01-01-2004 in ascending order of ParticipantsNum.*

Solution.

(i)
```
SELECT Acode, ActivityName FROM ACTIVITY
ORDER BY Acode DESC ;
```

(ii)
```
SELECT Stadium, SUM(PrizeMoney) FROM ACTIVITY
GROUP BY Stadium ;
```

(iii)
```
SELECT Name, Acode FROM COACH
ORDER By Acode ;
```

(iv)
```
SELECT * FROM ACTIVITY WHERE SchduleDate < '01-Jan-2004'
ORDER BY ParticipantsNum ;
```

17. *Consider the following table GAMES. Write SQL commands for the following statements.* [CBSE SP II. 12]

Table : **GAMES**

GCode	GameName	Type	Number	PrizeMoney	ScheduleDate
101	Carom Board	Indoor	2	5000	23-Jan-2004
102	Badminton	Outdoor	2	12000	12-Dec-2003
103	Table Tennis	Indoor	4	8000	14-Feb-2004
105	Chess	Indoor	2	9000	01-Jan-2004
108	Lawn Tennis	Outdoor	4	25000	19-Mar-2004

(i) *To display the name of all GAMES with their GCodes*

(ii) *To display details of those GAMES which are having PrizeMoney more than 7000.*

(iii) *To display the content of the GAMES table in ascending order of ScheduleDate.*

(iv) *To display sum of PrizeMoney for each Type of GAMES.*

Solution.

(i)
```
SELECT GameName, GCode FROM GAMES ;
```

(ii)
```
SELECT * FROM GAMES
WHERE PrizeMoney > 7000 ;
```

(iii)
```
SELECT * FROM GAMES
ORDER BY ScheduleDate ;
```

(iv)
```
SELECT SUM(PrizeMoney), Type FROM GAMES
GROUP BY Type ;
```

18. *Consider the following tables CABHUB and CUSTOMER. Write SQL commands for the following statements.* [CBSE D 12]

Table : **CABHUB**

Vcode	VehicleName	Make	Color	Capacity	Charges
100	Innova	Toyota	WHITE	7	15
102	SX4	Suzuki	BLUE	4	14
104	C Class	Mercedes	RED	4	35
105	A-Star	Suzuki	WHITE	3	14
108	Indigo	Tata	SILVER	3	12

Table : **CUSTOMER**

CCode	CName	Vcode
1	Hemant Sahu	101
2	Raj Lal	108
3	Feroza Shah	105
4	Ketan Dhal	104

(i) *To display the names of all the white colored vehicles.*

(ii) *To display name of vehicle, make and capacity of vehicles in ascending order of their seating capacity.*

(iii) *To display the highest charges at which a vehicle can be hired from CABHUB.*

(iv) *To display the customer name and the corresponding name of the vehicle hired by them.*

Solution.

(i)
```
SELECT VehicleName
FROM CABHUB
WHERE Color = "WHITE" ;
```

(ii)
```
SELECT VehicleName, Make, Capacity
FROM CABHUB
ORDER BY Capacity ;
```

(iii)
```
SELECT Max(Charges)
FROM CABHUB ;
```

(iv)
```
SELECT CName, VehicleName
FROM CUSTOMER, CABHUB
WHERE CUSTOMER.Vcode = CABHUB.Vcode ;
```

19. *Consider the following tables ACTIVITY and COACH and answer the following parts of this question :* [CBSE Sample Paper L 12]

Table : ACTIVITY

Acode	ActivityName	Stadium	ParticipantsNum	PrizeMoney	ScheduleDate
1001	Relay 100 × 4	Star Annex	16	10000	23-Jan-04
1002	High jump	Star Annex	10	12000	12-Dec-03
1003	Shot Put	Super Power	12	8000	14-Feb-04
1005	Long Jump	Star Annex	12	9000	01-Jan-04
1008	Discuss Throw	Super Power	10	15000	19-Mar-04

Table : COACH

Pcode	Name	Acode
1	Ahmad Hussain	1001
2	Ravinder	1008
3	Janila	1001
4	Naaz	1003

Give the output of the following SQL queries :

(*i*) SELECT COUNT (DISTINCT ParticipantsNum) FROM ACTIVITY ;

(*ii*) SELECT MAX(ScheduleDate), MIN(ScheduleDate) FROM ACTIVITY ;

(*iii*) SELECT Name, ActivityName FROM ACTIVITY A, COACH C
WHERE A.Acode = C.Acode AND A.ParticipantsNum = 10 ;

(*iv*) SELECT DISTINCT ParticipantsNum FROM ACTIVITY ;

Solution.

(*i*) 3

(*ii*)
```
MAX(ScheduleDate)      MIN(ScheduleDate)
19-Mar-04              12-Dec-03
```

(*iii*)
```
Name        ACTIVITYNAME
Ravinder    Discuss Throw
```

(*iv*)
```
16
10
12
```

20. *Consider the following tables GAMES and PLAYER and answer the following parts of this question :* [CBSE Sample Paper II, 12]

Table : GAMES

GCode	GameName	Type	Number	PrizeMoney	ScheduleDate
101	Carom Board	Indoor	2	5000	23-Jan-2004
102	Badminton	Outdoor	2	12000	12-Dec-2003
103	Table Tennis	Indoor	4	8000	14-Feb-2004
105	Chess	Indoor	2	9000	01-Jan-2004
108	Lawn Tennis	Outdoor	4	25000	19-Mar-2004

Table : PLAYER

PCode	Name	GCode
1	Nabi Ahmad	101
2	Ravi Sahai	108
3	Jatin	101
4	Nazneen	103

Give the output of the following SQL queries :

(*i*) SELECT COUNT (DISTINCT Number) FROM GAMES ;

(*ii*) SELECT MAX(ScheduleDate), MIN(ScheduleDate) FROM GAMES ;

(*iii*) SELECT Name, GameName FROM GAMES G, PLAYER P
WHERE G.GCode = P.GCode AND G.PrizeMoney > 10000 ;

(*iv*) SELECT DISTINCT GCode FROM PLAYER ;

Solution.

(*i*) 2

(*ii*)
```
MAX(ScheduleDate)      MIN(ScheduleDate)
19-Mar-2004            12-Dec-2003
```

(*iii*)
```
Ravi Sahai      Lawn Tennis
```

(*iv*)
```
101
108
103
```

21. *Consider the following tables CABHUB and CUSTOMER and answer the following parts of this question :*

Table : CABHUB

Vcode	VehicleName	Make	Color	Capacity	Charges
100	Innova	Toyota	WHITE	7	15
102	SX4	Suzuki	BLUE	4	14
104	C Class	Mercedes	RED	4	35
105	A-Star	Suzuki	WHITE	3	14
108	Indigo	Tata	SILVER	3	12

Table : CUSTOMER

CCode	CName	Vcode
1	Hemant Sahu	101
2	Raj Lal	108
3	Feroza Shah	105
4	Ketan Dhal	104

Give the output of the following SQL queries :

(*i*) SELECT COUNT (DISTINCT Make) FROM CABHUB ;

(*ii*) SELECT MAX(Charges), MIN(Charges) FROM CABHUB ;

(*iii*) SELECT COUNT(*), Make FROM CABHUB ;

(*iv*) SELECT VehicleName FROM CABHUB WHERE Capacity = 4 ;

Solution.

(*i*) 4

(*ii*) 35 12

(*iii*) Invalid query

(*iv*)
```
SX4
C Class
```

22. *Consider the following tables WORKER and PAYLEVEL and answer the following parts of this* [CBSE D 11]

Table : WORKER

ECODE	NAME	DESIG	PLEVEL	DOJ	DOB
11	Radhe Shyam	Supervisor	P001	13-Sep-2004	23-Aug-1981
12	Chander Nath	Operator	P003	22-Feb-2010	12-Jul-1987
13	Fizza	Operator	P003	14-Jun-2009	14-Oct-1983
15	Ameen Ahmed	Mechanic	P002	21-Aug-2006	13-Mar-1984
18	Sanya	Clerk	P002	19-Dec-2005	09-Jun-1983

Table : PAYLEVEL

PLEVEL	PAY	ALLOWANCE
P001	26000	12000
P002	22000	10000
P003	12000	6000

Give the output of the following SQL queries :

(i) SELECT COUNT (PLEVEL), PLEVEL FROM WORKER GROUP BY PLEVEL ;

(ii) SELECT MAX(DOB), MIN(DOJ) FROM WORKER ;

(iii) SELECT Name, Pay FROM WORKER W, PAYLEVEL P
WHERE W.PLEVEL = P.PLEVEL AND W.ECODE < 13 ;

(iv) SELECT PLEVEL, PAY + ALLOWANCE FROM PAYLEVEL WHERE PLEVEL = 'P003' ;

Solution.

(i)
```
1       P001
2       P003
2       P002
```

(ii)
```
12-Jul-1987     13-Sep-2004
```

(iii)
```
Name            Pay
Radhe Shyam     25000
Chander Nath    12000
```

(iv)
```
P003    18000
```

23. *Consider the following tables EMPLOYEE and SALGRADE and answer the following parts of th* [CBSE OD 11]

Table : EMPLOYEE

ECODE	NAME	DESIG	SGRADE	DOJ	DOB
101	Abdul Ahmad	EXECUTIVE	S03	23-Mar-2003	13-Jan-1980
102	Ravi Chander	HEAD-IT	S02	12-Feb-2010	22-Jul-1987
103	John Ken	RECEPTIONIST	S03	24-Jun-2009	24-Feb-1983
105	Nazar Ameen	GM	S02	11-Aug-2006	03-Mar-1984
108	Priyam Sen	CEO	S01	29-Dec-2004	19-Jan-1982

Table : SALGRADE

SGRADE	SALARY	HRA
S01	56000	18000
S02	32000	12000
S03	24000	8000

Give the output of the following SQL queries :

(i) SELECT COUNT (SGRADE), SGRADE FROM EMPLOYEE GROUP BY SGRADE ;

(ii) SELECT MIN(DOB), MAX(DOJ) FROM EMPLOYEE ;

(iii) SELECT NAME, SALARY FROM EMPLOYEE E, SALGRADE S
WHERE E.SGRADE = S.SGRADE AND E.ECODE < 103 ;

(iv) SELECT SGRADE, SALARY + HRA FROM SALGRADE WHERE SGRADE = 'S02' ;

Solution.

(i)
```
COUNT   SGRADE
2       S03
2       S02
1       S01
```

(ii)
```
13-Jan-1980     12-Feb-2010
```

(iii)
```
NAME                SALARY
Abdul Ahmad         24000
Ravi Chander        32000
```

(iv)
```
SGRADE      SALARY + HRA
S02         44000
```

24. *Consider the following tables STORE and SUPPLIERS and answer the following parts of this question :*

Table : STORE

ItemNo	Item	Scode	Qty	Rate	LastBuy
2005	Sharpener Classic	23	60	8	31-Jun-09
2003	Ball Pen 0.25	22	50	25	01-Feb-10
2002	Gel Pen Premium	21	150	12	24-Feb-10
2006	Gel Pen Classic	21	250	20	11-Mar-09
2001	Eraser Small	22	220	6	19-Jan-09
2004	Eraser Big	22	110	8	02-Dec-09
2009	Ball Pen 0.5	21	180	18	03-Nov-09

Table : SUPPLIERS

Scode	Sname
21	Premium Stationers
23	Soft Plastics
22	Tetra Supply

Give the output of the following SQL queries :

(i) SELECT COUNT (DISTINCT Scode) FROM Store ;

(ii) SELECT Rate * Qty FROM Store WHERE ItemNo = 2004 ;

(iii) SELECT Item, Sname FROM Store S, Suppliers P
WHERE S.Scode = P.Scode AND ItemNo = 2006 ;

(iv) SELECT MAX (LastBuy) FROM Store ;

Solution.

(i)
```
3
```

(ii)
```
880
```

(iii)
```
Gel Pen Classic     Premium Stationers
```

(iv)
```
24-Feb-10
```

GLOSSARY

Single Row function	A function that works on a value in single row.
Multiple Row function	A function that works on values of a group of rows.
Group function	Function that works on a group of rows. A multiple row function.
Aggregate function	Function that works on aggregate of rows. A multiple row function.

Assignment

Type A : Short Answer Questions/Conceptual Questions

1. What is the use of ORDER BY clause?
2. What is the default sort order of ORDER BY clause ?
3. Write an example query that sorts on three columns.
4. Write a query that sorts the data of table **student** on the basis of Project-Group (in ascending order), section (in descending order), Marks (in descending order).
5. What is the difference between HAVING and WHERE clause?
6. What is the use of GROUP BY clause?
7. What are aggregate functions? What is their use? Give some examples.
8. What type of functions can you use with GROUP BY and HAVING clauses?

Type B : Application Based Questions

Following questions are based on these tables :

Table ***BOOK_INFORMATION***

Column Name
BOOK_ID
BOOK_TITLE
PRICE

Table ***SALES***

Column Name
STORE_ID
SALES_DATE
SALES_AMOUNT

Table ***EXAM_RESULTS***

STU_ID	FNAME	LNAME	EXAM_ID	EXAM_SCORE
10	LAURA	LYNCH	1	90
10	LAURA	LYNCH	2	85
11	GRACE	BROWN	1	78
11	GRACE	BROWN	2	72
12	JAY	JACKSON	1	95
12	JAY	JACKSON	2	92
13	WILLIAM	BISHOP	1	70
13	WILLIAM	BISHOP	2	100
14	CHARLES	PRADA	2	85

1. Which SQL statement allows you to find the highest price from the table BOOK_INFORMATION?
   (a) `SELECT BOOK_ID, BOOK_TITLE, MAX(PRICE) FROM BOOK_INFORMATION;`
   (b) `SELECT MAX(PRICE) FROM BOOK_INFORMATION;`
   (c) `SELECT MAXIMUM(PRICE) FROM BOOK_INFORMATION;`
   (d) `SELECT PRICE FROM BOOK_INFORMATION ORDER BY PRICE DESC;`
2. Which SQL statement lets you find the sales amount for each store?
   (a) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES;`
   (b) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES ORDER BY STORE_ID;`
   (c) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID;`
   (d) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES HAVING UNIQUE STORE_ID;`
3. Which SQL statement lets you list all stores whose total sales amount is over 5000 ?
   (a) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID HAVING SUM(SALES_AMOUNT) > 5000;`
   (b) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID HAVING SALES_AMOUNT > 5000;`
   (c) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE SUM(SALES_AMOUNT) > 5000 GROUP BY STORE_ID;`
   (d) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE SALES_AMOUNT > 5000 GROUP BY STORE_ID;`
4. Which SQL statement lets you find the total number of stores in the SALES table?
   (a) `SELECT COUNT(STORE_ID) FROM SALES;`
   (b) `SELECT COUNT(DISTINCT STORE_ID) FROM SALES;`
   (c) `SELECT DISTINCT STORE_ID FROM SALES;`
   (d) `SELECT COUNT(STORE_ID) FROM SALES GROUP BY STORE_ID;`
5. Which SQL statement allows you to find the total sales amount for Store ID 25 and the total sales amount for Store ID 45?
   (a) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE STORE_ID IN (25,45) GROUP BY STORE_ID;`
   (b) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES GROUP BY STORE_ID HAVING STORE_ID IN (25,45);`
   (c) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE STORE_ID IN (25,45);`
   (d) `SELECT STORE_ID, SUM(SALES_AMOUNT) FROM SALES WHERE STORE_ID = 25 AND STORE_ID = 45 GROUP BY STORE_ID;`
6. What SQL statement do we use to find the average exam score for EXAM_ID = 1?
   (a) `SELECT AVG(EXAM_SCORE) FROM EXAM_RESULTS;`
   (b) `SELECT AVG(EXAM_SCORE) FROM EXAM_RESULTS GROUP BY EXAM_ID WHERE EXAM_ID = 1;`
   (c) `SELECT AVG(EXAM_SCORE) FROM EXAM_RESULTS GROUP BY EXAM_ID HAVING EXAM_ID = 1;`
   (d) `SELECT COUNT(EXAM_SCORE) FROM EXAM_RESULTS WHERE EXAM_ID = 1;`

7. Which SQL statement do we use to find out how many students took each exam?
   (*a*) `SELECT COUNT(DISTINCT STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;`
   (*b*) `SELECT EXAM_ID, MAX(STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;`
   (*c*) `SELECT EXAM_ID, COUNT(DISTINCT STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;`
   (*d*) `SELECT EXAM_ID, MIN(STU_ID) FROM EXAM_RESULTS GROUP BY EXAM_ID;`

8. What SQL statement do we use to print out the record of all students whose last name starts with 'L'?
   (*a*) `SELECT * FROM EXAM_RESULTS WHERE LNAME LIKE 'L%';`
   (*b*) `SELECT * FROM EXAM_RESULTS WHERE LNAME LIKE 'L';`
   (*c*) `SELECT * FROM EXAM_RESULTS WHERE LNAME = 'L';`
   (*d*) `SELECT * FROM EXAM_RESULTS WHERE LNAME <> 'L';`

9. What is the result of the following SQL statement?

```
SELECT MAX(EXAM_SCORE) FROM EXAM_RESULTS GROUP BY EXAM_ID HAVING EXAM_ID = 1;
```

   (*a*) 90 (*b*) 85 (*c*) 100 (*d*) 95

10. Given the following table :

Table : CLUB

COACH-ID	COACHNAME	AGE	SPORTS	DATOFAPP	PAY	SEX
1.	KUKREJA	35	KARATE	27/03/1996	1000	M
2.	RAVINA	34	KARATE	20/01/1998	1200	F
3.	KARAN	34	SQUASH	19/02/1998	2000	M
4.	TARUN	33	BASKETBALL	01/01/1998	1500	M
5.	ZUBIN	36	SWIMMING	12/01/1998	750	M
6.	KETAKI	36	SWIMMING	24/02/1998	800	F
7.	ANKITA	39	SQUASH	20/02/1998	2200	F
8.	ZAREEN	37	KARATE	22/02/1998	1100	F
9.	KUSH	41	SWIMMING	13/01/1998	900	M
10.	SHAILYA	37	BASKETBALL	19/02/1998	1700	M

Give the output of following SQL statements :

(*i*) SELECT COUNT (DISTINCT SPORTS) FROM Club ;
(*ii*) SELECT MIN(Age) FROM CLUB WHERE Sex = 'F' ;
(*iii*) SELECT AVG(Pay) FROM CLUB WHERE Sports = 'KARATE' ;
(*iv*) SELECT SUM(Pay) FROM CLUB WHERE Datofapp > '31/01/98' ;

11. Given the following table :

Table : STUDENT

No.	Name	Stipend	Stream	AvgMark	Grade	Class
1	Karan	400.00	Medical	78.5	B	12B
2	Divakar	450.00	Commerce	89.2	A	11C
3	Divya	300.00	Commerce	68.6	C	12C
4	Arun	350.00	Humanities	73.1	B	12C
5	Sabina	500.00	Nonmedical	90.6	A	11A
6	John	400.00	Medical	75.4	B	12B
7	Robert	250.00	Humanities	64.4	C	11A
8	Rubina	450.00	Nonmedical	88.5	A	12A
9	Vikas	500.00	Nonmedical	92.0	A	12A
10	Mohan	300.00	Commerce	67.5	C	12C

Give the output of following SQL statements :

(*i*) SELECT MIN(AvgMark) FROM STUDENT WHERE AvgMark < 75 ;
(*ii*) SELECT SUM(Stipend) FROM Student WHERE Grade = 'B' ;
(*iii*) SELECT AVG(Stipend) FROM Student WHERE Class = '12A' ;
(*iv*) SELECT COUNT(DISTINCT) FROM Student ;

12. *In a Database, there are two tables given below :*

Table : EMPLOYEE

EMPLOYEEID	NAME	SALES	JOBID
E1	SUMIT SINHA	1100000	102
E2	VIJAY SINGH TOMAR	1300000	101
E3	AJAY RAJPAL	1400000	103
E4	MOHIT RAMNANI	1250000	102
E5	SHAILJA SINGH	1450000	103

Table : JOB

JOBID	JOBTITLE	SALARY
101	President	200000
102	Vice President	125000
103	Administration Assistant	80000
104	Accounting Manager	70000
105	Accountant	65000
106	Sales Manager	80000

Write SQL Queries for the following :

(*i*) *To display employee ids, names of employees, job ids with corresponding job titles.*
(*ii*) *To display names of employees, sales and corresponding job titles who have achieved sales more than 1300000.*
(*iii*) *To display names and corresponding job titles of those employees who have 'SINGH' (anywhere) in their names.*
(*iv*) *Identify foreign key in the table EMPLOYEE.*
(*v*) Write SQL command to change the JOBID to 104 of the EMPLOYEE with ID as E4 in the table 'EMPLOYEE'. [CBSE D 15]

13. Show the average salary for all departments with more than 3 people for a job.
14. Display only the jobs with maximum salary greater than or equal to 3000.
15. Find out number of employees having "Manager" as Job.
16. List the count of employees grouped by deptno. (table EMPL)
17. List the sum of employees' salaries grouped by department. (table EMPL)
18. List the maximum salary of employee grouped by their department number.

Consider the tables Customers, Parts and Orders given in solved problems.
Answer Questions 19 to 24 based on these :

19. List the total of customers' orders grouped by customer (id).
20. List all customers (name) who have orders (use EXISTS).
21. List the sum of the totals of orders grouped by customer and state.
22. List the sum of the totals of orders where this sum is greater than $1000 grouped by customer (id) and state and ordered by state.
23. List the customers (name) and their orders' details.
24. List the customers (name) and the total amount of all their orders.

Consider tables EMPL, Dept, SalaryGrade. The Empl table has already been listed in earlier chapters. Schemas of tables SalaryGrade and Dept are being shown below :

```
SALARYGRADE (Lowsal, Highsal, Grade)
Dept (Deptno, DeptName, Location)
```

Answer Questions 25 and 26 on the basis of these tables :

25. List the department names and the number of their employees.
26. List the employee names and the name of their departments.
27. Give output for following SQL queries as per given table(s) : [CBSE D 09]

Table : GARMENT

GCODE	Description	Price	FCODE	READYDATE
10023	PENCIL SKIRT	1150	F03	19-DEC-08
10001	FORMAL SHIRT	1250	F01	12-JAN-08
10012	INFORMAL SHIRT	1550	F02	06-JUN-08
10024	BABY TOP	750	F03	07-APR-07
10090	TULIP SKIRT	850	F02	31-MAR-07
10019	EVENING GOWN	850	F03	06-JUN-08
10009	INFORMAL PANT	1500	F02	20-OCT-08
10017	FORMAL PANT	1350	F01	09-MAR-08
10020	FROCK	850	F04	09-SEP-07
10089	SLACKS	750	F03	31-OCT-08

Table : FABRIC

FCODE	TYPE
F04	POLYSTER
F02	COTTON
F03	SILK
F01	TERELENE

(*i*) SELECT SUM (PRICE) FROM GARMENT WHERE FCODE = 'F01' ;
(*ii*) SELECT DESCRIPTION, TYPE FROM GARMENT, FABRIC
WHERE GARMENT.FCODE = FABRIC.FCODE AND GARMENT.PRICE >= 1260 ;
(*iii*) SELECT MAX (FCODE) FROM FABRIC ;
(*iv*) SELECT COUNT (DISTINCT PRICE) FROM GARMENT ;

28. Give output for following SQL queries as per given table(s) : [CBSE D 08]

Table : PRODUCT

P_ID	ProductName	Manufacturer	Price
TP01	Talcom Powder	LAK	40
FW05	Face Wash	ABC	45
BS01	Bath Soap	ABC	55
SH06	Shampoo	XYZ	120
FW12	Face Wash	XYZ	95

Table : CLIENT

C_ID	ClientName	City	P_ID
01	Cosmetic Shop	Delhi	FW05
06	Total Health	Mumbai	BS01
12	Live Life	Delhi	SH06
15	Pretty Woman	Delhi	FW12
16	Dreams	Banglore	TP01

(*i*) SELECT DISTINCT City FROM Client ;
(*ii*) SELECT Manufacturer, MAX(Price), Min(Price), Count(*)
FROM Product GROUP BY Manufacturer ;
(*iii*) SELECT ClientName, ProductName
FROM Product, Client
WHERE Client.P_Id = Product.P_Id ;
(*iv*) SELECT ProductName, Price * 4 FROM Product ;

29. Give output for following SQL queries as per given table(s) : [CBSE OD 08]

Table : ITEM

I_ID	ItemName	Manufacturer	Price
PC01	Personal Computer	ABC	35000
LC05	Laptop	ABC	55000
PC03	Personal Computer	XYZ	32000
PC06	Personal Computer	COMP	37000
LC03	Laptop	PQR	57000

Table : CUSTOMER

C_ID	CustomerName	City	I_ID
01	N Roy	Delhi	LC03
06	H Singh	Mumbai	PC03
12	R Pandey	Delhi	PC06
15	C Sharma	Delhi	LC03
16	K Agarwal	Bangalore	PC01

(i) SELECT DISTINCT City FROM Customer ;

(ii) SELECT ItemName, MAX(Price), Count(*)
FROM Item GROUP BY ItemName ;

(iii) SELECT CustomerName, Manufacturer
FROM Item, Customer
WHERE Item.I_ID = Customer.I_ID ;

(iv) SELECT ItemName, Price * 100
FROM Item WHERE Manufacturer = 'ABC' ;

30. Give output for following SQL queries as per given table(s) : [CBSE D 07]

Table : SENDER

SenderID	SenderName	SenderAddress	SenderCity
ND01	R Jain	2, ABC Appts	New Delhi
MU02	H Sinha	12, Newtown	Mumbai
MU15	S Jha	27/A, Park Street	Mumbai
ND50	T Prasad	122-K, SDA	New Delhi

Table : RECIPIENT

RecID	SenderID	RecName	RecAddress	RecCity
KO05	ND01	R Bajpayee	5, Central Avenue	Kolkata
ND08	MU02	S Mahajan	116, A Vihar	New Delhi
MU19	ND01	H Singh	2A, Andheri East	Mumbai
MU32	MU15	P K Swamy	B5, C S Terminus	Mumbai
ND48	ND50	S Tripathi	13, B1 D, Mayur Vihar	New Delhi

(i) SELECT DISTINCT SenderCity FROM Sender ;

(ii) SELECT A.SenderName, B.RecName
FROM Sender A, Recipient B
WHERE A.SenderID = B.SenderID AND B.RecCity = 'Mumbai' ;

(iii) SELECT RecName, RecAddress
FROM Recipient
WHERE RecCity NOT IN('Mumbai', 'Kolkata') ;

(iv) SELECT RecID, RecName
FROM Recipient
WHERE SenderID = 'MU02' OR SenderID = 'ND50' ;

8 JOINS and SET Operations

Outline

8.1 Introduction
8.2 Joining Tables
8.3 Performing SET Operations on Relations

8.1 Introduction

So friends! How are you finding SQL ? Isn't it easy and interesting? See, how easy it becomes for you to extract desired information from a relation. But you must have encountered some situations where the simple queries could not work up to your expectations. For instance, you might have wanted to extract columns from more than one table or to group records coming from multiple relations/tables or to find common records of two relations/tables.

This chapter will try to answer such type of problems. Two important concepts of SQL – JOINS and SET Operations (*UNION, INTERSECTION, MINUS*) will be covered in this chapter. Let us begin with the Joins on relations. (Note, in this chapter we shall use the words relations and tables interchangeably.)

8.2 JOINING TABLES

A ***join*** is a query that combines rows from two or more tables. In a join-query, more than one table are listed in the FROM clause of SELECT query. The function of combining data from multiple tables is called **joining**. The joining of two tables can be ***restricted*** (*i.e.*, based on some condition) or ***unrestricted*** (*i.e.*, no binding condition). Let us talk about these *two* ways of joining tables.

8.2.1 Unrestricted Join (Cartesian Product)

When two tables are joined without any binding condition, such type of join is called **unrestricted join** and the result produced is called the **Cartesian product.** A Cartesian product of two tables consists of all possible combinations of rows coming from two tables.

Consider the following query :

```
SELECT *
FROM EMP1, DEPT ;
```

← *See, no condition is given (WHERE clause is missing); it will result into* ***Cartesian product***

This query will give you the Cartesian product *i.e.*, all possible concatenations are formed of all rows of both the tables EMP1 and DEPT. That is, when no particular rows (using WHERE clause) and columns (through SELECT list) are selected. Such an operation is also known as **Unrestricted Join.** It returns $n1 \times n2$ rows where $n1$ is the number of rows in first table and $n2$ is the number of rows in second table.

A Cartesian product is formed when :

- A join condition is omitted
- A join condition is invalid
- All rows in the first table are joined to all rows in the second table.

But joining is not this at all. It is much more useful in many situations where we need to derive data from more than one table.

Let us talk about restricted joins.

8.2.2 Restricted Join (Join)

In a SELECT query, fetching rows from two tables, if we put a restriction by adding a condition (WHERE clause), it will no longer be the Cartesian product. It will now be called a ***restricted join*** or simply a **join**.

Consider the **empl** table, you have been using in earlier chapters :

CARTESIAN PRODUCT

"The **unrestricted join** or **Cartesian product** of two tables is a relation that contains all possible concatenations of all rows of both the participating relations(tables)."

NOTE

The cardinality of a Cartesian product is the product of cardinalities of the two participating tables, *i.e.*, $n1 \times n2$ where $n1$ and $n2$ are cardinalities of the two participating tables.

JOIN

"A join is a query that combines rows from two or more tables based on a condition."

Table 8.1 *Database table empl*

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
8369	SMITH	CLERK	8902	1990-12-18	800.00	NULL	20
8499	ANYA	SALESMAN	8698	1991-02-20	1600.00	300.00	30
8521	SETH	SALESMAN	8698	1991-02-22	1250.00	500.00	30
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	20
8654	MOMIN	SALESMAN	8698	1991-09-28	1250.00	1400.00	30
8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	30
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	10
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500.00	0.00	30
8882	SHIAVNSH	MANAGER	8839	1991-06-09	2450.00	NULL	10
8886	ANOOP	CLERK	8888	1993-01-12	1100.00	NULL	20
8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	20
8900	JATIN	CLERK	8698	1991-12-03	950.00	NULL	30
8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	20
8934	MITA	CLERK	8882	1992-01-23	1300.00	NULL	10

Looking at the EMPL table we can see that we have a column called DEPTNO. Suppose we also have a table called DEPT which carries attributes such as *department number, department name,* and *location* of the department. Here is the data stored in that table :

```
SELECT * FROM DEPT ;
```

Table 8.2 *Dept*

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW DELHI
20	RESEARCH	CHENNAI
30	SALES	KOLKATA
40	OPERATIONS	MUMBAI
50	MARKETING	BANGLORE
60	PRODUCTION	AHMEDABAD

Now suppose we need to know the office location for the employee named 'ANOOP'.

In reviewing the DEPT table above we note that department 20 is in 'CHENNAI' but the table does not include the *employee's name field.*

By reviewing the EMPL table (Table 8.1) we can see that 'ANOOP' works in department 20 but the location of that department is not contained within this table.

Now the question is : *How do we get this information without doing two separate SELECT statements ?*

And the answer is *Join-Query*. Let us see how we can do this.

Since both the EMPL table and the DEPT table carry the column DEPTNO, we can join the tables on this common column and relate rows from the DEPT table with rows in the EMPL table. In doing this, we are creating a virtual table (a table created in main memory), which contains all the attributes for rows from the DEPT table and the EMPL table where the DEPTNO is equal. This virtual table contains the following attributes : DEPTNO, DNAME, LOC, EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM.

In MySQL we are able to join a multitude of tables together in this fashion. Although it may not be practical, but it is possible. Be aware that the more tables we join, the more complicated the query becomes.

In order to find the location of the employee named ANOOP, we would use the following query.

```
mysql> SELECT ENAME, LOC
       FROM EMPL, DEPT
       WHERE ENAME = 'ANOOP'
       AND EMPL.DEPTNO = DEPT.DEPTNO ;        -- see this is join condition
```

Additional condition along with the join condition.

```
+--------+---------+
| ename  | loc     |
+--------+---------+
| ANOOP  | CHENNAI |
+--------+---------+
1 row in set (0.13 sec)
```

The virtually joined table that we created, consists of one row that meets the specifications of the WHERE clause (ENAME = 'ANOOP'). Since we selected only ENAME and LOC fields, only those columns were returned. However, we can retrieve as many attributes from each table as we need.

Thus, we can say that a *join* is used when an SQL query requires data from more than one table on database. *Joins* can compare two or more tables (or views) by specifying a column from each, comparing the values in those columns row by row and concatenating rows that have matching values. The FROM clause is used to specify which tables and views have to be joined. The columns being joined must have same or comparable datatypes.

Let us consider one more example. If you want to know the *name, job, deptno, deptname* and *dept-location* of all the employees, what would you do ? Yes, you are right : you have to use joins and you need to join the two tables **empl** and **dept** on the basis of their common field **deptno**.

And the query will be :

EXAMPLE 1 *Write query to join two tables empl and dept on the basis of field deptno.*

```
mysql> SELECT * FROM empl, dept
    -> WHERE empl.deptno = dept.deptno ;
```

And the output produced by above join query will be :

See two identical columns

empno	ename	job	mgr	hiredate	sal	comm	deptno	deptno	dname	loc
8369	SMITH	CLERK	8902	1990-12-18	800.00	NULL	20	20	RESEARCH	CHENNAI
8499	ANYA	SALESMAN	8698	1991-02-20	1600.00	300.00	30	30	SALES	KOLKATA
8521	SETH	SALESMAN	8698	1991-02-22	1250.00	500.00	30	30	SALES	KOLKATA
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	20	20	RESEARCH	CHENNAI
8654	MOMIN	SALESMAN	8698	1991-09-28	1250.00	1400.00	30	30	SALES	KOLKATA
8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	30	30	SALES	KOLKATA
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	10	10	ACCOUNTING	NEW DELHI
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500.00	0.00	30	30	SALES	KOLKATA
8882	SHIVANSH	MANAGER	8839	1991-06-09	2450.00	NULL	10	10	ACCOUNTING	NEW DELHI
8886	ANOOP	CLERK	8888	1993-01-12	1100.00	NULL	20	20	RESEARCH	CHENNAI
8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	20	20	RESEARCH	CHENNAI
8900	JATIN	CLERK	8698	1991-12-03	950.00	NULL	30	30	SALES	KOLKATA
8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	20	20	RESEARCH	CHENNAI
8914	RITA	CLERK	8882	1992-01-23	1300.00	NULL	10	10	ACCOUNTING	NEW DELHI

14 rows in set (0.01 sec)

Hey, did you notice, there are two fields by the name *deptno* – one deptno from table **empl** and one from table **dept**. This happened because we specified * (asterisk) in the select-list of our join query. If you specifically mention the field names in the select-list of join query, then only the specified fields will appear. That is, if you change above query to following query then duplicate **deptno** column will get removed :

```
SELECT empl.*, dname, loc
FROM empl, dept
WHERE empl.deptno = dept.deptno ;
```

This means that *empl.** i.e., all the fields of table *empl* along with *dname, loc* fields (from table dept) should appear in the output.

Qualified Names

Did you notice that in all the WHERE conditions of Join-queries given so far, the field names are given as :

```
<tablename>.<fieldname>
```

This type of field names are called **qualified field names**. Qualified field names are very useful in identifying a field if the two joining tables have fields with same name. For example, if we say **deptno** field from joining tables **empl** and **dept**, you'll definitely ask – '**deptno** field of which table ?' To avoid such an ambiguity, the qualified field names are used.

Therefore, in the above given queries, the columns that have similar names in both the tables have been given *qualified names i.e., table-name.column-name*, to avoid ambiguity. If the tables do not have the columns with the same name, using table names as qualifiers in the SELECT statement is optional.

NOTE

Qualifying a column name uniquely identifies it from similar columns in other tables.

The FROM clause lists all the tables involved and WHERE clause specifies the join. Without a WHERE clause, each row of the first table will be joined with every row of the second table which results in a *Cartesian Product*. Multiple conditions can be incorporated in the WHERE clause using the logical operators AND, OR and NOT.

And the relational operators =, >, <, >=, <=, <> are used as join operators.

Now consider the following example.

EXAMPLE 2 *Display details like department number, department name, employee number, employee name, job, and salary. And order the rows by employee number with department number.*

SOLUTION

```
mysql> SELECT EMPL.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
       FROM EMPL, DEPT
       WHERE EMPL.DEPTNO = DEPT.DEPTNO        -- join on DEPTNO
       ORDER BY EMPL.DEPTNO, EMPNO ;
```

The output produced by this query will be as shown below :

```
+-------+------------+------+----------+-----------+----------+
|deptno |dname       |empno |  ename   |    job    |sal       |
+-------+------------+------+----------+-----------+----------+
| 10    | ACCOUNTING | 8839 | AMIR     | PRESIDENT |  5000.00 |
| 10    | ACCOUNTING | 8882 | SHIAVNSH | MANAGER   |  2450.00 |
| 10    | ACCOUNTING | 8934 | MITA     | CLERK     |  1300.00 |
| 20    | RESEARCH   | 8369 | SMITH    | CLERK     |   800.00 |
| 20    | RESEARCH   | 8566 | MAHADEVAN| MANAGER   |  2985.00 |
| 20    | RESEARCH   | 8886 | ANOOP    | CLERK     |  1100.00 |
| 20    | RESEARCH   | 8888 | SCOTT    | ANALYST   |  3000.00 |
| 20    | RESEARCH   | 8902 | FAKIR    | ANALYST   |  3000.00 |
| 30    | SALES      | 8499 | ANYA     | SALESMAN  |  1600.00 |
| 30    | SALES      | 8521 | SETH     | SALESMAN  |  1250.00 |
| 30    | SALES      | 8654 | MOMIN    | SALESMAN  |  1250.00 |
| 30    | SALES      | 8698 | BINA     | MANAGER   |  2850.00 |
| 30    | SALES      | 8844 | KULDEEP  | SALESMAN  |  1500.00 |
| 30    | SALES      | 8900 | JATIN    | CLERK     |   950.00 |
+-------+------------+------+----------+-----------+----------+
14 rows in set (0.01 sec)
```

8.2.3 Using Table Aliases

Consider once again the query given in example 2. Don't you find something unusual about this query ? Since we wanted to list DEPTNO as part of the output and this attribute resides within each table, we had to tell MySQL which table (it doesn't matter which one) we wanted to use to extract DEPTNO. When you have an attribute, which resides in more than one table if you do not specify the table in the SELECT statement you will get an ambiguity error.

For example, had we given the following query for example 2 :

```
mysql> SELECT deptno, dname, empno, ename, job, sal
    -> FROM empl, dept
    -> WHERE empl.deptno = dept.deptno
    -> ORDER BY empl.deptno, empno ;
```

ambiguous column (from which table ? – not specified)

MySQL would have shown us the error as given below :

```
Column 'deptno' in field list is ambiguous
```

TABLE ALIAS

> A **Table Alias** is a temporary **label** given along with table name in FROM clause.

That means every time you have to refer to a column, which is in more than one table participating in the join query, you have to *qualify the column name by* giving its table name along with it. Doesn't this method require more typing since for every such column, table name is to be typed ? But don't worry, there is a shorter way of doing so – *the table aliases*. A table alias can be used anywhere in the SELECT statement.

To cut down on the amount of typing required in your queries you can use aliases for table names in the SELECT and WHERE clauses. For example, if you wanted to use the abbre- viation 'E' for the EMPL table in your query all you need to is tell MySQL that EMPL will be referenced by *E* in the FROM clause.

The following queries would result in the same output :

```
mysql> SELECT E.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
       FROM EMPL E, DEPT
       WHERE E.DEPTNO = DEPT.DEPTNO          -- table alias E used for table EMPL.
       ORDER BY E.DEPTNO, EMPNO ;            -- join on field DEPTNO
```

Table alias

```
mysql> SELECT E.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
       FROM EMPL E, DEPT D
                                             -- table alias E for table EMPL,
       WHERE E.DEPTNO = D.DEPTNO             -- table alias D for table DEPT
       ORDER BY E.DEPTNO, EMPNO ;            -- join on field DEPTNO
```

8.2.4 Additional Search Conditions in Joins

Once you have joined tables, you can filter information from joined table by incorporating additional search conditions. For example, refer to *example 8.9* once again. If you want to extract such information only for SALES department, you can do so by giving a query similar to the one given in example 3.

EXAMPLE 3 *Refer to example 2 Do this only for SALES department.*

SOLUTION

```
mysql> SELECT E.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
       FROM EMPL E, DEPT D WHERE E.DEPTNO = D.DEPTNO
       AND DNAME = 'SALES'  ORDER BY E.DEPTNO, EMPNO ;
```

```
+-------+-------+------+-------+---------+---------+
|deptno | dname | empno| ename |   job   |   sal   |
+-------+-------+------+-------+---------+---------+
|  30   | SALES | 8499 | ANYA  | SALESMAN| 1600.00 |
|  30   | SALES | 8521 | SETH  | SALESMAN| 1250.00 |
|  30   | SALES | 8654 | MOMIN | SALESMAN| 1250.00 |
|  30   | SALES | 8698 | BINA  | MANAGER | 2850.00 |
|  30   | SALES | 8844 | KULDEEP| SALESMAN| 1500.00 |
|  30   | SALES | 8900 | JATIN | CLERK   |  950.00 |
+-------+-------+------+-------+---------+---------+
```

EXAMPLE 4 *Display details like department number, department name, employee number, employee name, job, and salary. And order the rows by employee number with department number. These details should be only for employees earning atleast Rs. 1500 and of SALES department.*

SOLUTION

```
mysql> SELECT E.DEPTNO, DNAME, EMPNO, ENAME, JOB, SAL
       FROM EMPL E, DEPT D WHERE E.DEPTNO = D.DEPTNO
       AND DNAME = 'SALES' AND SAL >= 1500
       ORDER BY E.DEPTNO, EMPNO ;
```

```
+-------+-------+------+-------+---------+---------+
|deptno | dname | empno| ename |   job   |   sal   |
+-------+-------+------+-------+---------+---------+
|  30   | SALES | 8499 | ANYA  | SALESMAN| 1600.00 |
|  30   | SALES | 8698 | BINA  | MANAGER | 2850.00 |
|  30   | SALES | 8844 | KULDEEP| SALESMAN| 1500.00 |
+-------+-------+------+-------+---------+---------+
```

8.2.5 Joining More Than Two Tables

Sometimes you need to extract information from more than two tables. In this case, you need to join more than two tables. This can be achieved by specifying all the required table names in FROM clause and by providing all join conditions using AND operator. To understand this, consider the following example.

EXAMPLE 5 *Display the employee details in the following format :*

```
Ename Department Job Sal Grade.
```

Here the Department is department and Grade is the salary grade. The salary grades as per salary ranges are given in table SalaryGrade that contains following information.

Table SalaryGrade

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

That is, the salaries falling between range specified with losal and hisal have the corresponding grade e.g., salaries falling between 700 to 1200 have grade 1 ; between 1201 to 1400, have grade 2 ; and so on.

Frame an SQL query to attain the desired result.

SOLUTION

```
mysql> SELECT ENAME, DNAME, JOB, SAL, GRADE
       FROM EMPL E, DEPT D, SalaryGrade S
       WHERE E.DEPTNO = D.DEPTNO                -- first join condition
       AND SAL BETWEEN LOSAL AND HISAL          -- second join condition
       ORDER BY E.DEPTNO, EMPNO ;
```

```
+----------+-----------+----------+---------+-------+
| ename    | dname     | job      | sal     | grade |
+----------+-----------+----------+---------+-------+
| AMIR     | ACCOUNTING | PRESIDENT | 5000.00 |   5   |
| SHIAVNSH | ACCOUNTING | MANAGER   | 2450.00 |   4   |
| MITA     | ACCOUNTING | CLERK     | 1300.00 |   2   |
| SMITH    | RESEARCH   | CLERK     |  800.00 |   1   |
| MAHADEVAN | RESEARCH  | MANAGER   | 2985.00 |   4   |
| ANOOP    | RESEARCH   | CLERK     | 1100.00 |   1   |
| SCOTT    | RESEARCH   | ANALYST   | 3000.00 |   4   |
| FAKIR    | RESEARCH   | ANALYST   | 3000.00 |   4   |
| ANYA     | SALES      | SALESMAN  | 1600.00 |   3   |
| SETH     | SALES      | SALESMAN  | 1250.00 |   2   |
| MOMIN    | SALES      | SALESMAN  | 1250.00 |   2   |
| BINA     | SALES      | MANAGER   | 2850.00 |   4   |
| KULDEEP  | SALES      | SALESMAN  | 1500.00 |   3   |
| JATIN    | SALES      | CLERK     |  950.00 |   1   |
+----------+-----------+----------+---------+-------+
```


Notice that the second join condition is not equality based. We can have join conditions based on equality and other operators also. Depending upon the way these joins are created, they are called differently. Coming sections discuss such types of joins.

8.2.6 Equi Join

In an *equijoin*, the values in the columns being joined are *compared for equality*. All the columns in the tables-being-joined are included in the results.

> **EQUI-JOIN**
> "The Join, in which columns are compared for equality, is called **Equi-Join**."

Consider the following example.

Recall the output of example 1, that has double **deptno** fields. This was the result of an **equi-join**.

In equi-join, all the columns from joining table appear in the output even if they are identical.

8.2.7 Non-Equi-Joins

The comparison operator used in the join condition in Example 2 was the equals sign, "=". Such join queries are called *equi-joins*. A *non-equi-join* is a query that specifies some relationship other than equality between the columns.

If you refer to *example 5* carefully, you'll find that the second join condition is the example of a non-equi-join. If we had to join only EMPL table and SalaryGrade table, it would have been a non equijoin.

This has been illustrated in example 6.

EXAMPLE 6 *Display employee details for Analysts in the following format.*

```
Ename Job Sal Grade
```

SOLUTION

```
mysql> SELECT ENAME, JOB, SAL, GRADE
       FROM EMPL E, SalaryGrade S
       WHERE SAL BETWEEN LOSAL AND HISAL        -- non-equijoin condition
       AND JOB = 'ANALYST' ;
```

```
+-------+---------+---------+-------+
| ename |  job    |  sal    | grade |
+-------+---------+---------+-------+
| SCOTT | ANALYST | 3000.00 |   4   |
| FAKIR | ANALYST | 3000.00 |   4   |
+-------+---------+---------+-------+
```

8.2.8 Natural Join

By definition, the results of an equijoin contain two identical columns. One of the two identical columns can be eliminated by restating the query. This result is called a ***Natural Join*** (Equi join minus one of the two identical columns).

> **NATURAL JOIN**
> "The Join in which only one of the identical columns (coming from joined tables) exists, is called **Natural Join**."

For example, if you restate the query of example 1 as :

```
SELECT  empl.*, dname, loc
FROM    empl, Dept
WHERE   empl.deptno = dept.dept no ;
```

See this time no duplicate column

empno	ename	job	mgr	hiredate	sal	comm	deptno	dname	loc
8369	SMITH	CLERK	8902	1990-12-18	800.00	NULL	20	RESEARCH	CHENNAI
8499	ANYA	SALESMAN	8698	1991-02-20	1600.00	300.00	30	SALES	KOLKATA
8521	SETH	SALESMAN	8698	1991-02-22	1250.00	500.00	30	SALES	KOLKATA
8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	20	RESEARCH	CHENNAI
8654	MOMIN	SALESMAN	8698	1991-09-28	1250.00	1400.00	30	SALES	KOLKATA
8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	30	SALES	KOLKATA
8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	10	ACCOUNTING	NEW DELHI
8844	KULDEEP	SALESMAN	8698	1991-09-08	1500.00	0.00	30	SALES	KOLKATA
8882	SHIAVNSH	MANAGER	8839	1991-06-09	2450.00	NULL	10	ACCOUNTING	NEW DELHI
8886	ANOOP	CLERK	8888	1993-01-12	1100.00	NULL	20	RESEARCH	CHENNAI
8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	20	RESEARCH	CHENNAI
8900	JATIN	CLERK	8698	1991-12-03	950.00	NULL	30	SALES	KOLKATA
8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	20	RESEARCH	CHENNAI
8934	MITA	CLERK	8882	1992-01-23	1300.00	NULL	10	ACCOUNTING	NEW DELHI

14 rows in set (0.01 sec)

8.2.9 Joining Tables Using JOIN Clauses of SQL SELECT

Till now you have learnt how to create joined-tables using traditional SQL style. MySQL also supports special clauses – **JOIN (and NATURAL JOIN) clause** – as part of SQL SELECT statement. The syntax for using JOIN clasue of SQL SELECT :

```
SELECT *
FROM <table1>
[CROSS][NATURAL] JOIN <table2>
[ON (<Join-Condition>) | USING (<joinfields>)] ;
```

It works as follows :

- To create **Cartesian product** of two tables (say **empl** and **dept**) write query as :

```
SELECT *
FROM empl JOIN Dept ;
```

If no join-condition is specified with a JOIN clause, cartesian-product is output.

Cross Join

You can also produce cartesian product of two tables using CROSS JOIN clause as follows :

```
SELET *
FROM empl CROSS JOIN dept ;
```

The above query will produce cartesian product of tables *empl* and *dept*.

The cartesian product of two tables is also known as **CROSS JOIN**. The **cross join** (or **cartesian product**) is a very basic type of join that simply matches each row from one table to every row from another table. Cross join, by default, does not apply any filtering condition on the joined data. However, if you want to filter, you can use WHERE clause of SELECT query.

- To create equijoin on some common field (say **deptno**) of two tables (**empl** and **dept** here), write query as :

```
SELECT *
FROM empl e
JOIN dept d
ON (e.deptno = d.deptno) ;
```

Join condition is specified with ON clause.

CROSS JOIN

"The **cross join** (or cartesian product) is a very basic type of join that simply matches each row from one table to every row from another table."

You may write above query without table-aliases *i.e.*, as

```
SELECT *
FROM empl JOIN dept
ON (empl.deptno = dept.deptno);
```

Both the above queries will produce output similar to that of Example 1.

Natural Join

As you know that equi-join has two identical columns, to remove which you specify the desired columns in the **select-list**.

However, MySQL provides a NATURAL JOIN clause for you that does the same for you that is, one of the duplicate columns get removed from the output.

All you need to do is to write :

```
SELECT *
FROM <table1> ;
NATURAL JOIN <table2> ;
```

NOTE

Please note that with **NATURAL JOIN** clause, you need not specify a JOIN-Condition. It will automatically create natural-join through appropriate common field.

That is, above query will now become :

```
SELECT *
FROM empl NATURAL JOIN dept ;
```

with NATURAL JOIN clause, you need not specify Join Condition

Filtering Output

If you want to further filter the records from a join-output, you can write the condition with a WHERE clause. For instance, following query will first create a natural join from tables **empl** and **dept** and then extract only those records where **sal** is more than 2000 ;

```
mysql>  SELECT * FROM empl e
        NATURAL JOIN dept d
        WHERE sal > 2000 ;
```

*Two joining tables whose **Natural Join** is being produced.*

deptno	empno	ename	job	mgr	hiredate	sal	comm	dname	loc
20	8566	MAHADEVAN	MANAGER	8839	1991-04-02	2985.00	NULL	RESEARCH	CHENNAI
30	8698	BINA	MANAGER	8839	1991-05-01	2850.00	NULL	SALES	KOLKATA
10	8839	AMIR	PRESIDENT	NULL	1991-11-18	5000.00	NULL	ACCOUNTING	NEW DELHI
20	8882	SHIAVNSH	MANAGER	8839	1991-06-09	2450.00	NULL	ACCOUNTING	NEW DELHI
20	8888	SCOTT	ANALYST	8566	1992-12-09	3000.00	NULL	RESEARCH	CHENNAI
20	8902	FAKIR	ANALYST	8566	1991-12-03	3000.00	NULL	RESEARCH	CHENNAI

Similarly, you can use WHERE clause with JOIN to filter records from an equi-join. For example, following query will first create equi-join from tables *empl* and *dept* and then filter those records where *sal* is < 2000.

```
mysql>  SELECT * FROM empl JOIN dept ON (empl.deptno = dept.deptno)
    ->  WHERE sal < 2000 ;
```

```
+-----+------+--------+----+----------+-------+-------+---+---+--------+--------+
|empno|ename |job     |mgr |hiredate  |sal    |comm   |deptno|dname   |loc     |
+-----+------+--------+----+----------+-------+-------+---+---+--------+--------+
|8369 |SMITH |CLERK   |8902|1990-12-18|800.00 |NULL   |20 |20 |RESEARCH|CHENNAI |
|8499 |ANYA  |SALESMAN|8698|1991-02-20|1600.00|300.00 |30 |30 |SALES   |KOLKATA |
|8521 |SETH  |SALESMAN|8698|1991-02-22|1250.00|500.00 |30 |30 |SALES   |KOLKATA |
|8654 |MOMIN |SALESMAN|8698|1991-09-28|1250.00|1400.00|30 |30 |SALES   |KOLKATA |
|8844 |KULDEEP|SALESMAN|8698|1991-09-08|1500.00|  0.00 |30 |30 |SALES   |KOLKATA |
|8886 |ANOOP |CLERK   |8888|1993-01-12|1100.00|NULL   |20 |20 |RESEARCH|CHENNAI |
|8900 |JATIN |CLERK   |8698|1991-12-03| 950.00|NULL   |30 |30 |SALES   |KOLKATA |
|8934 |MITA  |CLERK   |8882| 1992-01-23|1300.00|NULL   |10 |10 |ACCOUNTING|NEW DELHI|
+-----+------+--------+----+----------+-------+-------+---+---+--------+--------+
8 rows in set (0.01 sec)
```

You can specify join fields through **USING** clause also in place of **ON** clause, but with **USING** clause you just have to specify the name of join-field whereas with **ON** clause, join-condition is specified.

That is above query can also be reframed as :

```
SELECT *
FROM empl
JOIN dept
USING (deptno) ;
```

USING subclause with JOIN clause produces a ***Natural join*** whereas ON subclause with JOIN clause produces ***equi-join***.

NOTE

Difference between ON and USING sub-clauses of JOIN clause of SELECT is that **ON** clause requires a complete join-condition whereas USING clause requires just the name of a join field. USING subclause produces *natural join* whereas ON clause produces *equi-join*.

8.2.9A Left, Right Joins

When you join tables based on some condition, you may find that only some, not all rows from either table match with rows of other table. When you display an equi-join or natural-join, it shows only the matched rows. What if you want to know which all rows from a table did not match with other. In such a case, MySQL left or right JOIN can be very helpful and it refers to the order in which the tables are put together and the results are displayed.

Left Join

You can use LEFT JOIN clause of SELECT to produce left join *i.e.*, as :

```
SELECT <select-list>
FROM <table1> LEFT JOIN <table2>
ON <joining-condition> ;
```

When using LEFT JOIN all rows from the first table will be returned whether there are matches in the second table or not. For unmatched rows of first table, NULL is shown in columns of

second table. For example, if table **members** contains members' main infor- mation and table **options** contains members' optional information, any records in **options** table would be tied to a particular *id* in the **members** table.

```
SELECT name, lastname, optonalinfo
FROM members LEFT JOIN options ON members.id = options.id ;
```

The result of this query would return *name* and *lastname* values from the **members** table and all available values from the **options** table. NULL is returned for non-existing values in **options** table. Consider the tables *empl* and *dept*. Table dept has now the following records (notice new records)

```
+---------+-------------+-----------+
| deptno  |    dname    |    loc    |
+---------+-------------+-----------+
|  10     | ACCOUNTING  | NEW DELHI |
|  20     | RESEARCH    | CHENNAI   |
|  30     | SALES       | KOLKATA   |
|  40     | OPERATIONS  | MUMBAI    |
|  50     | MARKETING   | BANGLORE  |
|  60     | PRODUCTION  | AHMEDABAD |
|  70     | EXPORT      | MUMBAI    |
|  80     | IMPORT      | LUCKNOW   |
+---------+-------------+-----------+
```

Now if we write a query as shown below :

This is the first table, hence all the rows from dept table will be returned and unmatched rows will be filled with NULL for the right table's columns.

```
SELECT dept.*, ename, mgr FROM dept LEFT JOIN empl
ON dept.deptno = empl.deptno ;
```

It will produce output that follows :

```
+------+------------+-----------+----------+------+
|deptno| dname      |    loc    |  ename   | mgr  |
+------+------------+-----------+----------+------+
|  10  | ACCOUNTING | NEW DELHI | AMIR     | NULL |
|  10  | ACCOUNTING | NEW DELHI | SHIAVNSH | 8839 |
|  10  | ACCOUNTING | NEW DELHI | MITA     | 8882 |
|  20  | RESEARCH   | CHENNAI   | SMITH    | 8902 |
|  20  | RESEARCH   | CHENNAI   | MAHADEVAN| 8839 |
|  20  | RESEARCH   | CHENNAI   | ANOOP    | 8888 |
|  20  | RESEARCH   | CHENNAI   | SCOTT    | 8566 |
|  20  | RESEARCH   | CHENNAI   | FAKIR    | 8566 |
|  20  | RESEARCH   | CHENNAI   | ANYA     | 8698 |
|  30  | SALES      | KOLKATA   | SETH     | 8698 |
|  30  | SALES      | KOLKATA   | MOMIN    | 8698 |
|  30  | SALES      | KOLKATA   | BINA     | 8839 |
|  30  | SALES      | KOLKATA   | KULDEEP  | 8698 |
|  30  | SALES      | KOLKATA   | JATIN    | 8698 |
|  40  | OPERATIONS | MUMBAI    | NULL     | NULL |
|  50  | MARKETING  | BANGLORE  | NULL     | NULL |
|  60  | PRODUCTION | AHMEDABAD | NULL     | NULL |
|  70  | EXPORT     | MUMBAI    | NULL     | NULL |
|  80  | IMPORT     | LUCKNOW   | NULL     | NULL |
+------+------------+-----------+----------+------+
19 rows in set (0.00 sec)
```

See, how MySQL works on it.

MySQL starts with the left table (***dept*** here). For each row from the table ***dept*** MySQL scans the table *empl*, finds the deptno of the employee and returns employee name(ename) and manager's code(mgr). Then the employee name(ename) and manager's code(mgr) are joined with the matching row from the table ***dept***. For unmatched rows, it returns NULL for ename and mgr fields.

Right Join

RIGHT JOIN works just like the LEFT JOIN but with table order reversed. All rows from the second table are going to be returned whether or not there are matches in the first table. Right join is produced in the same manner as that of left join, just you need to replace the keyword LEFT with RIGHT.

You can use RIGHT JOIN clause of SELECT to produce right join *i.e.,* as per follwoing syntax :

```
SELECT <select-list>
FROM <table1> RIGHT JOIN <table2>
ON <joining-condition> ;
```

That is to produce right join of **members** and **options** table, your query will be :

```
SELECT name, lastname, optonalinfo
FROM members RIGHT JOIN options ON members.id = options.id ;
```

Consider the following tables **Orders** and **Customers**. Examples 7 to 10 are based on these.

ORDERS
OrderId
OrderDate
CustomerId
TotalAmount

ORDERITEMS
OrderId
ProductId
Quantity

CUSTOMERS
CustomerId
Name
City
Country
Phone

PRODUCTS
ProductId
ProductName
SupplierId
UnitPrice

EXAMPLE 7 *Consider the Orders and Customers tables given above. Write an SQL query to list all orders(OrderId) with customer information. (name, city and country).*

SOLUTION

```
SELECT OrderId, TotalAmount, Name, City, Country
FROM Orders, Customers
WHERE Orders.CustomerId = Customers.Customer.Id ;
```


EXAMPLE 8 *Consider the Orders and Customers tables given above. Write an SQL query to list all customer details (name, phone) along with order date.*

SOLUTION

```
SELECT Name, City, Country, Orderdate
FROM Orders, Customers
WHERE Orders.CustomerId = Customers.Customer.Id ;
```

EXAMPLE 9 *Consider the Orders and OrderItems tables given above. Write an SQL query to list order details along with product ids and quantities.*

SOLUTION

```
SELECT Orders.OrderId, OrderDate, ProductId, Quantity
FROM Orders, OrderItems
WHERE Orders.OrderId = OrderItems.OrderId ;
```

EXAMPLE 10 *Consider the Orders, OrderItems and Products tables given above. Write a query to list all orders with product names, quantities and prices, arranged orderId wise.*

SOLUTION

```
SELECT O.OrderINTEGER, O.OrderDate, P.ProductName, I.Quantity, P.UnitPrice
FROM Orders O , Products P, OrderItems I
WHERE O.OrderId = I.OrderId
AND P.ProductId = I.ProductId
ORDER BY O.OrderId ;
```

8.3 Performing SET Operations on Relations

Like joins, SQL set operations (UNION, INTERSECTION, MINUS) also combine the rows coming from two or more relations(tables). But unlike SQL joins, which combine the two relations on columns (*i.e.,* width wise), SQL set operations combine the relations on rows, *i.e.,* rows from one relation are appended beneath other.

In this section, we shall talk about three set operators in SQL – UNION, INTERSECT and MINUS.

NOTE

SQL joins tend to combine columns from two or more tables (*i.e,* width wise) and SQL set operators tend to combine rows from two or more tables (*i.e.,* length wise).

8.3.1 UNION

Multiple queries can be combined into one by forming a union of them. The SQL UNION operator allows manipulation of results returned by two or more queries by combining the results of each query into a single result set.

Consider the data given below that contains rows from relations *A* and *B*.

Relation A

Rollno.	Name	Age
101	Sidharth	19
102	Kushagra	18
103	Usman	18

Relation B

Rollno.	Name	Age
301	Anubha	17
102	Kushagra	18
303	Sba	18

The UNION of these two relations can be created as follows :

```
SELECT * FROM A
UNION
SELECT * FROM B ;
```

The result produced will be as follows :

Rollno.	Name	Age
101	Sidharth	19
102	Kushagra	18
103	Usman	18
301	Anubha	18
303	Sba	18

See the duplicate row (102 Kushagra 18) has automatically been removed. By default, the UNION operator removes duplicate rows from the result. If the ALL option is used, all rows, including duplicates, are included in the results.

The general form of using UNION operator is :

```
SELECT statement
UNION [ALL]
SELECT statement ;
```

NOTE

UNION ALL will retain all the duplicate rows from tables in the final output while UNION will remove the duplicate rows from the final output

8.3.1A Rules of Using UNION Operator

1. Both the SELECT statements must be UNION compatible, that is, the *select-lists* (column-lists) of SELECT statement must have same number of columns having similar data types and order. For instance, the following union query is *invalid* :

   ```
   SELECT ecode, ename FROM branch1
   UNION
   SELECT ename, ecode FROM branch2 ;
   ```

   It is because the order of columns is different. Also, the following union is invalid because the SELECT statements do not contain compatible *select-lists*.

   ```
   SELECT ecode, ename, sex FROM branch1
   UNION
   SELECT ecode, ename, sex, grade FROM branch2 ;
   ```

2. The ORDER BY clause can occur only at the end of the UNION statement. It can't be used within the individual queries that make the UNION statement.
3. The GROUP BY and HAVING clauses are allowed only within individual queries. These clauses cannot be used to affect the final result set.

Let us consider some examples. In this section, we shall use the following tables for examples of SQL set operators.

Table venue1

```
+----------+-----------+
| Match_no | city      |
+----------+-----------+
| 1        | DELHI     |
| 2        | MOHALI    |
| 3        | KOCHI     |
| 4        | MUMBAI    |
| 5        | BANGLURU  |
| 6        | CHENNAI   |
+----------+-----------+
6 rows in set (0.05 sec)
```

Table venue2

```
+----------+-----------+
| Match_no | city      |
+----------+-----------+
| 1        | MOHALI    |
| 2        | BANGLURU  |
| 3        | MUMBAI    |
| 4        | BANGLURU  |
| 5        | MUMBAI    |
| 6        | DELHI     |
| 7        | BANGLURU  |
+----------+-----------+
7 rows in set (0.05 sec)
```

EXAMPLE 11 *Write an SQL query to union cities from tables venue1 and venue2.*

SOLUTION

```
SELECT city FROM venue1
UNION
SELECT city FROM venue2;
```

```
+-----------+
| city      |
+-----------+
| DELHI     |
| MOHALI    |
| KOCHI     |
| MUMBAI    |
| BANGLURU  |
| CHENNAI   |
+-----------+
6 rows in set (0.05 sec)
```

In the above query, firstly table ***Venue1*** is given and then the table ***Venue2***. What if we fetch records from ***Venue2*** first and then from ***Venue1*** ? Let us see ourselves.

EXAMPLE 12 *Write an SQL query to union cities from tables **venue1** and **venue2***

SOLUTION

```
SELECT city FROM venue2
UNION
SELECT city FROM venue1;
```

```
+-----------+
| city      |
+-----------+
| MOHALI    |
| BANGLURU  |
| MUMBAI    |
| DELHI     |
| KOCHI     |
| CHENNAI   |
+-----------+
6 rows in set (0.05 sec)
```

MySQL puts the rows from first query and then from the second query in the UNION result.

But many other Databases like Oracle return sorted result with UNION.

8.3.1B UNION vs. UNION ALL

Both UNION and UNION ALL give the combine rows of two tables, but UNION removes the duplicate entries while UNION ALL retains the duplicate entries (*see* Fig. 8.1 below).

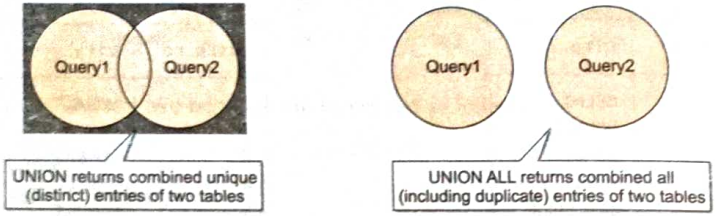


Figure 8.1 UNION vs. UNION ALL operations

Following example illustrates it. Compare its output with the output of example 11.

EXAMPLE 13 *Write an SQL query to union cities from tables venue1 and venue2 containing all the rows.*

SOLUTION

```
SELECT city FROM venue1
UNION ALL
SELECT city FROM venue2 ;
```

```
+----------------+
| city           |
+----------------+
| DELHI          |
| MOHALI         |
| KOCHI          |
| MUMBAI         |
| BANGLURU       |
| CHENNAI        |
| MOHALI         |
| BANGLURU       |
| MUMBAI         |
| BANGLURU       |
| MUMBAI         |
| DELHI          |
| BANGLURU       |
+----------------+
13 rows in set (0.01 sec)
```

See, example 11 returned 6 rows (only unique cities), but this query returned 13 rows as it keeps all the duplicate entries because of UNION ALL.

8.3.2 MINUS Operator

The MINUS operator is used to get the unique rows from the ***table1***, *i.e.* the rows which are in ***table1*** but not in ***table2***. ***MINUS* is also known as *EXCEPT***. (*see* figure below)

With MINUS operator, you need to be careful with the order of SELECT statements as the output will have only those rows that are in the first table and not in the second. In other words, ***table1* MINUS *table2*** is not the same as ***table2*** MINUS ***table1***.

The MINUS operator is used as :

```
SELECT * FROM table1
MINUS
SELECT * FROM table2;
```

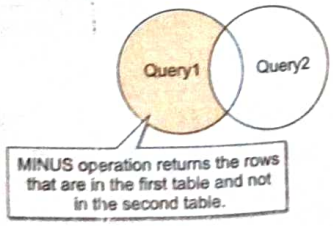


Figure 8.2 MINUS operation

MySQL does not support MINUS operator directly but you can simulate MINUS operator by using LEFT JOIN as per the following syntax :

```
SELECT <list> FROM <table1>
LEFT JOIN <table2>
ON <table1>.<common column> = <table2>.<common column>
WHERE <table2>.<common column> IS NULL ;
```

NOTE

For MINUS also, the two SELECT queries should be compatible just as in UNION.

Suppose if you have following tables :

Vendor

VendorID	Name	Address	CreditLimit
V011	ABC Com.	Mumbai	₹26000
V020	K Creations	Delhi	₹50000

Customer

CustID	Name	Address
C035	K Creations	Delhi
C041	XYZ P Ltd.	Kolkata

Then the following query

```
SELECT Name FROM Vendor
MINUS
SELECT Name FROM Customer ;
```

will give output (*i.e.*, for *Vendor* MINUS *Customer*) as :

ABC Com.

as this is the name which is in **table1** (Vendor) but not in **table2** (Customer).

But if we swap the tables so that ***Customer*** becomes table1 and ***Vendor*** become table2, *i.e.*, for the query

```
SELECT Name FROM Customer
MINUS
SELECT Name FROM Vendor;
```

The output will be :

as this is the name which is in **table1**(Customer) but not in **table2** (Vendor) .

Consider another example using *venue tables* shown earlier.

EXAMPLE 14 *Write an SQL query to get cities only in table **venue1** and not in table venue2.*

SOLUTION

```
SELECT v1.city FROM venue1 v1
LEFT JOIN venue2 v2
ON v1.city = v2.city
WHERE v2.city IS NULL ;
```

```
+----------+
| city     |
+----------+
| KOCHI    |
| CHENNAI  |
+----------+
2 rows in set (0.00 sec)
```

8.3.3 INTERSECT Operator

In SQL, you can use an intersect operation (INTERSECT) to return rows that are common between two tables. In other words, INTERSECT operation returns common rows from both the left and right tables/queries (*see* figure below). This is useful when you want to find results that are in common between two queries.

Let us consider the same **Vendor** and **Customer** tables given in the previous sub-section. If you want to find out about ***vendors*** that are also ***customers***, you can use INTERSECT operator as per following syntax :

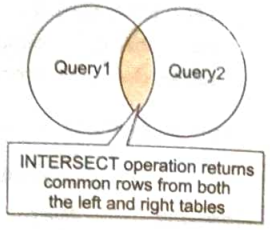


Figure 8.3 INTERSECT operation.

```
<SELECT query1>
INTERSECT
<SELECT query2>
```

e.g.,

```
SELECT Name FROM Vendor
INTERSECT
SELECT Name FROM Customer ;
```

The above query will give result as

```
K Creations
```

You can also use an INNER JOIN to answer the same question.

```
SELECT Distinct V.Name
FROM Vendor V
INNER JOIN Customer C
ON V.Name = C.Name
```

NOTE

For INTERSECT also, the two SELECT queries should be compatible just as in UNION.

The above query returns the same results as that of earlier one. Please note here that MySQL supports no INTERSECT thus you have to stick to the second way of doing INTERSECT operation (INNER JOIN as described above) if working in MySQL.

Check Point 8.1

1. What is unrestricted join ?
2. What is join ?
3. How is equi join different from non-equi join ?
4. What set of operations can you perform through SQL ?
5. How is UNION different from UNION ALL ?
6. How is INTERSECT different from MINUS operation ?

EXAMPLE 15 *Write an SQL query to get cities common to tables venue1 and venue2.*

SOLUTION

```
SELECT DISTINCT v1.city
FROM Venue1 v1
INNER JOIN Venue2 v2
ON v1.city = v2.city ;
```

```
+----------+
| city     |
+----------+
| MOHALI   |
| BANGLURU |
| MUMBAI   |
| DELHI    |
+----------+
4 rows in set (0.00 sec)
```


Rules applicable to all set operators (UNION, INTERSECT, MINUS)

All the three set operators covered above must follow these rules :

- column count must be the same in the select lists.
- data types of retrieved columns should match or at least should be implicitly convertible by database.
- the order of columns in all select lists should be compatible.

LET US REVISE

- *A join is a query that combines rows from two or more tables.*
- *Columns having same name in more than one tables being joined, must have qualified names i.e., they must be specified along with their table name or table alias.*
- *A table alias is a temporary label given along with table name in FROM clause.*
- *Equi-joins are joins based on equality conditions.*
- *Non-equi joins are those where columns being joined are compared for non-equality.*
- *When one of the two identical columns of joined tables is removed in join query, this is called natural join.*
- *The UNION operator returns all unique rows from the SELECT queries.*
- *The MINUS operator returns the rows that are only in the first result set but not in the second.*
- *The INTERSECT operator returns all rows that are in both result sets.*

Objective Type Questions OTQs

Multiple Choice Questions

1. A ____ is a query that retrieves rows from more than one table or view :
   (a) Start (b) End (c) Join (d) All of these
2. A condition given in a Join query is referred to as ____
   (a) Join in SQL (b) Join condition
   (c) Join in SQL & Condition (d) None of these
3. Which of the following is not a join type ?
   (a) Empty join (b) Natural join
   (c) Equi Join (d) Right join
4. Which product is returned in a join query having no join condition ?
   (a) Equijoin (b) Cartesian
   (c) Both Equijoins and Cartesian (d) Natural
5. The Cartesian product is also called ____ join.
   (a) Equijoin (b) Natural
   (c) Unrestricted Join (d) Restricted Join
6. In which type of join, the join condition contains an equality operator ?
   (a) Equijoin (b) Natural (c) Left Join d) Right Join

7. In which type of join, duplicate columns are there ?
   (*a*) Equijoin (*b*) Natural (*c*) Left Join (*d*) Right Join
8. In which type of join, no duplicate columns are there ?
   (*a*) Equijoin (*b*) Natural (*c*) Left Join (*d*) Right Join
9. With SELECT statement used for joins, the USING subclause produces _____ join.
   (*a*) Equijoin (*b*) Natural (*c*) Left Join (*d*) Right Join
10. With SELECT statement used for joins, the ON subclause produces _____ join.
    (*a*) Equijoin (*b*) Natural (*c*) Left Join (*d*) Right Join
11. What is the correct statement for describing the UNION operation ?
    (*a*) It combines the rows of any two different queries
    (*b*) It combines the unique rows of two different queries which have the same set of attributes in the select clause
    (*c*) It combines the rows of two different queries which have the same condition in the where clause
    (*d*) It gives the Cartesian product of the results of any two queries
12. What is the correct statement for describing the INTERSECT operation?
    (*a*) It returns the common values from the results of any two different queries
    (*b*) It returns the common rows of two different queries which have the same set of attributes in the select clause
    (*c*) It returns the common rows of two different queries which have the same condition in the where clause
    (*d*) None of these
13. What is the correct statement for describing the EXCEPT operation ?
    (*a*) It excludes all the rows present in both the queries
    (*b*) It includes the rows of the second query but excludes the results of the first query
    (*c*) It includes the rows of the first query but excludes the results of the second query
    (*d*) It includes all the rows of both queries but removes duplicates
14. What is the other name of MINUS operator?
    (*a*) UNION (*b*) UNION ALL
    (*c*) EXCEPT (*d*) INTERSECT
15. Examine the structure of the EMPL and DEPT tables:

EMPL	Column name	Data type	Remarks
	EMPLOYEE_ID	NUMBER	NOT NULL, Primary Key
	EMP_NAME	VARCHAR(30)	
	JOB_ID	VARCHAR(20)	
	SALARY	NUMBER	
	MGR_ID	NUMBER	References EMPLOYEE_ID COLUMN
	DEPARTMENT ID	NUMBER	Foreign key to DEPARTMENT ID column of the DEPT table

DEPT	Column name	Data type	Remarks
	DEPARTMENT_ID	NUMBER	NOT NULL, Primary Key
	DEPARTMENT_NAME	VARCHAR(30)	
	MGR_ID	NUMBER	References MGR_ID column of the EMPL table

Evaluate this SQL statement :

```
SELECT employee_id, e.department_id, department_name, salary
FROM EMPL e, DEPT d
WHERE e.department_id = d.department_id;
```

Which SQL statement is equivalent to the above SQL statement ?

(*a*)
```
SELECT employee_id, department_id, department_name, salary
FROM EMPL
WHERE department_id MATCHES department_id of DEPT;
```

(*b*)
```
SELECT employee_id, department_id, department_name,salary
FROM EMPL
NATURAL JOIN DEPT;
```

(*c*)
```
SELECT employee_id, d.department_id, department_name,salary
FROM EMPL e
JOIN DEPT d
ON e.department_id = d.department_id;
```

(*d*)
```
SELECT employee_id, department_id, department_name,Salary
FROM EMPL
JOIN DEPT
USING (e.department_id, d.department_id);
```

Fill in the Blanks

1. An SQL _____ clause combines records from two or more tables in a database.
2. An _____ is a specific type of join that uses only equality comparisons in the join-condition.
3. _____ join selects all data starting from the left table and matching rows in the right table.
4. _____ join is a reversed version of the left join.
5. _____ join produces a data set that includes only those rows from the left table which have matching rows from the right table .
6. An _____ query returns the intersection of 2 or more tables.
7. _____ _____ returns all rows from both the SELECT queries and it does not remove duplicate rows between the various SELECT statements.
8. _____ returns all rows from both the SELECT queries after removing duplicate rows between the two SELECT statements.
9. The _____ operation combines results of two SELECT statements and returns only those rows in the final result, which belong to the first set of the result.
10. The INTERSECT operation can be simulated in MySQL using _____ JOIN.

True/False Questions

1. Non-equi join is the name of Natural join.
2. Natural join contains the duplicate columns.
3. Equi join contains the duplicate columns.

4. Inner Join combines only the matching rows from the two tables.
5. A join condition can only use equality for comparison.
6. Left join has all the rows from the Left table.
7. Outer join has all the rows from both the tables.
8. MINUS operator is also called EXCEPT operator.
9. UNION returns unique rows.
10. UNION and UNION ALL produce similar results.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *Define the following : (i) Cross Join (ii) Left join (iii) Natural join (iv) Equi-join*

Solution.

Cross Join A form of join that produces all possible combinations of row from two tables.

Left Join A form of join where all the rows (even non-matching ones) from the first table are returned in the result.

Natural Join A join where identical column from both tables is shown just one in the result.

Equi-join A join based on equality of a common field of two tables and in which the identical column is shown twice.

2. *What will be the join of following two relations*

R1 (*A, B, C*) :

A	B	C
I	1	Y
J	3	Y
K	3	Z
L	2	Y
M	1	Z
N	7	Y

R2 (*B, D, E*)

B	D	E
1	p	S
2	q	T
5	p	U
7	q	T

Solution.

Equi join

A	B	C	B	D	E
I	1	Y	1	p	S
L	2	Y	2	q	T
M	1	Z	1	p	S
N	7	Y	7	q	T

Natural Join

A	B	C	D	E
I	1	Y	p	S
L	2	Y	q	T
M	1	Z	p	S
N	7	Y	q	T

3. *Given two tables.*

Employee

```
employee_id
last_name
first_name
middle_name
marital_status
```

PayTable

```
employee_id
salary
department
supervisor
```

Find out the salary taken by all SMITH i.e., all those who have their last name as SMITH.

Solution.

```
SELECT e.employee_id, e.last_name, ep.salary
FROM employee_table, employee_pay_tbl ep
WHERE e.employee_id = ep.employee_id
AND e.last_name = 'SMITH' ;
```

4. *Write query to produce all the details only for female pets records of pet and event tables (joined)*

Solution.

```
SELECT *
FROM pet
JOIN event
USING (name)
WHERE sex = 'f' ;
```

5. *A table FLIGHT has 4 rows and 2 columns and another table AIR HOSTESS has 3 rows and 4 columns. How many rows and columns will be there if we obtain the Cartesian product of these two tables ?* [CBSE D 12]

Solution. 12 rows, 6 columns.

6. *Table Employee has 4 records and Table Dept has 3 records in it. Mr. Jain wants to display all information stored in both of these related tables. He forgot to specify equi-join condition in the query. How many rows will get displayed on execution of this query ?* [CBSE D 11 ; OD 11]

Solution. 12 rows (4×3)

Given following tables : *Customers, Parts, Orders*. Perform questions (7-10) based on these problems.

TABLE : CUSTOMERS

Name	Address	State	Zip	Phone	Remarks
True wheel	550 Husker	NE	58702	555-4545	None
Bike Spec	CPT Shrive	LA	45678	555-1234	None
Le Shoppe	Hometown	KS	54678	555-1278	None
AAA Bike	10 Oldtown	NE	56784	555-3421	John-Mgr
Jacks Bike	24 Eglin	FL	34567	555-2314	None

TABLE : PARTS

Partnum	Description	Price
54	Pedals	54.25
42	Seats	24.50
46	Tires	15.25
23	Mountain Bike	350.45
76	Road Bike	530.00
10	Tandem	1200.00

TABLE : ORDERS

Orderedon	Name	Partnum	Quantity	Remarks
15-MAY-1996	True wheel	23	6	Paid
19-MAY-1996	True wheel	76	3	Paid
2-SEP-1996	True wheel	10	1	Paid
30-JUNE-1996	True wheel	42	8	Paid
30-JUNE-1996	Bike Spec	54	10	Paid
30-MAY-1996	Bike Spec	10	2	Paid
30-MAY-1996	Bike Spec	23	8	Paid
17-JAN-1996	Bike Spec	76	11	Paid
17-JAN-1996	Le Shoppe	76	5	Paid
1-JUNE-1996	Le Shoppe	10	3	Paid
1-JUNE-1996	AAA Bike	10	1	Paid
1-JUL-1996	AAA Bike	76	4	Paid
1-JUL-1996	AAA Bike	46	14	Paid
11-JUL-1996	Jacks Bike	76	14	Paid

7. *Find out cartesian product of Orders and Parts i.e., Order × Parts.*

Solution.

```
SELECT orderedon, name, ord.partnum, pr.partnum, description
FROM orders ord, parts pr ;
```

8. *What will be the equi join of tables orders, parts ?*

Solution.

```
SELECT ord.orderedon, ord.name, ord.partnum, pr.partnum, pr.description
FROM orders ord, parts pr
WHERE ord.partnum = pr.partnum ;
```

9. *Determine the total order amount for part 'Road Bike'.*

Solution.

```
SELECT SUM (ord.quantity * pr.price) 'Total Amount'
FROM orders ord, parts pr
WHERE ord.partnum = pr.partnum
AND pr.description = 'Road Bike' ;
```


10. *Find out order details of parts having description like 'Road ..... '*

Solution.

```
SELECT * FROM orders
WHERE partnum = (SELECT partnum
                 FROM parts
                 WHERE description LIKE 'Road%') ;
```

11. *In a Database Company, there are two tables given below :*

Table : SALES

SALESMANID	NAME	SALES	LOCATIONID
S1	ANITA SINGH ARORA	250000	102
S2	Y.P. SINGH	1300000	101
S3	TINA JAISWAL	1400000	103
S4	GURDEEP SINGH	1250000	102
S5	SIMI FAIZAL	1450000	103

Table : LOCATION

LOCATIONID	LOCATIONNAME
101	Delhi
102	Mumbai
103	Kolkata
104	Chennai

Write SQL queries for the following :

(i) *To display SalesmanID, names of salesmen, LocationID with corresponding location names.*

(ii) *To display names of salesmen, sales and corresponding location names who have achieved Sales more than 1300000.*

(iii) *To display names of those salesmen who have 'SINGH' in their names.*

(iv) *Identify Primary key in the table SALES. Give reason for your choice.*

(v) *Write SQL command to change the LocationID to 104 of the Salesman with ID as S3 in the table 'SALES'.* **[CBSE OD 15]**

Solution.

(i)
```
SELECT SALESMANID, NAME, LOCATIONID, LOCATIONNAME
FROM SALES S, LOCATION L
WHERE S.LOCATIONID = L.LOCATIONID ;
```

(ii)
```
SELECT NAME, SALES, LOCATIONNAME
FROM SALES S, LOCATION L
WHERE S.LOCATIONID = L.LOCATIONID AND SALES > 1300000 ;
```

(iii)
```
SELECT NAME
FROM SALES
WHERE NAME LIKE '%SINGH%' ;
```

(iv) Primary Key : SALESMANID

Reason. It uuniquely identifies all ROWS in the table and does not contain empty/zero or null values.

(v)
```
UPDATE SALES
SET LOCATIONID = 104
WHERE SALESMANID = 'S3' ;
```

12. *In a Database Multiplexes, there are two tables with the following data. Write MySQL queries for (i) and (iii), which are based on* **TicketDetails** *and* **AgentDetails** *:* **[CBSE D 2014]**

Table : TicketDetails

Tcode	NAME	Tickets	A_code
S001	Meena	7	A01
S002	Vani	5	A02
S003	Meena	9	A01
S004	Karish	2	A03
S005	Suraj	1	A02

Table : AgentDetails

ACode	AName
A01	Mr. Robin
A02	Mr. Ayush
A03	Mr. Trilok
A04	Mr. John

(i) *To display Tcode, Name and Aname of all the records where the number of tickets sold is more than 5.*
(ii) *To display total number of tickets booked by agent "Mr. Ayush".*
(iii) *To display Acode, Aname and corresponding Tcode where Aname ends with "k".*

Solution.

(i)
```
Select Tcode, Name, AName
From TicketDetails TD, AgentDetails AD
Where TD.A_Code = AD.Acode
AND Tickets > 5 ;
```

(ii)
```
Select Count(*)
From TicketDetails TD, AgentDetails AD
Where TD.A_Code = AD.Acode
AND AName = "Mr. Ayush" ;
```

(iii)
```
Select Acode, AName, Tcode
From TicketDetails TD, AgentDetails AD
Where TD.A_Code = AD.Acode
AND AName Like '%K' ;
```

13. *In a Database - SAMS and VENDOR are two tables with the following information. Write MySQL queries for (i) to (iii), based on tables SAMS and VENDOR :* **[CBSE OD 2014]**

Table : SAMS

ICode	IName	Price	Colour	VCode
S001	Refrigerator	20000	Blue	P01
S002	Mobile Phone	45000	Black	P02
S003	LCD	60000	Silver	P03
S004	Washing Machine	12500	Smoke	P01
S005	Air conditioner	16000	White	P03

Table : VENDOR

VCode	VNAME
P01	Satish
P02	Manoj
P03	Subodh
P04	Jacob

(i) *To display ICode, IName and VName of all the vendors, who manufacture "Refrigerator".*
(ii) *To display IName, ICode, VName and Price of all the products whose price is more than 20000.*
(iii) *To display vendor names and names of all items manufactured by vendor whose code is "P03".*

Solution.

(i)
```
SELECT ICode, IName, VName
FROM SAMS S, VENDOR V
WHERE S.VCode = V.Vcode
and IName = "Refrigerator" ;
```

(ii)
```
SELECT IName, ICode, VName, Price
FROM SAMS S, VENDOR V
WHERE S.VCode = V.Vcode
and Price > 20000 ;
```

(iii)
```
SELECT VName, IName FROM SAMS S, VENDOR V
WHERE S.Vcode = V.Vcode
and V.Vcode = "P03" ;
```

14. *In a Database there are two tables :* **[CBSE D 12]**

Table : ITEM

Item_Code	Item_Name	Price
111	Refrigerator	90,000
222	Television	75,000
333	Computer	42,000
444	Washing Machine	27,000

Table : BRAND

Item_Code	Brand_Name
111	LG
222	Sony
333	HCL
444	IFB

Write MySQL queries for the following :

(i) *To display Item_Code, Item_Name and corresponding Brand_Name of those Items, whose Price is between 20000 and 40000 (both values inclusive).*
(ii) *To display Item_Code, Price and Brand_Name of the item, which has Item_Name as "Computer".*

Solution.

(i)
```
SELECT I.Item_Code, Item_Name, Brand_Name
FROM Item I, Brand B
WHERE I.Item_Code = B.Item_Code
AND Price BETWEEN 20000 AND 40000 ;
```

(ii)
```
SELECT I.Item_Code, Price, BrandName
FROM Item I, Brand B
WHERE I.Item_Code = B.Item_Code
AND Item_Name = "Computer" ;
```

15. *Examine the structure of the EMPLOYEES, DEPARTMENTS, and LOCATIONS tables.*

EMPLOYEES

EMPLOYEE_ID	NUMBER	NOT NULL, Primary Key
EMP_NAME	VARCHAR(30)	
JOB_ID	VARCHAR(20)	
SALARY	NUMBER	
MGR_ID	NUMBER	References EMPLOYEE_ID column
DEPARTMENT_ID	NUMBER	Foreign key to DEPARTMENT_ID column of the DEPARTMENTS table

DEPARTMENTS

DEPARTMENT_ID	NUMBER	NOT NULL, Primary Key
DEPARTMENT_NAME	VARCHAR(30)	
MGR_ID	NUMBER	References MGR_ID column of the EMPLOYEES table
LOCATION_ID	NUMBER	Foreign key to LOCATION_ID column of the LOCATIONS table LOCATIONS
LOCATION_ID	NUMBER	NOT NULL, Primary Key,
CITY	VARCHAR(30)	

Which two SQL statements produce the name, department name, and the city of all the employees who earn more than 10000? (Choose two)

A.
```
SELECT emp_name, department_name, city
FROM employees e
JOIN departments d
USING (department_id)
JOIN locations l
USING (location_id)
WHERE salary > 10000;
```

B.
```
SELECT emp_name, department_name, city
FROM employees e, departments d, locations l
JOIN ON (e.department_id = d.department id)
AND (d.location_id = l.location_id)
AND salary > 10000;
```

C.
```
SELECT emp_name, department_name, city
FROM employees e, departments d, locations l
WHERE salary > 10000;
```

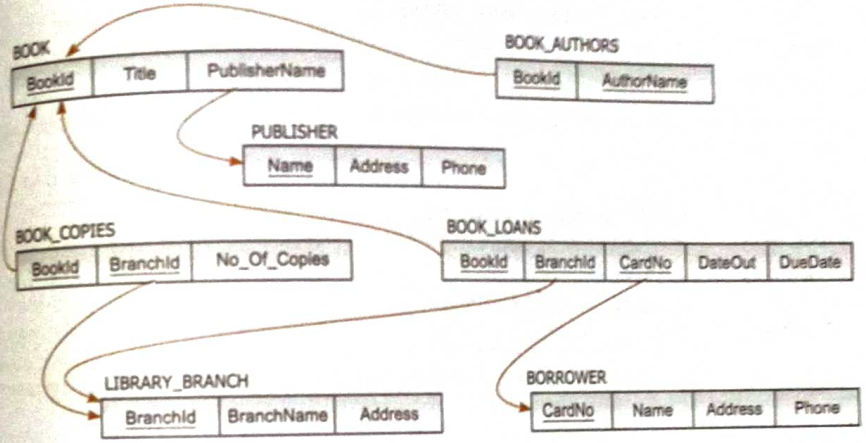
D.
```
SELECT emp_name, department_name, city
FROM employees e, departments d, locations l
WHERE e.department_id = d.department_id
AND d.location_id = l.location_id
AND salary > 10000;
```

E.
```
SELECT emp_name, department_name, city
FROM employees e
NATURAL JOIN departments, locations
WHERE salary > 10000;
```

Solution. B, D

Retrieve the required information using SQL language.

Given a database schema for a library management system as the following picture.



16. *How many copies of the book titled The Lost Tribe are owned by the library branch whose name is "Sharpstown" ?*

Solution. (*i*)
```
SELECT   bc.No_Of_Copies
FROM     BOOK b, BOOK_COPIES bc, LIBRARY_BRANCH bl
WHERE    b.BookId = bc.BookId AND
         bc.BranchId = bl.BranchId AND
         Title = 'The Lost Tribe' AND BranchName = 'Sharpstown' ;
```

(*ii*)
```
SELECT   No_Of_Copies
FROM     ((BOOK NATURAL JOIN BOOK_COPIES ) NATURAL JOIN LIBRARY_BRANCH)
WHERE    Title = 'The Lost Tribe' AND BranchName = 'Sharpstown' ;
```

17. *How many copies of the book titled The Lost Tribe are owned by each library branch ?*

Solution.
```
SELECT   BranchName, No_Of_Copies
FROM     ((BOOK NATURAL JOIN BOOK_COPIES ) NATURAL JOIN LIBRARY_BRANCH)
WHERE    Title = 'The Lost Tribe' ;
```

18. *For each book that is loaned out from the "Sharpstown" branch and whose DueDate is today, retrieve the book title, the borrower's name, and the borrower's address.*

Solution.
```
SELECT   B.Title, R.Name, R.Address
FROM     BOOK B, BORROWER R, BOOK_LOANS BL, LIBRARY_BRANCH LB
WHERE    LB.BranchName = 'Sharpstown' AND LB.BranchId = BL.BranchId AND
         BL.DueDate = 'today' AND BL.CardNo = R.CardNo AND BL.BookId = B.BookId ;
```

19. *For each library branch, retrieve the branch name and the total number of books loaned out from that branch.*

Solution.

```
SELECT      L.BranchName, COUNT(*)
FROM        LIBRARY_BRANCH L, BOOK_LOANS BL
WHERE       BL.BranchId = L.BranchId
GROUP BY    L.BranchName ;
```

20. *Retrieve the names, addresses, and number of books checked out for all borrowers who have more than five books checked out.*

Solution.

```
SELECT      B.Name, B.Address, COUNT(*)
FROM        BORROWER B, BOOK_LOANS L
WHERE       B.CardNo = L.CardNo
GROUP BY    B.CardNo, B.Name, B.Address
HAVING      COUNT(*) > 5 ;
```

21. *For each book authored (or co-authored) by "Stephen King", retrieve the title and the number of copies owned by the library branch whose name is "Central"*

Solution.

```
SELECT  Title, No_Of_Copies
FROM    (((BOOK_AUTHORS NATURAL JOIN BOOK) NATURAL JOIN BOOK_COPIES) NATURAL JOIN
        LIBRARY_BRANCH) WHERE Author_Name = 'Stephen King' AND BranchName = 'Central';
```

22. *Following SQL query is giving ERROR. State the reason.*

```
SELECT * FROM venue1
UNION
SELECT city FROM venue2;
```

Solution. The reason for error is that the two SELECT queries in the given SQL statement are not union-compatible. For two queries to be union-compatible, they both must have same number and type of columns in their select lists.

23. *Consider the following query. Will it produce error? Why / Why not?*

```
SELECT department_id, department_name
FROM  departments
WHERE department_id <= 30
UNION
SELECT department_id, department_name
FROM  departments
WHERE department_id >= 20 ;
```

Solution. The above query will produce no error as their SELECT lists are fully compatible and it is possible to UNION rows from the same table.

24. *Write an SQL query that lists the common cities in two match-schedule tables namely* ***Schedule1*** *and* ***Schedule2****.*

Solution.

```
SELECT city FROM schedule1
INTERSECT
SELECT city FROM schedule2 ;
```


25. *Consider the following query ? Which SQL set operator is being simulated by it ?*

```
SELECT id FROM t1
LEFT JOIN t2
USING (id)
WHERE t2.id IS NULL ;
```

Solution. The given query is implementing MINUS operator.

GLOSSARY

Join	A query that combines rows from one or more tables.
Table Alias	Temporary label given along with table in FROM clause.
Equi Join	Join in which tables are compared for equality.
Non-Equi Join	Join in which tables are compared for non-equality.
Natural Join	Join that consists of only one of the identical columns, coming from joined tables.

Assignment

Type A : Short Answer Questions/Conceptual Questions

1. What is cartesian product ? How is it related to join ?
2. What is a join ? How is natural join different from an equi-join ?
3. What is a table alias ? What is the purpose of table alias ?
4. Define an equi-join. What is non-equi-join ?
5. What is join ? How many different types of joins can you create in MySQL ?
6. There are multiple ways to create cartesian product of two tables in MySQL. Describe them.
7. How is a left join different from a natural join ?
8. How is a cross join different from natural join ?
9. Can you join two tables without using the keyword JOIN ?
10. What is the difference between ON and USING join-clauses ?
11. A table STUDENT has 4 rows and 2 columns and another table TEACHER has 3 rows and 4 columns. How many rows and columns will be there if we obtain the Cartesian product of these two tables ? [CBSE OD 12]
12. What does UNION do ?
13. Are UNION and UNION ALL the same ? Why ?
14. What does INTERSECT do ?
15. What does MINUS operation do ?

Type B : Application Based Questions

1. *In a Database, there are two tables given below :*

Table : EMPLOYEE

EMPLOYEEID	NAME	SALES	JOBID
E1	SUMIT SINHA	1100000	102
E2	VIJAY SINGH TOMAR	1300000	101
E3	AJAY RAJPAL	1400000	103
E4	MOHIT RAMNANI	1250000	102
E5	SHAILJA SINGH	1450000	103

Table : JOB

JOBID	JOBTITLE	SALARY
101	President	200000
102	Vice President	125000
103	Administration Assistant	80000
104	Accounting Manager	70000
105	Accountant	65000
106	Sales Manager	80000

Write SQL Queries for the following :

(i) To display employee ids, names of employees, job ids with corresponding job titles.

(ii) To display names of employees, sales and corresponding job titles who have achieved sales more than 1300000.

(iii) To display names and corresponding job titles of those employees who have 'SINGH' (anywhere) in their names.

(iv) Identify foreign key in the table EMPLOYEE.

(v) Write SQL command to change the JOBID to 104 of the EMPLOYEE with ID as E4 in the table 'EMPLOYEE'. [CBSE D 15]

2. Show the average salary for all departments with more than 3 people for a job.
3. Display only the jobs with maximum salary greater than or equal to 3000.

Consider tables EMPL, Dept, SalaryGrade. The Empl table has already been listed in earlier chapters. Schemas of tables SalaryGrade and Dept are being shown below :

```
SALARYGRADE (Lowsal, Highsal, Grade)
Dept (Deptno, DeptName, Location)
```

Answer Questions 4 and 5 on the basis of the above given tables :

4. List the department names and the number of their employees.
5. List the employee names and the name of their departments.
6. What is join ? How many different types of joins can you create in MySQL.
7. In a Database Karnataka_Sangam there are two tables with the instances given below : [CBSE D 2013]

Table : STUDENTS

ADMNO	NAME	CLASS	SEC	RN	ADDRESS	PHONE
1211	Meena	12	D	4	A-26	3245678
1212	Vani	10	D	1	B-25	5456789
1213	Meena	12	A	1		
1214	Karish	10	B	3	AB-234	4567890
1215	Suraj	11	C	2	ZW12	4345677

Table : SPORTS

ADMNO	GAME	COACHNAME	GRADE
1215	Cricket	Mr. Ravi	A
1213	Vollyball	Ms. Chadha	B
1211	Vollyball	Mr. Govardhan	A
1212	Basket Ball	Mr. Tewani	B

Write SQL queries for the following :

(i) To count how many addresses are not having NULL values in the address column of STUDENTS table.

(ii) To display Name, Class from STUDENTS table and the corresponding Grade from SPORTS table.

(iii) To display Name of the student and their corresponding Coachnames from STUDENTS and SPORTS tables.

8. In a Database BANK there are two tables with a sample data given below : [CBSE OD 2013]

Table : EMPLOYEE

ENO	ENAME	SALARY	ZONE	AGE	GRADE	DEPT
1.	Mona	70000	East	40	A	10
2.	Muktar	71000	West	45	B	20
3.	Nalini	60000	East	26	A	10
4.	Sanaj	65000	South	36	A	20
5.	Surya	58000	North	30	B	30

Table : DEPARTMENT

DEPT	DNAME	HOD
10	Computers	1
20	Economics	2
30	English	5

Note.

- ENAME refers to Employee Name
- DNAME refers to Department Name
- DEPT refers to Department Code
- HOD refers to Employee number (ENO) of the Head of the Dept.

Write SQL queries for the following :

(i) To display ENO, ENAME, SALARY and corresponding DNAME of all the employees whose age is between 25 and 35 (both values inclusive).

(ii) To display DNAME and corresponding ENAME from the tables DEPART- MENT and EMPLOYEE. **Hint.** HOD of the DEPARTMENT table should be matched with ENO of the EMPLOYEE table for getting the desired result.

(iii) To display ENAME, SALARY, ZONE and INCOME TAX (**Note.** Income Tax to be calculated as 30% of salary) of all the employees with appropriate column headings.

9. There are multiple ways to create cartesian product of two tables in MySQL. Describe them.
10. How is a left join different from a natural join ? Give example.
11. How is a cross join different from natural join ? Give example.

12. A table "TRAINS" in a database has degree 3 and cardinality 8. What is the number of rows and columns in it ?

13. In a database there are two tables "Product" and "Client" as shown below : [CBSE D 11]

Table : Product

P_ID	ProductName	Manufacture	Price
P001	Moisturiser	XYZ	40
P002	Sanitizer	LAC	35
P003	Bath Soap	COP	25
P004	Shampoo	TAP	95
P005	Lens Solution	COP	350

Table : Client

C_ID	ClientName	City	P_ID
01	Dreamz Disney	New Delhi	P002
05	Life Line Inc	Mumbai	P005
12	98.4	New Delhi	P001
15	Appolo	Banglore	P003

Write the commands in SQL queries for the following :

(*i*) To display the details of Product whose Price is in the range of 40 and 120 (Both values included).

(*ii*) To display the ClientName, City from table Client and ProductName and Price from table Product, with their corresponding matching P_ID.

(*iii*) To increase the Price of all the Products by 20.

14. In a Database School there are two tables Member and Division as shown below. [CBSE D 11]

Table : Member

EmpId	Name	Pay	Divno
1001	Shankhya	34000	10
1003	Ridhima	32000	50
1002	Sunish	45000	20

Table : Division

Divno	Divname	Location
10	Media	TF02
20	Dance	FF02
30	Production	SF01

(*i*) Identify the foreign key in the table Member.

(*ii*) What output, you will get, when an equi-join query is executed to get the NAME from Member Table and corresponding from Division table ?

Consider following tables to answer questions 15-20 :

`Salesman (salesman_id, name , city, commission ) and`
`Customer (customer_id, cust_name, city, grade , salesman_id )`

15. Write an SQL query to display all salesmen and customer located in Chennai.
16. Write an SQL query to display distinct salesman and their cities.
17. Write an SQL query to list all the salesmen who do not have customers in 'DELHI' and 'JAMMU'.
18. Write an SQL query to list all the salesmen who have customers with grade 200 or higher.
19. Write an SQL query to list all the salesmen who have customers with grade 200 from city 'JAMMU'.
20. Write an SQL query to list all the customers whose salesmen have earned a commission of at least 15%.

Consider the tables Member and Division of question 14 and answer questions 21-25

21. Write a SQL query to display all the divnos from both the tables
22. Write a SQL query to display all the distinct divnos from both the tables
23. Write a SQL query to display all the divnos in Member table but not in the Division table.
24. Write a SQL query to display all the divnos in Division table but not in the Member table.
25. Write a SQL query to display common divnos from both the tables.

9 Introduction to Computer Networks

Outline

9.1 Introduction
9.2 Computer Networks - AN Introduction
9.3 Types of Networks
9.4 Network Topologies
9.5 Network Devices

9.1 Introduction

In the history of mankind, networking in any form has always given better results, better outcomes and better utilization of resources. For instance, you always have benefitted from *human networks e.g.*, you want to buy a certain book, which is not available at the bookshop near you. But your friend, who lives in a different part of the city, says he can buy it for you as this book is available in the bookshop near his house. You happily agree to this and the next day in school, he hands over the book to you and you pay to him the book-price. Both are happy–he is happy to help ; you are happy to get the book and to have a helping friend. :). See, networking is so useful ; in fact, networking is useful in all forms – computer networking is no exception. Connections among humans make human network and connections among computers make **computer networks**.

In this chapter, we shall talk about the basics of computer networks, their structures, types, devices, protocols etc.

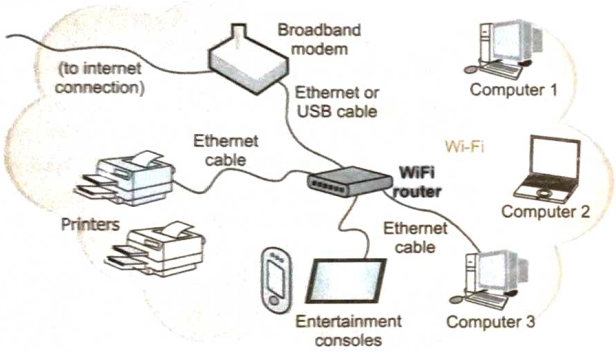
9.2 Computer Networks – AN Introduction

Two or more autonomous[1] **computing devices** connected to one another in order to exchange information or share resources, form a **computer network**. For example, if in your home, you can connect your *smartphone*, your *laptop* with your *smart TV*, *gaming console* and a *printer* simultaneously either using cables or through WiFi, it will be termed as a **Computer Network**.

Figure 9.1 shows a sample computer network.

COMPUTER NETWORK

"A **computer network** is a collection of interconnected autonomous computing devices so as to exchange information or share resources."



Advantages of Networks

★ **Share resources** : such as printers and scanners. This is cheaper than buying equipment for each computer.



★ **Share storage** – being able to access files from any machines on the network can share data.

★ **Can share software** : Software can be installed centrally rather than on each machine. Metering software can then be used to limit the number of copies being run at any one time. This is much cheaper than buying licenses for every machine.

★ **Improve communications** Messages can be sent - *e.g.*, internal email.

Disadvantages of Networks

★ The systems are more sophisticated and **complex** to run. This can add to **costs** and you may need specialist staff to run the network.



★ If networks are badly managed, services can become unusable and productivity falls.

★ If software and files are held centrally, it may be impossible to carry out any work if the central server fails. People become reliant on the communications, if these fail, it can cause havoc.

★ File security is more important especially if connected to WANs *e.g.*, protection from viruses.

Figure 9.1 (*a*) A sample computer network. (*b*) Advantages and disadvantages of computer networks.

Computer networks are very useful in many ways. They facilitate **resource sharing** (resources such as storage, software etc. on the network can be shared), **enhanced communication** (communicating with devices on a network is easier), **cost reduction** (resource sharing cuts on costs) and so forth. Figure 9.1(*b*) shows some advantages and disadvantages of computer networks.

NOTE

Please note that now onwards, we shall use the terms 'computer network' and 'network' interchangeably.

1. It means that no computing device on a network can start, stop or control other device(s).

9.2.1 Components of a Computer Network

Merely by joining two computers with a cable won't form a network. In fact, there are many components that together make a network. We are briefly discussing below, the major components of a computer network.

Major components of a Computer Networks are :

(*a*) **Hosts/Nodes** (such as PC, laptops, smartphones etc.)
(*b*) **Servers**
(*c*) **Client**
(*d*) **Network hardware** (such as *NIC, router, switch, hub* etc.)
(*e*) **Communication channel** (such as cables, radio-links etc.)
(*f*) **Software** (such as protocols, network operating system etc.)
(*g*) **Network services** (such as BNS, File-sharing etc.).

Let us talk about these, briefly.

(*a*) **Host or Nodes.** The term **host** or **node** refers to the computers that are attached to a network and are seeking to share the resources of the network. Of course, if there were no nodes (also called *workstations*), there would be no network at all.

NOTE

A computer becomes a workstation of a network as soon as it is attached to a network.

So your PCs, laptops, smartphones etc. when connect to a network become hosts.

SERVER

"**Server** A computer that facilitates the sharing of data, software, and hardware resources (*e.g.*, printers, modems etc.) on the network, is termed as a Server."

(*b*) **Server.** A **Server** is a very important computer in a network. A **server** is responsible for making the networking tasks happen. In other words, a server facilitates networking tasks like *sharing of data, resource-sharing, communication among hosts* etc.

On small networks, sometimes, all the shareable stuff (like files, data, software etc.) is stored on the server. A network can have more than one server also. Each server has a unique name on the network and all users of network identify the server by its unique name.

On big networks, there can be servers dedicated to specialized tasks *e.g.*, a **file server** only handles files' related requests ; a **printer server** only handles printing requests and so on.

(*c*) **Clients.** Client is a related term. A client computer is a host computer that requests for some services from a server. In other words, a *server computer* serves the requests of client *computers*.

(*d*) **Network Hardware.** Other than hosts and wiring, a network requires specialized hardware to carry out various roles, such as establishing corrections, controlling network traffic etc. There are many different types of hardware that are required in a network. Some examples of network hardware are :

- **NIC (Network Interface Unit).** It is a network card attached to a host so as to establish network connections. Every NIC card has a unique physical address called the MAC address, which is a 6 byte (48 bits) address assigned by the NIC manufacturer. An NIC is also called Network Interface Unit (NIU) or Terminal Access Point (TAP)
- ***Hub, switch, router.*** These are connectivity devices.
- many others.

(*e*) **Communication channel.** Hosts in a network interact with other hosts and server(s) through a ***communication channel*** or ***communication medium***. The communication channels can either be **wired** or **wireless** :

- **Wired Communication channels**. When hosts and server(s) are connected with one another through **guided media** like *network cables*, it is called a **wired communication channel/medium**. Examples of wired communication media are : **twisted-pair cables, coaxial cables, firbre-optic cables**.
- **Wireless Communication channels.** When hosts and server(s) are connected with one another through **unguided media** like ***radio waves***, satellite etc., it is called a wireless communication channel/medium. Examples of wireless communication media are : ***Microwaves, radio waves, satellites, infrared waves, laser*** etc.

(*f*) **Software.** The software layers of a network make networking possible. These comprise of network protocols, network operating system etc.

A **protocol** refers to a pre-decided set of rules using which all parties of a network connect and interact with one another.

A **network operating system** is a specialized operating system that can handle networking tasks.

(*g*) **Network Services.** These refer to the applications that provide different functionalities over a network, such as DNS (Domain Name System), File sharing, VoIP (Voice over IP) and many more.

Armed with this basic knowledge of computer networks, let us further our discussion by talking about types of networks.

9.3 Types of Networks

A computer network means a group of 'networked' computers *i.e.*, computers that are linked by means of a communication system. A network can mean a small group of linked computers to a chain of a few hundred computers of different types (*e.g.*, PCs, minis, mainframes etc.) spread around the world. Thus, networks vary in size, complexity and geographical spread.

Let us discuss types of networks based on these parameters.

9.3.1 Types of Networks based on Geographical Spread

Based on network span or geographical spread, networks can be divided into *two* basic types :

(*i*) LAN (Local Area Network) (*ii*) WAN (Wide Area Network)
(*iii*) PAN (Personal Area Network) (*iv*) MAN (Metropolitan Area Network)

9.3.1A Local Area Network (LAN)

Small computer networks that are confined to a localised area (*e.g.*, an office, a building or a factory) are known as *Local Area Networks* (LANs). The key purpose of a LAN is to serve its users in resource sharing. The hardware as well as software resources are shared through LANs. For instance, LAN users can share data, information, programs, printers, hard-disks, modems etc. Fig. 9.1(*a*) shows a Local Area Network.

In a typical LAN configuration, one computer is designated as the **file server**[2]. It stores all of the software that controls the network, as well as the software that can be shared by the computers attached to the network. Computers connected to the server are called workstations. On most LANs, cables are used to connect the ***network interface cards***[3] in each computer.

Figure 9.1(*a*) that you have seen earlier shows a LAN.

NOTE

Traditionally, LANs are said to have geographical spread of upto 1 km.

9.3.1B Wide Area Network (WAN)

The networks spread across countries or on a very big geographical area are known as WANs. A *Wide Area Network* (WAN) is a group of computers that are separated by large distances and tied together. It can even be a group of LANs that are spread across several locations and connected together to look like one big LAN. The WANs link computers to facilitate fast and efficient exchange of information at lesser costs and higher speeds.

Computers connected to a wide-area network are often connected through public networks, such as the telephone system. Sometimes they can be connected through *leased lines*[4] or satellites. The largest WAN in existence is the *Internet*.

Figure 9.2 shows you a LAN and a WAN.

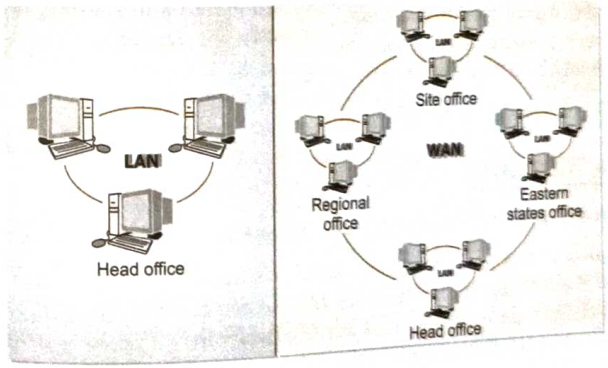


Figure 9.2 LAN *vs.* WAN.

There has been traditionally another type of networks – **MAN (Metropolitan Area Network)**, which refers to a network that is spread over an area as big as a city. But these days, this term has become redundant.

NOTE

The Internet is a giant WAN. The web is also a WAN. Please note that Web (World Wide Web) is a part of the Internet, not the complete Internet.

2. A server is a computer that just serves the requests of doing some tasks, made by other computers in its network. A *file server* serves the requests related to file sharing, storing etc. A *print server* serves the printer related requests, and so on.
3. The *network interface card* (NIC) provides the physical connection between the network and the computer workstation.
4. A leased line is a permanent telephone connection between two points set up by a government-regulated organization that provides telecommunications services to the public.

Following table (9.1) lists some basic differences between LAN and WAN.

Table 9.1 LAN *vs.* WAN

S.No.	LAN	WAN
1.	It is spread over a small area.	It is spread over a very large area.
2.	It usually costs less to set it up.	It costs higher to set it up.
3.	It is usually a single network.	It is usually a network of many networks.

9.3.1C Personal Area Network (PAN)

A personal area network (PAN) is the interconnection of information technology devices within the range of an individual person, typically within a range of 10 meters.

For example, a person traveling with a laptop, a personal digital assistant (PDA), and a portable printer could interconnect them without having to plug anything in, using some form of wireless technology such as *Wifi*. Typically, this kind of personal area network could also be interconnected without wires to the Internet or other networks. You can use PAN networks to transfer files including email and calendar appointments, digital photos and music etc from your portable devices such as phones and tablets to PC and vice versa.

Figure (Fig. 9.3) shows a typical PAN.

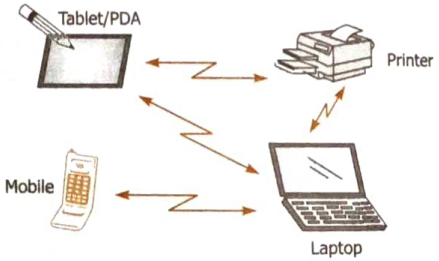


Figure 9.3 A Personal Area Network (PAN)

9.4 NETWORK TOPOLOGIES

The pattern of interconnection of nodes in a network is called the **Topology**.

The selection of a topology for a network cannot be done in isolation as it affects the choice of media and the access method used. There are a number of factors to consider in making this choice, the most important of which are set out below :

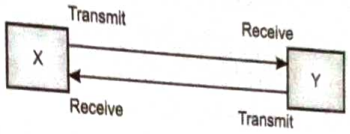
TOPOLOGY

"The pattern of interconnection of nodes in a network is called the Topology."

1. ***Cost***. For a network to be cost effective, one would try to minimize installation cost. This may be achieved by using well understood media and also, to a lesser extent, by minimizing the distances involved.
2. ***Flexibility***. Because the arrangement of furniture, internal walls etc. in offices is often subject to change, the topology should allow for easy reconfiguration of the network. This involves moving existing nodes and adding new ones.
3. ***Reliability***. Failure in a network can take two forms. Firstly, an individual node can malfunction. This is not nearly as serious as the second type of fault where the network itself fails to operate. The topology chosen for the network can help by allowing the location of the fault to be detected and to provide some means of isolating it.

9.4.1 Point-to-Point Link

Before we talk about topologies in details, let us learn about point-to-point link. To understand this, refer to Fig. 9.4. A *P-P* (Point to Point) link basically relies upon two functions – transmit and receive. The main characteristic of *P-P* network is that each station receives exactly from one transmitter, and each transmitter transmits to exactly one receiver. The transmit and receive operations can occur over separate wires (for better performance) or they can take turns over the same wire using a variety of techniques.



Figur 9.4 Point-to-Point Network.

Point-to-point networks can grow in several ways. One method is simply to install a *P-P* link between each pair of computers in the network. This approach is called a ***Mesh***.

Many topologies have been developed, but major ones are :

- the Star topology ;
- the Bus ;
- the Ring or circular ;
- the Tree ;
- the Mesh ;
- the Fully connected ;

9.4.2 The Star Topology

This topology consists of a central node to which all other nodes are connected by a single path. (*see* Fig. 9.5). It is the topology used in most existing information networks involving data processing or voice communications. The most common example of this is IBM 370 installations. In this case multiple 3270 terminals are connected to either a host system or a terminal controller.

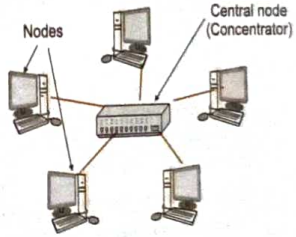


Figure 9.5 Star Topology

Advantages of the Star Topology

- *Ease of service*. The star topology has a number of concentration points (where connections are joined). These provide easy access for service or reconfiguration of the network.
- *One device per connection*. Connection points in any network are inherently prone to failure. In the star topology, failure of a single connection typically involves disconnecting one node from an otherwise fully functional network.
- *Centralized control/problem diagnosis*. The fact that the central node is connected directly to every other node in the network means that faults are easily detected and isolated. It is a simple matter to disconnect failing nodes from the system.
- *Simple access protocols*. Any given connection in a star network involves only the central node. In this situation, contention for who has control of the medium for the transmission purposes is easily solved. Thus in a star network, access protocols are very simple.

Disadvantages of the Star Topology

- *Long cable length*. Because each node is directly connected to the center, the star topology necessitates a large quantity of cable. Whilst the cost of cable is often small, congestion in cable ducts and maintenance and installation problems can increase cost considerably.

- *Difficult to expand.* The addition of a new node to a star network involves a connection all the way to the central node.
- *Central node dependency.* If the central node in a star network fails, the entire network is rendered inoperable. This introduces heavy reliability and redundancy constraints on this node.

The star topology has found extensive application in areas where intelligence in the network is concentrated at the central node.

9.4.3 The Bus or Linear Topology

Another popular topology for data networks is the linear. This consists of a single length of the transmission medium (normally coaxial cable) onto which the various nodes are attached (see Fig. 9.6). The topology is used in traditional data communication network where the host at one end of the bus communicates with several terminals attached along its length.

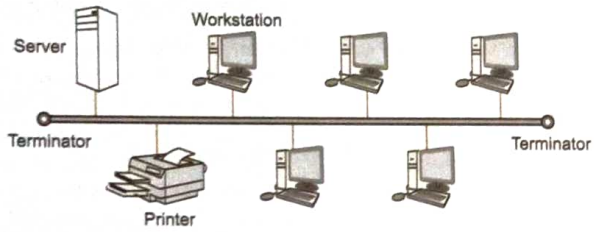


Figure 9.6 Bus Topology

The transmission from any station travels the length of the bus, in both directions, and can be received by all other stations. The bus has terminators at either end which absorb the signal, removing it from the bus.

Data is transmitted in small blocks, known as packets. Each packet has some data bits, plus a header containing its destination address. A station wanting to trans- mit some data sends it in packets along the bus. The destination device, on identifying the address on the packets, copies the data onto its disk.

Advantages of the Linear Topology

- *Short cable length and simple wiring layout.* Because there is a single common data path connecting all nodes, the linear topology allows a very short cable length to be used. This decreases the installation cost, and also leads to a simple, easy to maintain wiring layout.
- *Resilient Architecture.* The LINEAR architecture has an inherent simplicity that makes it very reliable from a hardware point of view. There is a single cable through which all the data propogates and to which all nodes are connected.
- *Easy to extend.* Additional nodes can be connected to an existing bus network at any point along its length. More extensive additions can be achieved by adding extra segments connected by a type of signal amplifier known as *repeater*.

Disadvantages of the Linear Topology

- *Fault diagnosis is difficult.* Although simplicity of the bus topology means that there is very little to go wrong, fault detection is not a simple matter. Control of the network is not centralized in any particular node. This means that detection of a fault may have to be performed from many points in the network.

- *Fault isolation is difficult.* In the star topology, a defective node can easily be isolated from the network by removing its connection at the center. If a node is faulty on the bus, it must be rectified at the point where the node is connected to the network.
- *Repeater configuration.* When BUS type network has its backbone extended using repeaters [5], reconfiguration may be necessary.
- *Nodes must be intelligent.* Each node on the network is directly connected to the central bus. This means that some way of deciding who can use the network at any given time must be performed in each node.

9.4.4 The Ring or Circular Topology

The third topology that we will consider is the ring or circular. In this case, each node is connected to two and only two neighbouring nodes. Data is accepted from one of the neighbouring nodes and is transmitted onwards to another (see Fig. 9.7). Thus data travels in one direction only, from node to node around the ring. After passing through each node, it returns to the sending node, which removes it.



Figure 9.7 Ring Topology

It is important to note that data 'passed through' rather than 'travels past' each node. This means that the signal may be amplified before being 'repeated' on the outward channel.

Advantages of the Ring Topology

- *Short cable length.* The amount of cabling involved in a ring topology is comparable to that of a bus and is small relative to that of a star. This means that less connections will be needed, which will in turn increase network reliability.
- *No wiring closet space required.* Since there is only one cable connecting each node to its immediate neighbours, it is not necessary to allocate space in the building for wiring closets.
- *Suitable for optical fibers.* Using optical fibers offers the possibility of very high speed transmission. Because traffic on a ring travels in one direction, it is easy to use optical fibers as a medium of transmission.

Disadvantages of the Ring Topology

- *Node failure causes network failure.* The transmission of data on a ring goes through every connected node on the ring before returning to the sender. If one node fails to pass data through itself, the entire network has failed and no traffic can flow until the defective node has been removed from the ring.
- *Difficult to diagnose faults.* The fact that failure of one node will affect all others has serious implications for fault diagnosis. It may be necessary to examine a series of adjacent nodes to determine the faulty one. This operation may also require diagnostic facilities to be built into each node.
- *Network reconfiguration is difficult.* It is not possible to shut down a small section of the ring while keeping the majority of it working normally.

[5] A repeater amplifies a signal being transmitted on the network.

9.4.5 The Tree Topology

A variation of bus topology is the *tree topology*. The shape of the network is that of an inverted tree with the central root branching and subbranching to the extremities of the network (Fig. 9.8).

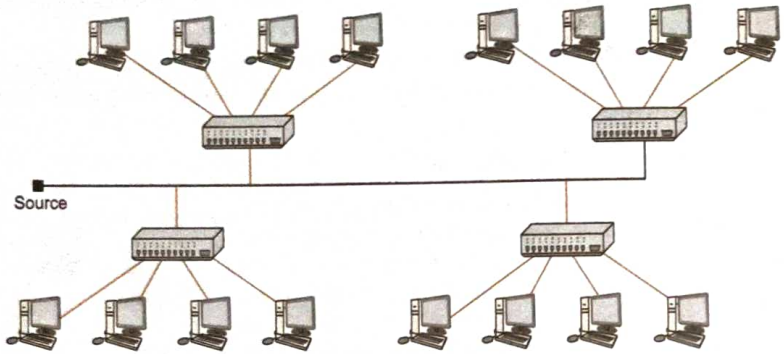


Figure 9.8 Tree Topology

Transmission in this topology takes place in the same way as in the bus topology. In both cases, there is no need to remove packets from the medium because when a signal reaches the end of the medium, it is absorbed by the terminators. Tree topology is best suited for applications which have a hierarchical flow of data and control. Since the tree topology is a modification of a 'pure' network topology, bus topology, it is a *hybrid* topology.

Aadvantages of the Tree Topology

- It uses point-to-point wiring for individual segments.
- It is supported by several hardware and software venders.

Disadvantages of the Tree Topology

- Overall length of each segment is limited by the type of cabling used.
- If the backbone line breaks, the entire segment goes down.
- It is more difficult to configure and wire than other topologies.

9.4.6 Mesh Topology

In this topology, each node is connected to more than one node to provide an alternative route in the case the host is either down or too busy. It is an extension to *P-P* network. (*see* Fig. 9.9).

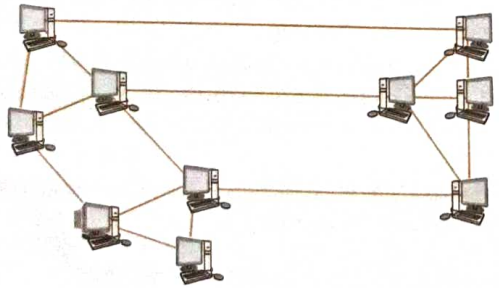


Figure 9.9 Mesh Topology

The mesh topology is excellent for long distance networking because it provides extensive back-up, rerouting and pass-through capabilities. Communication is possible between any two nodes on the network either directly or by passing through. This function is needed in the event of a line or node failure elsewhere in the network. The mesh topology is commonly used in large internetworking environ- ments with stars, rings, and buses attached to each node. This is also ideal for distributed networks.

Advantages of Mesh Topology

- Each connection can carry its own data load.
- It is robust and provides security and privacy.
- In this topology, fault diagnosis is easy.

Disadvantages of Mesh Topology

- Its installation and configuration is difficult.
- Cabling cost is more for mesh topology.
- Bulk wiring is required for mesh topology.

9.4.7 Fully Connected

When in a network each host is connected to other directly *i.e.*, there is a direct link between each host, then the network is said to be fully connected. This characteristic is termed as full connectivity. (Fig. 9.10)

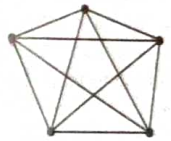


Figure 9.10 Fully connected network.

9.4.8 Factors to Consider for Topology Selection

As there are many different types of topologies that can be installed for a computer network, what all factors one must consider while deciding about the preferred topology for a network ?

The factors that you must keep in mind for this, are :

- **Cruciality of Work.** How crucial the continuity of work is a very important factor. For instance, military networks must not fail at any cost thus **fully connected topology** is a preferred for Military networks.
- **Cost.** Keeping in mind the budget, you should decide about a topology. A **linear bus topology** network may be ***the least expensive*** way to install a network; **Fully connected *is the most expensive*** way of creating a network.
- **Length of cable needed.** Sometimes, length of cable must be saved on; the **linear bus topology** network uses shorter lengths of cable.
- **Future growth.** If a network has to grow in future, then the topology must support expansion. With a **star topology**, expanding a network is easily done by adding another concentrator.
- **Communication Media.** Sometimes, difficult terrains like hilly areas do not allow use of regular cables. For such conditions, *linear bus topologies* are not possible. The most common cable in schools is *unshielded twisted pair*, which is most often used with *star topologies*.

9.5 Network Devices

In the smooth functioning of a computer network, other than computers and wiring, many devices or specialized hardware play important roles. In this section, we are discussing about these network devices/hardware.

9.5.1 Modem

Modems allow you to combine the power of your computer with the global reach of the telephone system. Modem is a device which allows a compuer to send and receive data over telephone lines or cable connections.

> **MODEM**
> "A Modem is a computer peripheral that allows you to connect and communicate with other computers via telephone lines."

Because ordinary telephone lines cannot carry digital information, a modem changes the digital data from your computer into analog data, a format that can be carried by telephone lines. In a similar manner, the modem receiving the call then changes the analog signal back into digital data that the computer can digest. This shift of digital data into analog data and back again, allows two computers to "speak" with one another. Called **modulation/demodulation**, this transformation of signals is how the modem received its name.

Modulation is the process of sending data on a wave, 3 types of modulation tech- niques are used :

AM (*Amplitude Modulation*),
FM (*Frequency Modulation*),
PM (*Phase Modulation*)

Modems come in *two* varieties :

- ***Internal modems*** the modems that are fixed within the computer [*see* Fig. 9.11(*a*)].
- ***External modems*** the modems that are connected externally to a computer as other peripherals are connected [*see* Fig. 9.11(*b*)].



(*a*) Internal modem (*b*) External modem

Figure 9.11

9.5.2 Hub

A **hub** is a hardware device used to connect several computers together. A *hub* contains multiple independent but connected modules of network and inter-networked equipment. A similar term is **concentrator**. A *concentrator* is a device that provides a central connection point for cables from workstations, servers, and peripherals.

> **HUB**
> "A **hub** is networking device having multiple ports that are used for connecting multiple computers or segments of a LAN together."

Basically, hubs are multi-slot concentrators into which a number of multi-port cards can be plugged to provide additional access as the network grows in size.

Hubs can be either *passive* or *active*.

- **Active hubs** electrically amplify the signal as it moves from one connected device to another. Active concentrators are used like *repeaters*[6] to extend the length of a network.
- **Passive hubs** allow the signal to pass from one computer to another without any change.

9.5.3 Switch

A switch is a device that is used to segment networks into different *subnetworks* called ***subnets*** or ***LAN segments***. Segmenting the network into smaller subnets, prevents traffic overloading in a network.

A switch is responsible for *filtering i.e.*, transforming data in a specific way and for forwarding *packets* (a piece of message being transmitted) between LAN segments. Switch can support any packet protocol. LANs that are segmented through switches are called *Switched LANs*.

How a Switch functions

To insulate the transmission from the other ports, the switch establishes a temporary connection between the source and destination, and then terminates the connection once the conversation is done.

A switch would be like a phone system with private lines in place of the hub's *party line*. For instance, *Meira Sen* at the Maurya Hotel calls *Ibrahim Soz* in another room, and the operator or phone switch connects the two of them on a dedicated line. This allows more conversations at any one time thereby allowing more guests to communicate.

9.5.4 Repeater

A *repeater* is a device that amplifies a signal being transmitted on the network. It is used in long network lines, which exceed the maximum rated distance for a single run.

Over distance, the cables connecting a network lose the signal transmitted. If the signal degrades too much, it fails to reach the destination. Or if it does arrive, the degradation of the message makes it useless. *Repeaters* can be installed along the way to ensure that data packets reach their destination. *Repeaters* are of two kinds : *amplifier* and *signal repeater*. The first merely amplifies all incoming signals over the network. However, it amplifies both the signal and any concurrent noise. The second type collects the inbound packet and then retransmits the packet as if it were starting from the source station.

> **REPEATER**
> "A Repeater is a network device that amplifies and restores signals for long- distance transmission."

9.5.5 Bridge

A *bridge* is a device that lets you link two networks together. *Bridges* are smart enough to know which computers are on which side of the bridge, so they only allow those messages that need to get to the other side to cross the bridge. Bridges can handle networks that follow same protocols.

9.5.6 Router

A device that works like a bridge but can handle different protocols, is known as a *router*. For example, a router can link Ethernet to a mainframe. The router is responsible for forwarding data from one network to a different network.

6. A repeater is a device that electrically amplifies the signal it receives and rebroadcasts it.

If the destination is unknown to a router it sends the traffic (bound to unknown destination) to another router (using logical addresses) which knows the destination. Based on a network road map called a *routing table*, routers can help ensure that packets are travelling the most efficient paths to their destinations. If a link between two routers fails, the sending router can determine an alternate route to keep traffic moving.

ROUTER

"A Router is a network device that forwards data from one network to another. A router works like a *bridge* but can handle different protocols."

A router differs from a bridge in a way that former uses logical addresses and the latter uses physical addresses.

9.5.7 Gateway

A *gateway* is a device that connects dissimilar networks. A gateway is actually a node on a network that serves as an entrance to another network. In enterprises, the gateway is the computer that routes the traffic from a workstation to the outside network that is serving the Web pages. In homes, the gateway is the ISP that connects the user to the Internet.

GATEWAY

"A **Gateway** is a network device that connects dissimilar networks. It establishes an intelligent con- nection between a local network and external networks with completely different structures."

In enterprises, the gateway node often acts as a *proxy server* (a machine that is not actually a server but appears as a server) and a *firewall* (a system designed to prevent unauthorized access to or from a private network). The gateway is also associated with both a *router*, which use headers and forwarding tables to determine where packets are sent, and a *switch*, which provides the actual path for the packet in and out of the gateway.

Check Point 9.1

1. Write two advantages and disadvantages of networks.
2. What is meant by topology ? Name some popular topologies.
3. What are the factors that must be considered before making a choice for the topology ?
4. What are the similarities and differences between bus and tree topologies ?
5. How is a PAN different from a LAN ?
6. What is NIC ?
7. Give two advantages and two disadvantages of following network topologies :
   (*i*) Star (*ii*) Tree (*iii*) Bus.
8. What is a modem ? Name two categories of modems.
9. Define the following : (*i*) Hub (*ii*) Switch (*iii*) Repeater (*iv*) Bridge (*v*) Router (*vi*) Gateway.

LET US REVISE

- *A network is a collection of interlinked computers by means of a communication system.*
- *The networks facilitate resource sharing, increased reliability, reduced costs, and increased and fast communication.*
- *On the basis of geographical spread, networks can be classified into LAN (Local Area Network), WAN (Wide Area Network), PAN (Personal Area Network) and MAN (Metropolitan Area Network).*
- *Small computer networks that are confined to a localised area e.g., an office, a building etc., are called LANs.*
- *MANs are the networks spread over a city.*
- *A WAN is a group of computers that are separated by large distances and tied together. It can even be a group of LANs that are spread across several locations and connected together to look like a big LAN.*
- *The pattern of interconnection of nodes in a network is called topology.*
- *Most popular topologies are star, bus, ring, tree and mesh.*
- *A modem is a computer peripheral that allows you to connect and communicate with other computers via telephone lines.*

- *A hub is a hardware device used to connect several computers together. Hubs can be either active hubs or passive hubs.*
- *A switch is a device that is used to segment networks into different subnetworks called subnets or LAN segments.*
- *A backbone network is a network that is used as a backbone to connect LANs together to form a WAN.*
- *A repeater is a device that amplifies a signal being transmitted on the network.*
- *A bridge is a device that links two networks together.*
- *A router is a device that works like a bridge but can handle different protocols.*
- *A gateway is a device that connects dissimilar networks.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. Two devices are in network if
   (*a*) a process in one device is able to exchange information with a process in another device
   (*b*) a process is running on both devices
   (*c*) the processes running of different devices are of same type
   (*d*) none of the mentioned
2. What is a stand alone computer ?
   (*a*) A computer that is not connected to a network
   (*b*) A computer that is being used as a server
   (*c*) A computer that does not have any peripherals attached to it
   (*d*) A computer that is used by only one person
3. Central Computer which is powerful than other computers in the network is called as _____ .
   (*a*) Client (*b*) Server (*c*) Hub (*d*) Switch
4. Which of these components is internal to a computer and is required to connect the computer to a network ?
   (*a*) Wireless Access Point (*b*) Network Interface card
   (*c*) Switch (*d*) Hub
5. Hub is a
   (*a*) Broadcast device (*b*) Unicast device
   (*c*) Multicast device (*d*) None of the above
6. Switch is a
   (*a*) Broadcast device (*b*) Unicast device
   (*c*) Multicast device (*d*) None of the above
7. The device that can operate in place of a hub is a :
   (*a*) Switch (*b*) Bridge (*c*) Router (*d*) Gateway
8. A repeater takes a weak and corrupted signal and _____ it.
   (*a*) Amplifies (*b*) Regenerates
   (*c*) Resembles (*d*) Reroutes

9. Which network topology requires a central controller or hub?
   (a) Star (b) Bus (c) Mesh (d) Tree
10. Which topology requires a multipoint connection on a single cable?
   (a) Star (b) Bus (c) Mesh (d) Tree
11. Which of the following topologies contains a backbone cable running through the whole length of the network ?
   (a) Star (b) Bus (c) Mesh (d) Tree
12. Which of the following devices can be used at the centre of a star topology?
   (a) Router (b) Repeater (c) Modem (d) Hub
13. If a computer connected to a star topology fails, the entire network will _____
   (a) also fail (b) work unaffectedly
   (c) only server will work (d) none of these
14. A combination of bus and star topologies is called a _____ topology.
   (a) Combination (b) Hybrid (c) Mesh (d) Tree
15. Internet is an example of _____ topology.
   (a) Star (b) Bus (c) Mesh (d) Bus
16. Data is converted in a form so as to travel over telephone lines using this device.
   (a) Modem (b) Hub (c) Switch (d) Router
17. Network device that regenerates and retransmits the whole signal is _____ .
   (a) Modem (b) Hub (c) Repeater (d) Bridge
18. Network device which connects networks of similar types (*same protocols*).
   (a) Hub (b) Router (c) Bridge (d) Gateway
19. Network device which connects dissimilar networks (*different protocols*).
   (a) Hub (b) Router (c) Bridge (d) Gateway
20. Networks devices that sends the data over optimizing paths through connected hops is
   (a) Hub (b) Router (c) Bridge (d) Gateway

Fill in the Blanks

1. A computer network that spans a relatively large geographical area is called WAN.
2. WAN stands for _____ .
3. A computer part of a computer network is called a node
4. A network having a span within a building is called a LAN
5. A network having a span of city, is called a MAN
6. The hub is a central connection point where all network cables are concentrated.
7. The bridge is a network device that can connect the network segments based on the same protocol.
8. The router is a network device that navigates the data packets over large networks through the most efficient route.
9. The _____ is a network device that connects dissimilar networks.
   Gateway

10. The reap is a networking device that regenerates or recreates a weak signal into its original strength and form.
11. The bus topology has a central line and all nodes are connected to it.
12. The star topology has a central controller.
13. The tree topology is said to be a combination of Star and bus topologies.
14. In mesh topology, there exists multiple paths between any two nodes of the network.
15. The _____ is a small network created by linking many personal devices.
    PAN

True/False Questions

1. A LAN is connected to large geographical area. F
2. A switch can work is place of a hub. T
3. A gateway is like a modem. F
4. LAN is the biggest network geographically. F
5. LAN is the smallest network geographically. F
6. PAN is the smallest network geographically. T
7. The Internet is an example of WAN. T
8. The bus topology is the simplest topology. T
9. The star topology ensures that the network will work even when a node fails. T
10. Amplifier and a repeater do the same thing. F
11. A hub broadcasts the received signal to all its connected devices. T
12. A switch is an intelligent hub. T
13. Repeater and router precisely do the same thing. F
14. A hub can be replaced with a switch. T
15. A router can be replaced with a bridge. F

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *What is a network ? Why is it needed ?* **Or** **(Outside Delhi 2001)**
   Mention one advantage of networking.

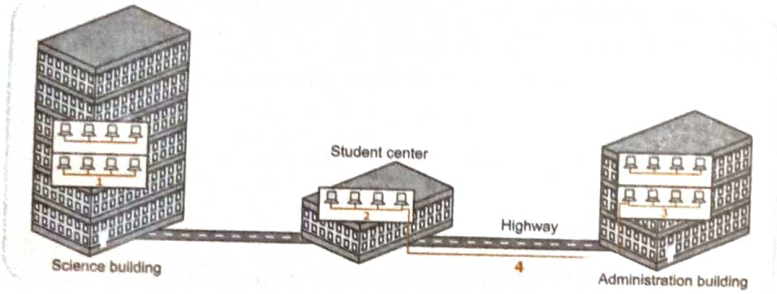
   Solution. A network is an interconnected collection of autonomous computers that can share and exchange information. Major reasons that emphasize on the need of networks are :
   (i) *Resource Sharing*. Through a network, data, software and hardware resources can be shared irrespective of the physical location of the resources and the user.
   (ii) *Reliability*. A file can have its copies on two or more computers of the network, so if one of them is unavailable, the other copies could be used. That makes a network more reliable.
   (iii) *Reduced Costs*. Since resources can be shared, it greatly reduces the costs.
   (iv) *Fast communication*. With networks, it is possible to exchange information at very fast speeds.

2. *What are the different types of networking/internetworking devices ?*

Solution. Different types of networking/internetworking devices are :

- *Repeater.* Also called a regenerator, it is an electronic device that operates only at physical layer. It receives the signal in the network before it becomes weak, regenerates the original bit pattern and puts the refreshed copy back in to the link.
- *Bridges.* These operate both in the physical and data link layers of LANs of same type. They divide a larger network in to smaller segments. They contain logic that allow them to keep the traffic for each segment separate and thus are repeaters that relay a frame only the side of the segment containing the intended recipent and control congestion.
- *Routers.* They relay packets among multiple interconnected networks (*i.e.*, LANs of different type). They operate in the physical, data link and network layers. They contain software that enable them to determine which of the several possible paths is the best for a particular transmission.
- *Gateways.* They relay packets among networks that have different protocols (*e.g.*, between a LAN and a WAN). They accept a packet formatted for one protocol and convert it to a packet formatted for another protocol before forwarding it. They operate in all seven layers of the OSI model.

3. *Govt. of Delhi has computer networks inside each of its buildings. It has now interconnected the networks of Administration building and of Student center building. The networks so formed are marked below as numbers. Mention which types of networks is each of these ?*



Solution.

Network number	Type of network
1, 2, 3	LAN
4	WAN (it is connecting only networks 2 and 3)

4. *What is the difference between Hub, Switch, and Router?*

Solution.

Hub	Switch	Router
Hub is the least expensive, least intelligent and least complicated of the three. It broadcasts all data to every port which may cause serious security and reliability concern.	Switches work similarly like Hubs but in a more efficient manner. It creates connections dynamically and provides information only to the requesting port.	The router is smartest and most complicated out of these three. It comes in all shapes and sizes. Routers are similar like little computers dedicated for routing network traffic.
In a Network, Hub is a common connection point for devices connected to the network. Hub contains multiple ports and is used to connect segments of LAN.	Switch is a device in a network which forwards packets in a network.	Routers are located at gateway and forwards data packets.

5. *What is a Hub ?*

Solution. A *hub* is a hardware device used to connect several computers together.

6. *What are the different types of networks ? What is the geographical scope of LAN, MAN and WAN ?*

Solution. Networks vary widely in their size, complexity and geographical spread. On the basis of geographical spread, networks can be classified into three categories :

(*i*) *Local Area Networks* (*LANs*). These are computer networks confined to a localised area such as an office or a factory.

(*ii*) *Metropolitan Area Networks* (*MANs*). These are the networks that link computer facilities within a city.

(*iii*) *Wide Area Networks* (*WANs*). These are the networks spread over large distances, say across countries or even continents. It can even include a group of LANs connected together.

7. *What are the important topologies for networks ?*

Solution. Important network topologies are :

- **BUS topology.** In this each computer is directly connected to primary network cable in a single line.

  Advantage. Inexpensive, easy to install, simple to understand, easy to extend.

- **STAR topology.** In this all computers are connected using a central hub.

  Advantage. Can be inexpensive, easy to install and reconfigure and easy to trouble shoot physical problems.

- **RING topology.** In this all computers are connected in loop.

  Advantage. All computers have equal access to network media, installation can be simple, and signal does not degrade as much as in other topologies because each computer regenerates it.

8. *Explain in brief the capabilities and services supported by LAN.*

Solution. Small computer networks that are confined to a localised area (*e.g.*, an office, a building or a factory) are known as *Local Area Networks* (LANs). The key purpose of a LAN is to serve its users in resource sharing. The hardware as well as software resources are shared through LANs. For instance, LAN users can share data, information, programs, printer, hard-disks, modems etc. One node has a printer connected to it and other nodes on the LAN can communicate with it in order to print files and hence allowing expensive peripherals to be shared among number of users.

9. *What is the purpose of using router ?* (CBSE Outside Delhi 1999)

Solution. A router can work like a bridge and can also handle different protocols. A router can locate the destination required by sending the traffic to another router, if the destination is unknown to itself.

10. (*a*) *Give two examples of PAN and LAN type of networks.* (CBSE Delhi 2016)

(*b*) *Differentiate between PAN and LAN types of networks.* (CBSE Outside Delhi 2016)

Solution.

(*a*) **PAN :** *Examples*

(*i*) Network formed by connecting smartphones of family members to laptop via bluetooth.

(*ii*) Network formed by connecting devices like printer, laptop, smartphone, digital recorder etc.

LAN : *Examples*

(*i*) Network formed by computers in an office.

(*ii*) Network formed by computers in a bank.

(*b*) PAN expands to *Personal Area Network*.

LAN expands to *Local Area Network*.

A PAN is a computer network organized around an individual person where a small network is formed by connecting various devices of the individual *e.g.*, a laptop, a printer, a smartphone, digital recorder etc.

LAN interconnects some stand-alone computers within a confined physical area upto a kilometer, *e.g.*, a LAN inside a university or a LAN inside a hospital etc.

11. *What are two types of modems ?*

Solution. Modems come in *two* varieties :

1. ***Internal modems*** are the modems that are fixed within the computer.
2. ***External modems*** are the modems that are connected externally to a computer as other peripherals are connected.

12. *What do you mean by network topology ? What are the most popular topologies ?* (CBSE Question Bank)

Solution. Topology refers to the way in which the workstations attached to the network are interconnected. The most popular topologies are :

▲ Bus or Linear ▲ Ring ▲ Star ▲ Tree

1. *Bus or Linear Topology*. In this topology, all devices on network are connected to a single continuous cable called a bus. Transmission from any station travels the length of the bus in both directions and can be received by all other stations. The destination device, on identifying the address on data packet copies the data onto its disk. When the data packet reaches at either end the terminator on that end absorbs the signal, removing from the bus.

   This topology can be used for smaller networks.

2. *Ring Topology*. A LAN using the ring topology is connected in the closed loop. The data packets transmitted, circulate along the ring. The destination station copies the packet content on recognizing its address on the packet. After a packet travels a full circle, it is removed at the source station.

3. *Star Topology*. In this topology each workstation is directly linked to a central node. Devices can be easily plugged or unplugged to the central node, as need dictates. Any communication between the stations must pass through the central node.

4. *Tree Topology*. In this topology the network is shaped as an inverted tree with the central root branching and sub-branching to the extremities of the network. Transmission in this topology takes place in the same way as in bus topology.

13. *What is a modem ? What is its function ?*

Solution. A modem is a computer peripheral that connects a workstation to other work-stations via telephone lines and facilitates communications. It is short form for Modulation / Demodulation.

Modem converts digital signals to A/F (Audio Frequency) tones which are in the frequency range that the telephone lines can transmit and also it can convert transmitted tones back to digital information.

14. *What are repeaters and routers ?*

Solution. **REPEATER.** A *repeater* is a device that amplifies a signal being transmitted on the network. It is used in long network lines, which exceed the maximum rated distance for a single run.

Over distance, the cables connecting a network lose the signal transmitted. If the signal degrades too much, it fails to reach the destination. Or if it does arrive, the degradation of the message makes it useless. *Repeaters* can be installed along the way to ensure that data packets reach their destination. *Repeaters* are of two kinds — *amplifier* and *signal repeater*.

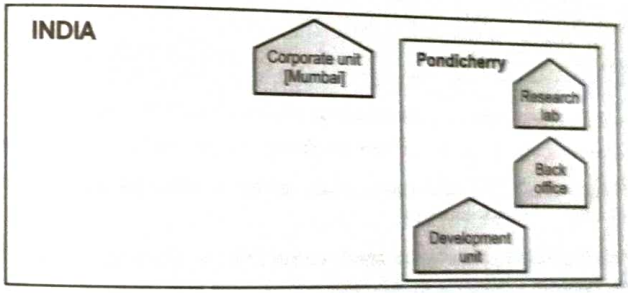
The *first* merely amplifies all incoming signals over the network. However, it amplifies both the signal and any concurrent noise. The *second* type collects the inbound packet and then retransmits the packet as if it were starting from the source station.

ROUTER. A device that works like a bridge but can handle different protocols, is known as a *router*. For example, a router can link Ethernet (ethernet is a very popular and widely accepted method of linking local stations to one another (*i.e.*, a LAN) for sharing data, program and equipment resources.) to a mainframe.

If the destination is unknown to a router it sends the traffic (bound to unknown destination) to another router (using logical addresses) which knows the destination.

A router differs from a bridge in a way that former uses logical addresses and the latter uses physical addresses.

15. *"Bias Methodologies" is planning to expand their network in India, starting with three cities in India to build infrastructure for research and development of their chemical products. The company has planned to setup their main office in Pondicherry – at three different locations and have named their offices as "Back Office", "Research Lab" and "Development Unit". The company has one more Research office namely "Corporate Office" in "Mumbai". A rough layout of the same is as follows :*



Approximate distance between these offices is as follows :

From	To	Distance
Research Lab	Back Office	110 Mtr
Research Lab	Development Unit	16 KM
Research Lab	Corporate Unit	1800 KM
Back Office	Development Unit	13 KM

In continuation of the above, the company experts have planned to install the following number of computers in each of their offices :

Research Lab	158
Back Office	79
Development Unit	90
Corporate Unit	51

(i) *Suggest the kind of network required (out of LAN, MAN, WAN) for connecting each of the following office units :*

- Research Lab and Back Office
- Research Lab and Development Unit

(ii) *Which one of the following devices will you suggest for connecting all the computers with in each of their office units ?*

- Switch/Hub
- Modem
- Telephone

(Delhi 2008)

Solution.

(i) Between Research Lab and Back office – LAN
Between Research Lab and Development unit – MAN

(ii) Switch/hub

16. *Software Development Company has set up its new center at Raipur for its office and web based activities. It has 4 blocks of buildings named Block A, Block B, Block C, Block D.* [CBSE Sample Paper 2019-20]

Number of Computers

Block A	25
Block B	50
Block C	125
Block D	10

Shortest distances between various Blocks in meters

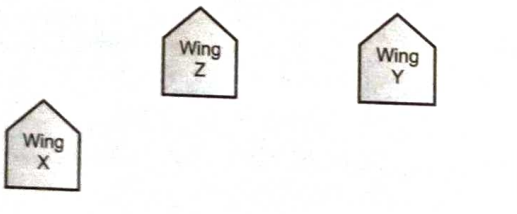
Block A to Block B	60 m
Block B to Block C	40 m
Block C to Block A	30 m
Block D to Block C	50 m

(i) *Suggest the type of network to connect all the blocks with suitable reason.*

(ii) *The company is planning to link all the blocks through a secure and high speed wired medium. Suggest a way to connect all the blocks.*

Solution. (i) LAN (ii) Star topology

17. *The Great Brain Organisation has set up its new Branch at Srinagar for its office and web based activities. It has 4 Wings of buildings as shown in the diagram :*



Center to center distances between various blocks

Wing X to Wing Z	50 m
Wing Z to Wing Y	70 m
Wing Y to Wing X	125 m
Wing Y to Wing U	80 m
Wing X to Wing U	175 m
Wing Z to Wing U	90 m

Number of Computers

Wing X	50
Wing Z	30
Wing Y	150
Wing U	15

(i) *Suggest the most suitable cable layout of connections between the Wings, and topology.*

(ii) *Suggest the placement of the following devices with justification :* (1) *Repeater* (2) *Hub/Switch*

(Delhi 2005)

Solution.

(i) **Bus Topology**

(iii) (1) As per suggested layout separate repeaters need not be installed as each building/wing will be having a hub that acts as a repeater.
(2) One hub per wing.

GLOSSARY

Hub	Hardware device used to connect several computers together.
Network	An interconnected collection of autonomous computers.
Repeater	Device that amplifies a signal being transmitted on a network.
Router	Device that works like a bridge but can handle different protocols.
Switch	Device used to segment networks into different subnetworks called subnets.
Topology	The pattern of interconnection of nodes in a network.

Assignment

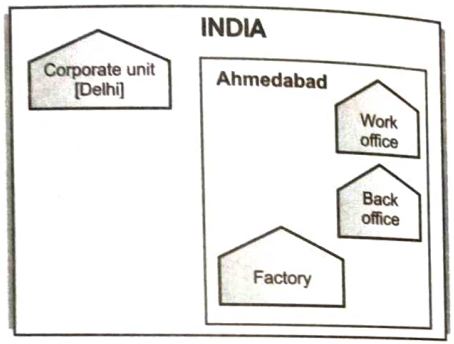
Short Answer Questions

1. What is a network ? What are its goals and applications ?
2. Discuss and compare various types of networks.
3. Explain various mostly used topologies.
4. Discuss the factors that govern the selection of a topology for a network.
5. Compare and contrast
   (i) Star and Bus topologies
   (ii) Star and Tree topologies
   (iii) Bus and Ring topologies.
6. What is the role of modem in electronic communications ?
7. What are hubs ? What are its types ?
8. What is the role of a switch in a network ?
9. Briefly discuss the role of following devices in the context of networking.
   (i) repeater (ii) router (iii) bridge (iv) gateway.
10. What is the importance of router?
11. What is gateway? What does it do?
12. How is a repeater different from an amplifier?
13. How is a LAN different from a MAN?
14. How is a LAN different from a PAN?
15. How is a LAN different from a WAN?

16. Can you term a PAN as a LAN? Why/Why not?
17. Compare Mesh with Tree topology.
18. Compare Tree and Star topology.
19. Write two advantages and two dis- advantages for STAR topology.
20. (*a*) Write advantages and disadvantages of bus topology.
    (*b*) Write advantages and disadvantages of tree topology.
    (*c*) Write advantages and disadvantages of mesh topology.

Long Answer/Pracrical Questions

1. "New York Avenue" is planning to expand their network in India, starting with two cities in India to provide infrastructure for distribution of their canned products. The company has planned to set up their main offices in Ahmedabad, at three different locations and have named their offices as "Work Office", "Factory" and "Back Office". The company has its Corporate Unit in Delhi. A rough layout of the same is as follows :



Approximate distance between these offices is as follows :

From	To	Distance
Work Office	Back Office	110 Mtr
Work Office	Factory	14 KM
Work Office	Corporate Office	1280 KM
Back Office	Factory	13 KM

In continuation of the above, the company experts have planned to install the following number of computers in each of their offices :

Work Office	200
Back Office	115
Factory	67
Corporate	75

(*i*) Suggest the kind of network required (out of LAN, MAN, WAN) for connecting each of the following offices :
- Work Office and Factory
- Work Office and Back Office

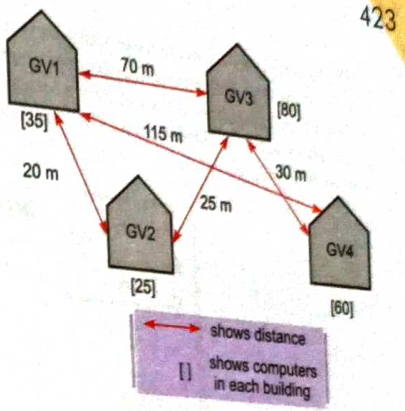
(*ii*) Which one of the following devices will you suggest for connecting all the computers within each of their office units ?
- Switch/Hub
- Modem
- Telephone

2. Global Village Enterprises has following four buildings in Hyderabad city :

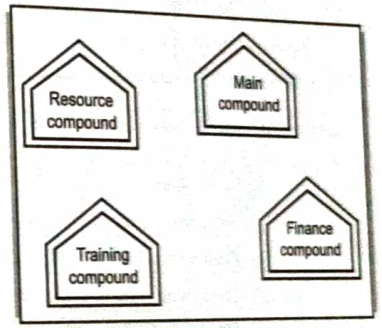
Computers in each building are networked but buildings are not networked so far. The company has now decided to connect buildings also.

(*a*) Suggest a cable layout for these buildings.

(*b*) In each of the buildings, the management wants that each LAN segment gets a dedicated bandwidth *i.e.*, bandwidth must not be shared. How can this be achieved ?



3. "Learn Together" is an educational NGO. It is setting up its new campus at Jabalpur for its webbased activities. The campus has 4 compounds as shown in the adjacent diagram.



Center to center distances between various Compounds as per architectural drawings (in Metre) is as follows :

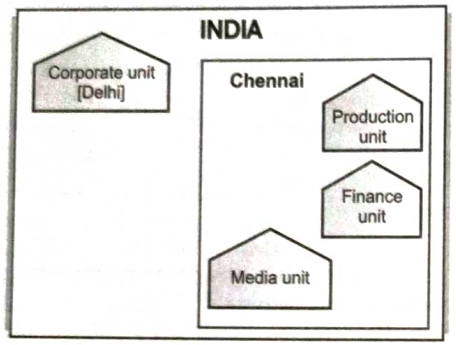
Main Compound to Resource Compound	110 m
Main Compound to Training Compound	115 m
Main Compound to Finance Compound	35 m
Resource Compound to Training Compound	25 m
Resource Compound to Finance Compound	135 m
Training Compound to Finance Compound	100 m

Expected Number of Computers in each Compound is as follows :

Main Compound	5
Resource Compound	15
Training Compound	150
Accounts Compound	20

(*a*) Suggest a cable layout of connections between the compounds.
(*b*) Suggest the placement of the following devices with justification : (*i*) Repeater (*ii*) Hub/Switch

4. "China Middleton Fashion" is planning to expand their network in India, starting with two cities in India to provide infrastructure for distribution of their product. The company has planned to setup their main office in Chennai at three different locations and have named their offices as "Production Unit", "Finance Unit" and "Media Unit". The company has its corporate unit in Delhi. A rough layout of the same is as follows :



Approximate distance between these Units is as follows :

From	To	Distance
Production Unit	Finance Unit	70 Mtr
Production Unit	Media Unit	15 KM
Production Unit	Corporate Unit	2112 KM
Finance Unit	Media Unit	15 KM

In continuation of the above, the company experts have planned to install the following number of computers in each of their offices :

Production Unit	150
Finance Unit	35
Media Unit	10
Corporate Unit	30

(*i*) Suggest the kind of network required (out of LAN, MAN, WAN) for connecting each of the following office units :

- Production Unit and Media Unit
- Production Unit and Finance Unit

(*ii*) Which one of the following devices will you suggest for connecting all the computers within each of their office units ?

- Switch/Hub
- Modem
- Telephone

(*iv*) Suggest a cable/wiring layout for connecting the company's local office units located in Chennai. Also, suggest an effective method/technology for connecting the company's office unit located in Delhi.

10 Introduction to Internet and Web

Outline

10.1 Introduction
10.2 Introduction to Internet
10.3 World Wide Web (WWW)
10.4 Web Servers
10.5 URLs
10.6 Website
10.7 Web Pages
10.8 Web Browsers
10.9 Internet Applications
10.10 Chat
10.11 Voice over Internet Protocol

10.1 Introduction

We all know the reach of Internet in our lives and society ; it is simply incalculable. Television programs cover it, newspapers and magazines write about it, every other person is using it, and so forth. Even the presidents, vice-presidents and prime ministers of various countries have cut a presence on the ***Internet*** or ***Net*** in short. The simplest definition of the Internet is that it is the largest computer network in the world. But technically speaking, the Internet is actually a network of many smaller networks that exist all over the world.

In this chapter we shall talk about **the Internet** and **the Web**, which generally people think is the other name of the Internet but which is not.

10.2 Introduction to Internet

Internet is a computer-based worldwide information *network*. It can be assumed as an example of WAN (Wide Area Network). The Internet is composed of a large number of smaller interconnected networks. These networks may link tens, hundreds, or thousands of *computers*, enabling them to share information with each other and to share various resources, such as powerful *supercomputers* and *databases* (collection of data) of information. The Internet has made it possible for people all over the world to effectively and inexpensively communicate with each other. Unlike traditional broadcasting media, such as radio and television, the Internet is a decentralized system. Each connected individual can communicate with anyone else on the Internet, can publish ideas, and can sell products with a minimum overhead cost.

History of Internet

The seeds of Internet were planted in 1969, when *U.S. Department of Defense* sponsored a project named ARPANET (acronym for **Advanced Research Projects Agency NETwork**). In mid 80's, another federal agency, the *National Science Foundation*, created a new, high-capacity network called **NSFnet**, which was more capable than ARPANET.

It was the *Inter-networking i.e.*, the linking of these two and some other networks (*i.e.*, the ARPANET, NSFnet and some private networks) that was named **Internet**.

The original ARPANET was shut down in 1990, and the government funding for NSFnet discontinued in 1995. But the commercial Internet services came into picture, which are still running the Internet.

INTERNET

> "Internet is a computer-based world wide communications network, which is composed of large number of smaller interconnected networks."

Advantages of Internet

- Greater access to information reduces research times.
- Useful communication links.
- Global reach enables one to connect to anyone on the Internet.
- Easy communication with other people.
- Publishing documents on the Internet saves paper.
- A valuable resource for companies to advertise and conduct business.

Disadvantages of the Internet

- Cyber frauds may take place involving Credit/Debit card numbers and details.
- Unsuitable and undesirable material available that sometimes are used by notorious people such as terrorists.
- Computer viruses get downloaded and spread across machines connected to networks and have detrimental effects.
- Much of the information isn't checked and may be incorrect or irrelevant.
- Messages sent across the Internet can be easily intercepted and are open to abuse by others.
- Too much time spent on the Internet could result in a lack of face-to-face interaction with others and a loss of social skills.
- Going on-line runs the risk of hackers or viruses being able to damage your computer.

10.3 World Wide Web (WWW)

From the late 1960s to the early 1990s, the Internet was a communication and research tool used almost exclusively for academic and military purposes. This changed radically with the introduction of the *World Wide Web* (WWW) or Web in short in 1989. The WWW is a set of *programs*, standards and protocols governing the way in which *multimedia* files (*multimedia files are the documents that contain a combination of text, photographs, graphics, video and audio*) are created and displayed on the Internet.

WORLD WIDE WEB

> "The World Wide Web (WWW) is a set of *programs, standards,* and *protocols* that allows the multimedia and hypertext files to be created, displayed and linked on the Internet."

Before WWW, Internet was mainly used for obtaining textual information. But post-WWW, the Internet popu- larity grew tremendously because of graphic- intensive nature of WWW. Therefore, we may attribute the explosion in use and popularity of Internet to WWW only.

The *World Wide Web (WWW)* is also affectionately called "The Web". WWW was proposed between 1989-1991 by *Tim Berners-Lee*. Proposed in 1989, WWW became a reality with invention of HTML and a text browser in 1990.

NOTE

WWW was proposed between 1989-1991 by Tim Berners-Lee.

10.3.1 History of World Wide Web

Present day WWW is about 21 years of age with its official birth date as April 30, 1993. In the following table, we are giving a brief history of WWW.

1989	*On March 12,* **Tim Berners-Lee**, a British scientist working at CERN in Switzerland, submits a proposal for a "distributed information system" to allow researchers to deal with huge amounts of information. This is the genesis of the *World Wide Web*.
1990	*On December 20,* world's first website goes live at Info.cern.ch. The first web page is http://info.cern.ch/hypertext/WWW/TheProject.html
1991	*On August 6,* The *World Wide Web* is launched publicly as Berners-Lee publishes details of the project on the internet.
1993	*On April 30,* CERN releases the World Wide Web source code and announces it will be available free of charge. It also releases a basic browser along with a library of code. By the end of the year there are more than 500 web servers, and the web accounts for 1% of Internet traffic.

Figure 10.1. shows you the world wide web timeline.

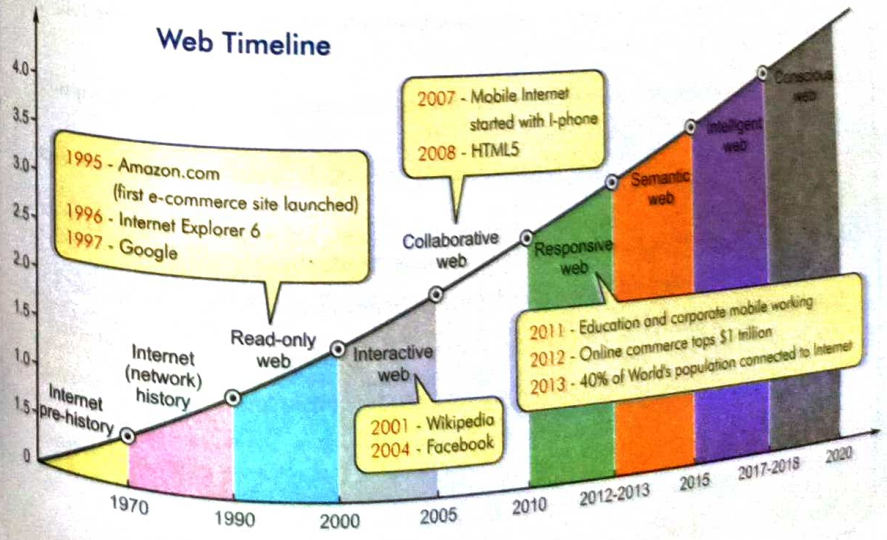


Figure 10.1 World Wide Web Timeline

10.3.2 Difference between WWW and Internet

Most people think that WWW is Internet and vice versa, which is NOT TRUE.

A large part of the Internet is the 'World Wide Web" (WWW). This is made up of web sites that have one or more web pages. This is the part of the Internet that you probably use most of the time.

But the Internet isn't just the web pages that you use every day. It also consists of :

- Forums where people can seek help and advice about a topic.
- Research databases used by universities and large companies.
- USENET – discussion groups about a specific subject from the latest boy band to favourite pets - for example *Google Groups*.
- Email and Instant Messaging (IM) systems.

It is really important to understand that the Internet is not the same as the WWW. The Internet is to the World Wide Web as Europe is to Germany. One is the container, the other is an item within the container.

NOTE

While Internet is a collection of computers or networking devices connected together ; WWW is a collection of documents, linked via special links called hyperlinks. WWW forms a large part of Internet but is not the Internet.

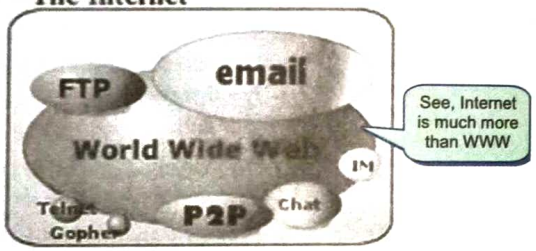


Figure 10.2 WWW *vs.* Internet

Internet Terminology

Before we proceed further, it will be good if we briefly talk about Internet terminology. In the following lines, we are introducing some basic terms related to Internet.

Client	Any computer on the networks that requests services from another computer on the Internet.
Server	Any computer that receives requests from client computers, processes and sends the output. Web Servers respond to web related request. There can be other types of servers to handle different type of requests/actions.
Webpage	Any digital page/document hosted on Internet (*i.e.*, accessible via Internet.)
Web Site	A collection of interconnected webpages, designed with a certain goal in mind.
Web Portal	A web site or service that offers a broad array of resources such as e-mail, forums, search-engines among others, *e.g.*, sites like MSN, Yahoo, Indiatimes etc.
Web Browser	A program or software on a client computer, that retrieves infor- mation from the web. Examples are : *Internet Explorer, Google Chrome, Mozilla FireFox, Netscape, Mosaic, Opera, Safari, Lynx* etc.
URL	*Uniform Resource Locator* It is the unique address of a web page on Internet.

10.4 Web Servers

A **Web server** is a computer that stores websites on the Internet and delivers web pages to viewers upon request. A **Web server** is also called ***WWW server.***

Web Servers are generally programs or software that wait for requests from the web browser. Some examples of Web servers are : *Apache, IIS, nginx, GWS, Netsite* etc.

WEB SERVER

"A Web Server is a WWW server that stores web documentes and responds to the requests made by web browsers."

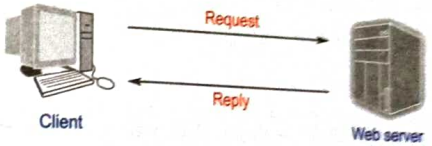


Figure 10.3 Working of a web server

Difference between a Server and a Webserver

Before we proceed further, it is important to understand the difference between a *server* and a *web server*.

Server	*Web Server*
A server is a computer or device on a network that manages network resources. A server serves requests mainly related to resources on the network such as for **sharing a resource** like printer or **sharing storage** etc. There are many different types of servers such as a *file server* or a *mail server* or a *webserver* etc. You'll learn about various server types later in this chapter.	Web servers are computers that deliver (serves up) Web pages. In technical terms, we can say that they handle HTTP[1] requests. For example, if you enter the URL **http://www.edupillar.com/index.html** in your browser, this sends a request to the web server whose domain name is **edupillar.com**. The server then fetches the page named **index.html** and sends it to your browser.

NOTE

A server is a computer that has server software loaded on it. Its main job is to share resources for clients. On the other hand, a web server is a specific protocol type of server, looking only for HTTP requests instead of many different types of resource requests.

10.4.1 Services Provided by a Server

As you already know that a server is a computer dedicated for managing network resources, it is responsible for handling the requests related to the resources on network.

Services provided by a server on a network include the following among many others :

1. **Equipment/Resource Sharing.** High quality printers and photocopiers are often designed to connect directly to a network and all authorised users can share the equipment. A server is responsible for such a resources sharing request.

2. **Centralised File Saving for Data Sharing.** Servers help the user save files to their home directories on the file server, workers can load their documents on any networked computer.

1. HTTP is HyperText Transfer Protocol; Protocols are being covered later in the chapter.

3. **Centralised Virus Scanning.** Most file servers run virus scanners constantly to detect and take care of viruses introduced by any user on any computer.

4. **Centralised Backups.** High-capacity tape devices in the file server back up all users' work, usually every night. In case of system failure (or worker failure), lost work can be recovered quickly and easily.

NOTE

The software that turn a computer into a web server, are called web server software. Examples of web server software include Apache web server, Apache Tomcat, Web server IIS, Lightpad server, Jigsaw web server, Sun Java system web server etc.

10.5 URLs

A location on a net server is called a web site. Each web site has a unique address called **URL** (Uniform Resource Locator) *e.g.*, the web site of Microsoft has an address or URL called **http://www.microsoft.com.**

Let us explore it further. The Internet structure of the World Wide Web is built on a set of rules called **Hypertext Transfer Protocol** (**HTTP**) and a page description language called **Hypertext Markup Language** (**HTML**) generally. HTTP uses Internet addresses in a special format called a **Uniform Resource Locator** or **URL**. URLs look like this :

```
type://address/path
```

where ***type***: specifies the type of server in which the file is located, ***address*** is the address of server, and ***path*** tells the location of file on the server.

For example, in the following URL

```
http://encycle.msn.com/getinfo/styles.asp
```

http: specifies the ***type*** of server, ***encycle.msn.com*** is the ***address*** of server and ***getinfo/style.asp*** is the ***path*** of the file *styles.asp*.

URL

"A location on a net server is called a **URL (Uniform Resource Locator).**"

The other examples of URLs are

```
ftp://ftp.prenhall.com, http://www.yahoo.com, news://alt.tennis etc.
```

10.5.1 Elements of URL

By now you have come to know that a URL (Uniform Resource Locator) is an address of a file on Internet. Let us discuss the components or syntax elements of URLs.

NOTE

The characters based naming system by which servers are identified is also known as domain name system (DNS).

A file's Internet address, or URL, is determined by the following :

- The type of server or protocol
- The name/address of the server on the Internet
- The location of the file on the server (this location may be related as a "path" through the file hierarchy)

The intelligent browsers like *Netscape Navigator* or *Mozilla FireFox or Microsoft Internet Explorer*, can display files in just about any format available on any of the common types of servers.

Table 10.1 lists the types of servers you may encounter, along with the protocol they use, and the type(s) of information they provide.

In any typical URL *e.g.*, **http://www.khoj.com**, the "http" identifies both the protocol and server. According to standard URL syntax, a colon (:) and two forward slashes (//) follow the protocol/server.

Table 10.1 Internet Servers and What they Provide

Server	Protocol	Information It Provides
ftp	File Transfer Protocol	Text and binary files that are organized in a hierarchical structure, much like a family tree
gopher	Transfer Control Protocol/Internet Protocol (TCP/IP)	Text and binary files that are organized in a menu structure.
http	Hypertext Transfer Protocol	Hypertext/hypermedia files (*i.e.*, multimedia documents that contain links to images, sounds, or other multi- media documents on the World Wide Web)
mail	Post Office Protocol (POP) Version 3 and Simple Mail Transfer Protocol (SMTP)	Messages sent via electronic mail
news	Network News Transfer Protocol (NNTP)	Newsgroups that are organized in a hierarchical structure

The next component of the address is the name of the server, in this case, **www.khoj.com.** Server names have multiple components. Commonly a Web server's name will begin "www" for World Wide Web. The ".com" suffix (called a *domain* indicator) indicates that Khoj is a commercial entity, as opposed to a nonprofit organization (".org"), a school or university (".edu"), a branch of the government (".gov"), etc. The naming scheme by which servers are identified is also known as the *domain name system*. The table 10.2 lists some most common domains.

Table 10.2 *Some Most Common Domains*

S.No.	Domain ID	Affiliation	Remarks
1.	com	Commercial	for commercial firms
2.	edu	Education	for educational firms
3.	gov	Government	for government organizations / bodies
4.	mil	Military	for Military
5.	net	Network resources	for ISPs / networks
6.	org	Usually non-profit organizations	for NGOs and other no-profit organizations
7.	co	Company	for listed companies
8.	tv	Television	for TV channels and companies

Some country abbreviations are being listed below :

au	Australia
ca	Canada
dk	Denmark
fr	France
in	India
jp	Japan
nz	New Zealand
uk	United Kingdom

In addition to it, a two letter abbreviation indicating the country name may be used *e.g.*,

```
http://www.microsoft.co.in
```

here the last **in** suggests that it is based in India (.in). Similarly, the URL **http://www.clearnet.nz** suggests that it is based in New Zealand (.nz).

10.5.2 URL Types

URLs give the address of files storing webpages or other documents like an image or a pdf or a doc file etc. While using a URL in a webpage's code, you can use URLs of following *two* types :

1. Absolute URLs

These are URLs that contain both the domain name and directory/page path. An absolute URL typically comes with the following syntax :

```
protocol://domain/path
```

For instance, if it is related to our **About Us** page's link of **Edupillar.com**, the URL should be mentioned as **http://www.edupillar.com/aboutus.aspx** that gives the complete information about the file location path.

Please note that the protocol may be
http://, https://, ftp://, gopher://

NOTE

An absolute URL gives complete location information. It begins with a protocol like http:// and continues, including every detail.

For web browsing, absolute URLs are typed in the address bar of a web browser.

2. Relative URLs

These are URLs with just the path excluding the domain name.

For example, **category/file.htm** or **/category/file.htm**.

Relative means "in relation to," and a relative URL tells a URL location in terms of the current location. Relative path is used to reference a given link of a file that exists **within the same domain.** This, however, doesn't give the full information about the file location but just gives a reference of the file.

Suppose a web programmer is setting up a website and links an image called "picture.jpg" as follows :

```
<IMG src = "picture.jpg">
```

It would internally look at it as :

It would internally be interpreted like this :

```
<img src="./picture.jpg ">
```

NOTE

Relative URLs are used for the files/documents being used as content in the webpage, *e.g.*, image files, doc files or pdf files etc.

Here the dot (.) before the / in the **src** attribute is a "special character" which means "start at the current directory to find a file that follows the slash(/) following the dot (.) . That is, pick up the file namely *picture.jpg* from the current folder wherefrom this webpage's file (.html file) is being picked.

10.6 WebSite

A web site is a collection of web pages which are digital files generally written using *HyperText Markup Language (HTML)*. For a web site to be available to everyone in the world at all times, it must be stored or "hosted" on a computer that is connected to the Internet round the clock. Such a computer is known as a ***Web server***.

The web pages of a website are linked together (through hyperlinks and hypertext) and share a common interface and design. The site might also contain additional documents and files such as images, videos or other digital assets.

WEBSITE

"A website or a site is a group of related web pages hosted on a web-server and are linked together and share a common interface and design."

With Internet invading in every sphere, we see websites for all kind of causes and purposes. So, we can also say that a website can also be thought of as a digital environment capable of delivering information and solutions and promoting interaction between people, places, and things to support the goals of the organization it was created for.

10.6.1 Components of a Website

As you know that a website is a collection of a related webpages hosted on a webserver. There are some components essential for making a website.

These are being listed below.

Webhost Hosting is where a website is physically located. Group of linked webpages qualify to be called a website **only when hosted on a webserver**. On a webserver, somewhere, are a set of files that are transmitted to user computers when they specify the address of the website.

Address This is the address of the website (also called URL of the site). When someone asks to see a website, (s)he needs to put this address into the web browser, and the asked site is delivered by the webserver.

Homepage Every website has a ***home page***. It is the **first web page that appears** when viewers go to a website. Home page of a website is very important as it sets the look and feel of the website and directs viewers to the rest of the pages in the website.

Design It is the overall look and feel the website has as a result of proper use and integration elements like *navigation menus, graphics, layout* etc.

Content All the webpages contained in the website together make up the content of the website.

The Navigation Structure

The navigation structure of a website is the order of the pages, the collection of what links to what. Usually it is held together by at least one navigation menu.

NOTE

We have given above the components of website, not a webpage. Often people mistakenly consider *components of webpage* as *components of a website*. But there is a difference, which will be clear to you when we cover later the components of a webpage.

10.7 Web Pages

A ***webpage*** is a digital document that is linked to the *World Wide Web* and viewable by anyone connected to the internet having a web browser. It can contain any type of information, such as *text, colour, graphics, animation, video* and *sound* etc.

10.7.1 Components of a Webpage

As you already know that a webpage is a digital document containing information in digital form, still we are asking this question – What does it contain or What are its components ? Well, this question can be answered in move than one way :

- Components of a webpage, **Content wise**
- Components of a webpage, **Structure wise**

Content Wise

Content wise the components of a webpage are : *Hypertext* and *Hyperlinks*

Hypertext refers to a digital text, which is more than just text as it can include information in various media formats such as :

- text
- graphics
- sound
- images
- video
- hyperlinks

HYPERTEXT OR HYPERLINK

"Hypertext refers to the combination of text, graphic images, audio & video tracks, and hyperlinks. A Hyperlink refers to a dynamic link upon clicking at which a new web page or program opens."

Hyperlink refers to a link from a hypertext file to another such file. A *hyperlink* can be in the form of a ***graphic*** or ***text***, upon clicking where the linked document opens up.

Following figure (Fig. 10.4) shows you hypertext and hyperlinks in a webpage.

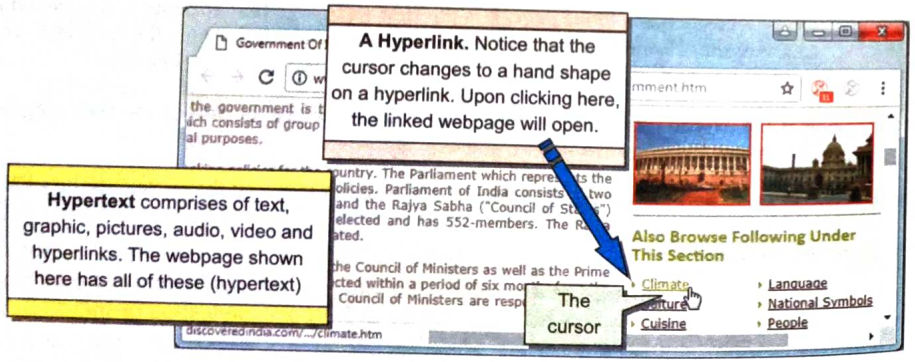


Figure 10.4 Hypertext and hyperlink.

Structure wise

Structure wise the components of a web page are :

- Page Title
- Header
- Body of the web page
- Navigational Links
- Footer

Page Title	This is a single line text which is displayed on the title bar of the browser displaying web page. (Fig. 10.5)
Header	This is generally a one or two line text (sometimes a graphics/image) defining the purpose of the web page. It is displayed at the top of the web page, below the address bar of the browser. (Fig. 10.5)
Body of the Web page	This is the section below the header of the web page and it contains the actual content of the web page (Fig. 10.5)
Navigational Links	These are the hyperlinks placed on the web page using which you can move the linked web pages/documents (Fig. 10.5)
Footer	This is the bottom section of the web page. This is the section where usually the copy right notice, website contact information etc. is put. (Fig. 10.5)

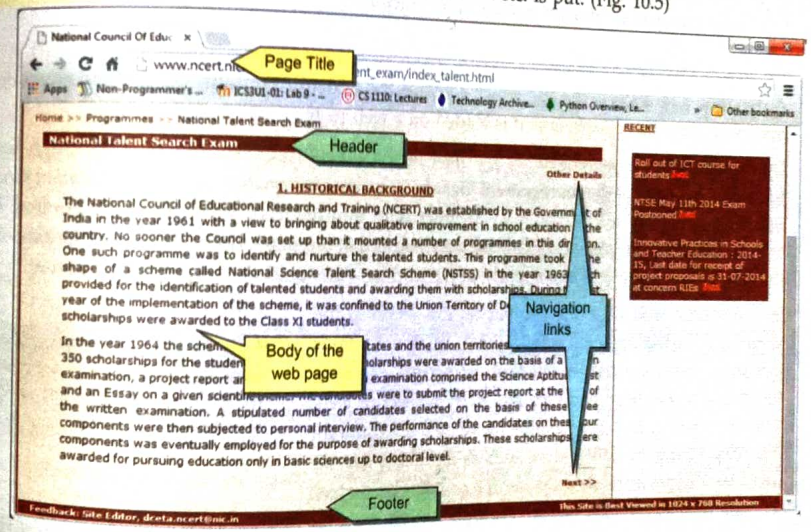


Figure 10.5 Components of a web page.

10.7.2 Types of Web Page

A web page may be a **static web page** or a **dynamic web page** depending upon what type of content it stores. A web page may contain some content which remains just the same throughout or some content that changes with time or place or type of visitor etc. Based on this property, the content of a web page can be of these types :

- Static Web Page
- Dynamic Web Page

10.7.2A Static Web Page

A static web page is the one which contains only the **static content** *i.e.*, fixed content. The static content is the one which once written and published through a webpage, remain the same every time a user views it. The static content *i.e.*, the static web pages are written via HTML code.

The **advantages** of static content are :

- It is the fastest and most efficient way to deliver content.
- It takes minimum of resources *i.e.*, it does not require any code to execute or any databases to be accessed.
- Since it uses minimum of resources, it the most secure way to deliver content.
- It is compatible with every type of webserver technology.

However, it also has the following **disadvantages** :

- It must be republished when it changes, or the viewers will not see the updates.
- It cannot display differently to different viewers, depending on their login status or other factors.

STATIC WEB PAGE

> A **static web page** is a web page which has a fixed content and thus it gets delivered to user's browser exactly as created.

10.7.2B Dynamic Web Page

A **dynamic web page** is the one which contains the **dynamic content**. Dynamic content does not remains the same everytime it is loaded on a web page.

Dynamic content is **generated** for the user at the time(s) requests the page through some scripts[2]. Examples of dynamic content include *the weather information* of a city or *location specific time* or *user specific information* (*e.g.*, member only information). With dynamic content, the document/ webpage you view exists only for you at that moment; if viewed by someone else at the same time, or by you at a slightly different time, you could get something different.

In a dynamic web page the content on the server is the same, but instead of just being HTML, it also contains **dynamic code** (written in the form of **scripts** *i.e.*, code written in a scripting language like PHP, Javascript etc.). This dynamic code is responsible for displaying different data depending on information such as *the time of day, the user who is logged in, the date, the search term* it has been given to look for. Let's look at an example — navigate to *www.flipkart.com* in your web browser, and search for 5 different products. Flipkart hasn't sent you 5 different pages; it has sent you the same page 5 times, but with different dynamic information filled in each time. This different information is kept in a database, which pulls up the relevant information when requested, and gives it to the web server to insert into the dynamic page.

DYNAMIC WEB PAGE

> A dynamic web page is a web page which displays different content (generated through some scripts) everytime it is loaded.

SCRIPTS

> Scripts are embedded codes in a web page to generate the dynamic content.

Dynamic content is good for :

- pages whose content changes too quickly to easily republish it *e.g.*, pages showing weather or latest news etc.
- pages that display viewer-specific content *e.g.*, user profile/home page of an online university student searched – items on an e-shopping site.
- pages that display content conditionally (*i.e.*, member-only pages)

2. A type of embedded program, either run at the server end or at the user's browser.

Following Fig. 10.6 shows you a webpage with dynamic content.



Figure 10.6 Webpage with dynamic content

However, dynamic content has the following **disadvantages** :

(*i*) Consumes more resources as compared to static pages.
(*ii*) Slower loading than static pages
(*iii*) Require more security features than static pages.

Static vs. Dynamic Web Page

Let us summarize the difference between these two content types.

S.No.	Static web page	Dynamic web page
1.	The content on the server is served up to any visitor the same, every time.	The content on the server is the same, but displays different data depen- ding on information such as *the time of day, the user who is logged in, the date, the search term* it has been given to look for.
2.	Contains simple HTML code	Along with HTML code, also contains dynamic code (in form of scripts)
3.	Requires no additional special software	Special software must be installed on the server to create a dynamic web page

10.7.3 Difference between a Webpage and Website

A website is composed of a group of webpages linked together. A website refers to a central location on a webserver that contains more than one web page. A website may have just a home page, or it may have many other internal webpages in addition to the home page.

For example, *Edupillar.com* is considered a website, which contains many webpages. However, a single webpage on this site would be its *home page* or *AboutUs page* and so on. The complete website contains many webpages like :

- Home page
- Content Demo page
- About Us page
- Register page
- FAQ page
- Contact Us page

Each of the above mentioned are individual webpages but together these all make the website *edupillar.com*

NOTE

A website may have just a home page, or it may have many other internal webpages in addition to the home page.

10.8 Web Browsers

A web browser is a program that lets you visit different sites on the Net and display their offerings on your own computer. You visit a site by supplying the browser with an address, or URL.

SERVER

> "A web browser is a program that lets you visit different sites on the Net and display their offerings on your own computer."

Specifically, browsers are intended to let you access resources on World Wide Web – the hypertext files discussed earlier in this chapter. However, you can use a browser to access other types of file systems as well.

There are many browsers available in the market *e.g, Google Chrome, Mosaic, Netscape, Internet Explorer, Safari, Qpera, Macweb* etc. but only few of them such as *Netscape Navigator, Google Chrome, Mozilla Firefox and Microsoft Internet Explorers (MSIE)* appear just about everywhere. Though, MOSAIC (developed by NCSA – *National Centre for Supercomputing Application*) was the first web browser to come into market.

10.8.1 Web Browser Toolbar

The toolbar of a browser is a bar within a browser's window, generally at the top and which offers many functions through buttons that can be performed within a web browser.

Major functions that are offered through a browser toolbar include generally the following :

- **Back** : Open previously opened page
- **Forward** : Move to the Page from where you have come to present page using Back button.
- **Home** : Open webpage whose URL is stored as the home page
- **Refresh** : Reload the current webpage
- **Search bar with suggestions** : Type search text
- **History** : List of webpages opened so far
- Links to news and other vertical search options
- Ability to save pages to favorites
- Pop-up blocker
- Some Optional features such as the ability to use translate, vote on pages and add other custom buttons etc.

Each web browser offers mostly all of the other mentioned functionality and even more.

Following figure shows some toolbars of different browsers :

Mozilla Firefox browser toolbar



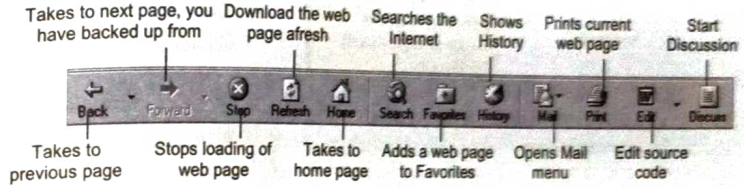
Netscape Navigator browser toolbar



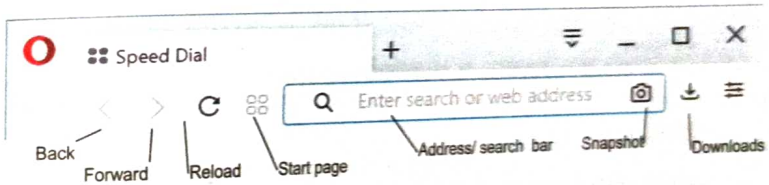
Google Chrome browser toolbar



Standard button toolbar of MSIE



Standard toolbar of Opera



Commonly used Web Browsers

There are many web browsers in use today. The most common of these are : *Mozilla FireFox, Google Chrome, Opera, Safari, Internet Explorer, Torch, Maxthon, Sea Monkey, Avant Browser, Deepnet Browser* etc.

10.8.2 Browser Settings

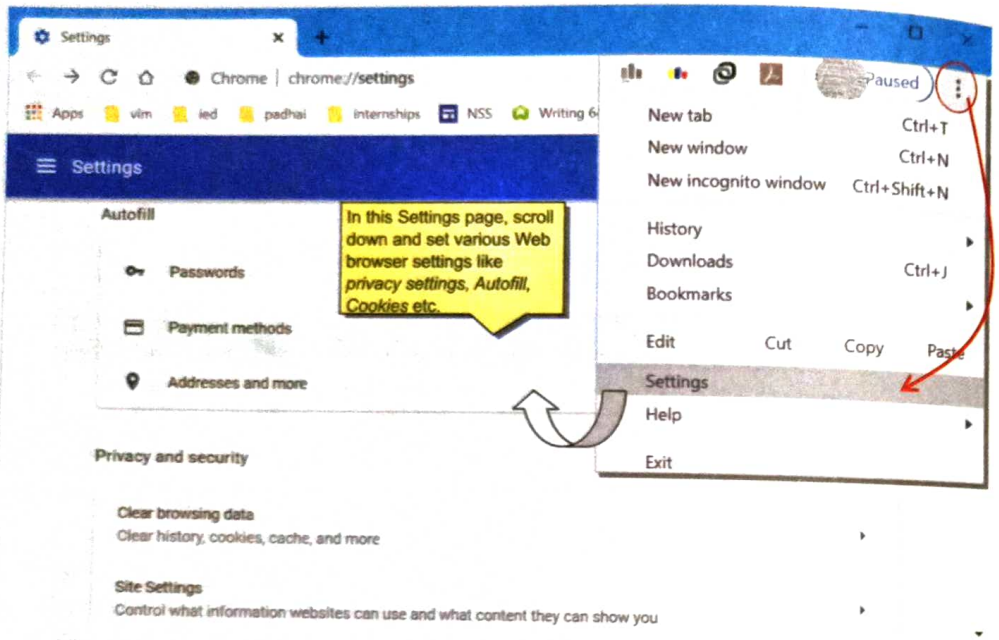
You can make changes in your web browser settings for things like *privacy settings, search engine preferences, autofill and autocomplete behaviour, cookies* and so forth. For any type of browser settings, you need to open the **Settings page** for your web browser as discussed below.

You can open ***Settings*** in different web browsers as illustrated below.

1. Open your web browser.
2. We are discussing here how to make changes in different browsers.

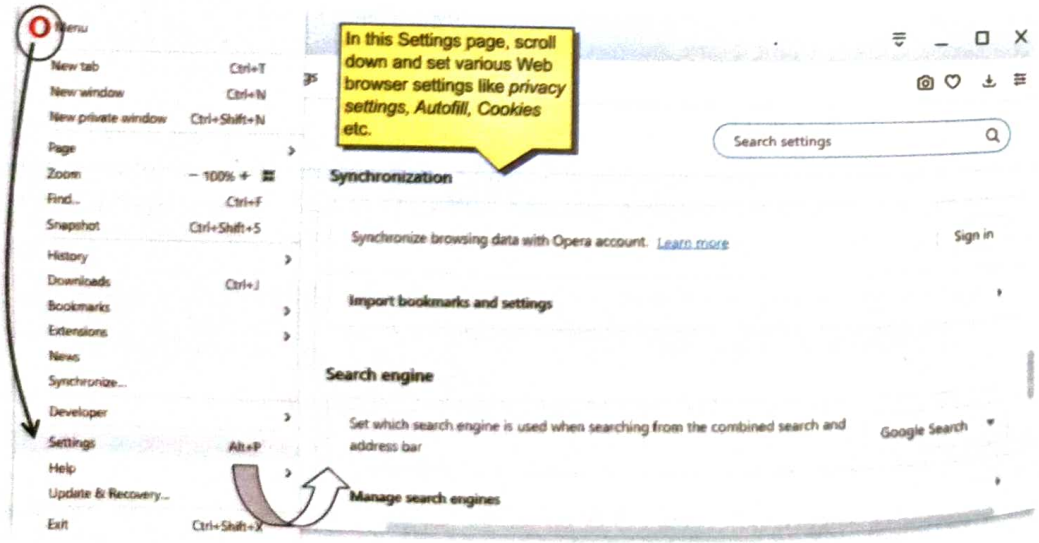
Google Chrome :

In the upper-right corner, click on Google Chrome triple dots (⋮) icon and then click on the **Settings** option in the menu.



Opera :

In the upper-left corner, click the **Menu** button ( ) and from the menu that appears, click on the **Settings** option.



Similarly, you can open the settings pages in other web browsers and choose the settings for your web browsers.

Mozilla Firefox :

In the upper-right corner, click the **Open menu** ( ) icon and click on **Options** in the menu.

Safari :

In the upper-left corner, click on **Safari** in the menu bar and click on the **Preferences** option.

Microsoft Edge :

In the upper-right corner, click the **Settings and more** ( ) icon and click on the **Settings** option in the menu.

1. Settings for Cookies

Cookies are small text files stored on a user's computer and created and used by websites to remember basic information or to record the user's browsing activity. You can enable or disable cookies through *browser settings*.

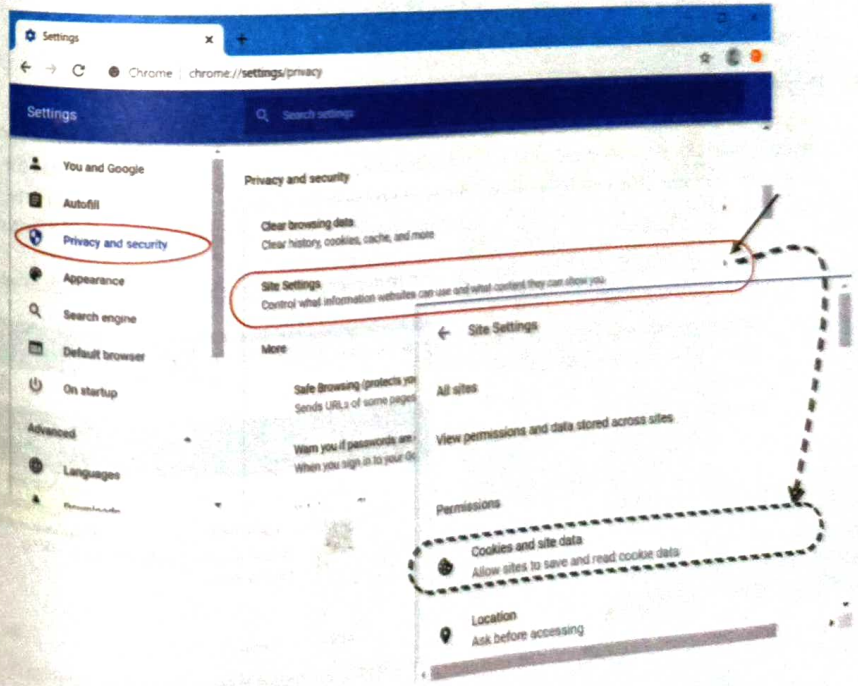
COOKIES

"Cookies are small text files stored on a user's computer and created and used by websites to remember basic information or to record the user's browsing activity."

For enabling or disabling cookies settings, you can do the following :

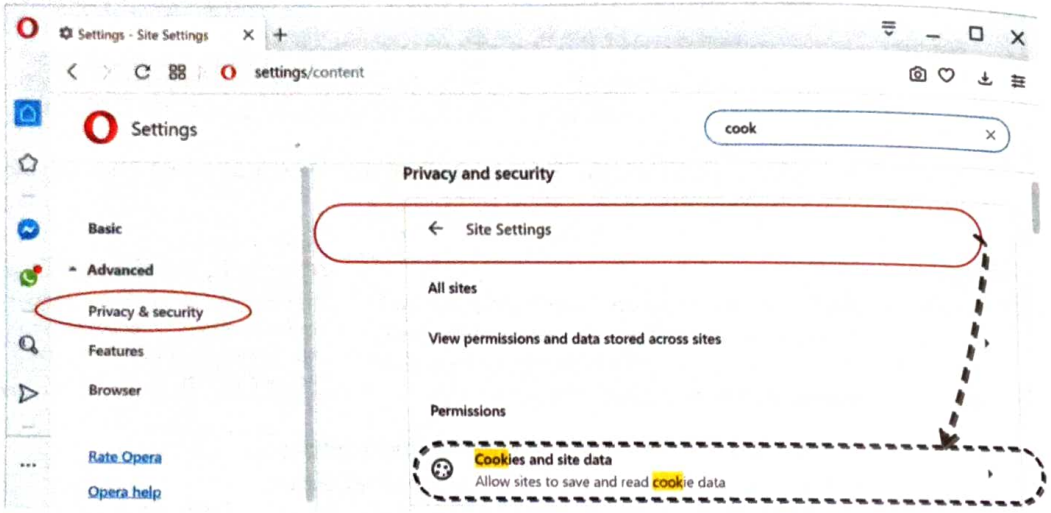
1. Open **Settings** *page* of your web browser as explained earlier.
2. Look for **Privacy and security settings**. If not directly found, click on **Advanced settings**. In **Site settings**, you can find the **cookies** setting option. Click on this and choose desired cookies settings.

   For example, in ***Google Chrome***, you can find it in
   Privacy and security settings → Site Settings → Cookies and site data



In *Opera browser*, you can find it in

Privacy & security → Site Settings → Cookies and site data



II. Permissions

For enabling/disabling ***Javascript***, ***Flash*** etc. for other types of permissions on your web browser :

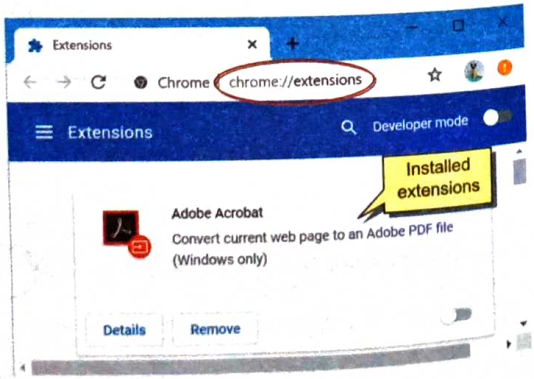
1. Open your web browser
2. Choose ***Privacy and Security*** settings
3. Go to ***Site settings*** (as you have done earlier ; see above figure)
4. And choose desired settings for *Javascript, Flash, Pop up blocks, Adblocker* etc.

III. Plugins or Add ons

A **plug-in** (also called ***plugin***, ***add-on***, or ***extension***) is a software component that adds a specific feature to an existing computer program such as a web browser. **Plugins** or **extensions** for web browsers enhance the functionality of your web browser by adding a variety of features to it, *e.g.*, enabling emoticons, reading pdfs, changing your browsing location, adding to do lists etc.

TO VIEW INSTALLED PLUGINS on your web browsers, look for ***Extensions*** and click on it.

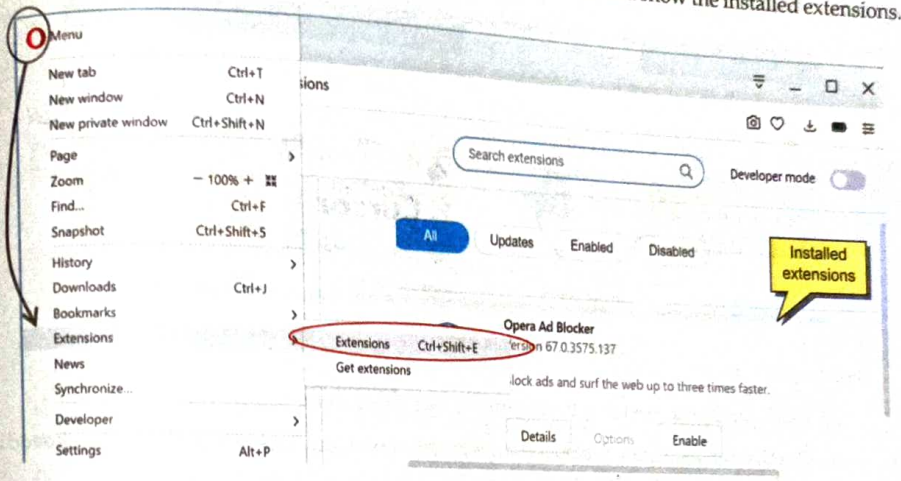
- On ***Google Chrome***, it is found in **Advanced settings**. You may also type chrome://extensions/ in the address bar. It will show you installed extensions.



PLUG-IN

"A **plug-in** (also called ***plugin***, ***add-on***, or ***extension***) is a software component that adds a specific feature to an existing computer program such as a web browser."

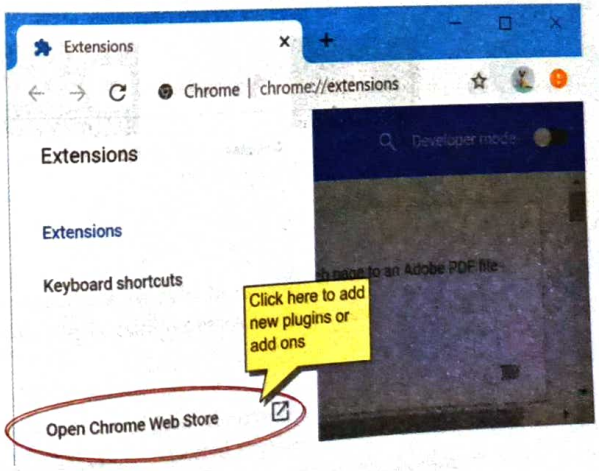
- On **Opera**, click on **Opera menu** [O] icon on the top left and click **Extensions → Extensions** or press ***Ctrl+Shift+E*** and it will show the installed extensions.



TO ADD A PLUGIN to your web browser, you need to look *for extensions store*, which for example on *Google Chrome*, it is **Chrome web store** and for *Opera*, it is **Opera Extension Store**.

For Google Chrome,

(*i*) open extensions by typing **chrome://extensions** in the address bar and from the side bar, click at **Open Chrome web store** at the bottom.



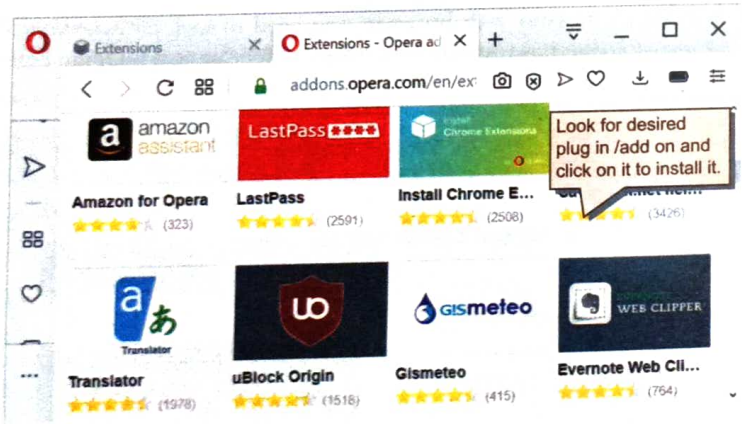
(*ii*) It will open ***Chrome web store***, wherefrom choose the desired plug in/add on and clik on it to install.



For Opera browser,

(*i*) Click on **Opera menu → Extensions → Get Extensions** command.

(*ii*) It will open the *Opera extension store*, wherefrom you can look for desired plug in/add on and click on it to install it.



10.9 Internet Applications

The spread and popularity of Internet is because of diverse applications and services the Internet offers. In this section, we shall talk about some popular applications of the Internet.

10.9.1 E-mail (Electronic Mail)

The most widely used tool on the Internet is electronic mail, or *e-mail*. E-mail is used to send written messages between individuals or groups of individuals, often geographically separated by large distances. E-mail messages are generally sent from and received by mail servers – *computers that are dedicated to processing and directing e-mail*. Once a server has

E-MAIL

> "E-Mail stands for Electronic Mail. It is the most widely used tool to send messages electronically (*i.e.*, through computers) on a network."

received a message, it directs it to the specific computer that the mail is addressed to. To send e-mail, the process is reversed. A very convenient and inexpensive way to transmit message, e-mail has dramatically affected scientific, personal, and business communications.

10.9.1A E-mail Account and Address

Although sending and receiving mail is extremely simple. However, there are a few parameters you must be clear about before you start sending and receiving mails. One such parameter is E-mail address. E-mail addresses commonly take this form :

`username@hostname`

Here are a number of typical e-mail addresses (well, almost typical) :

`pacasa@hotmail.com`, `paca@del6.vsnl.net.in`,
`bala_gm@india.com`, `madduRma@indya.com`,[3]
`Wada.Sachin@InfoMeet.com`

Here you see that before the character '@' (pronounced "at" or "at the rate of") the e-mail account holder's net name appears and after '@', the name of Net server appears which hosts this e-mail address. That is, in address *Wada.Sachin@InfoMeet.com*, *Wada.Sachin* is the net name of a user and *InfoMeet.com* is the name of server hosting it.

Similarly, in email-id *pacasa@hotmail.com*, *pacasa* is the net name of the user and it is hosted on server *hotmail.com*. Thus we see that an email-address has *two* parts separated by '@' ("at") symbol.

(*i*) ***Username.*** On the left side of @ separator is the user name. A user name can not have blanks.

(*ii*) ***Domain name for the host server.*** The portion to the right of @ identifies the server or host or network that services your e-mail. It is sometimes also called the *e-mail server*.

Advantages of E-mail

- Low cost
- Speed
- Waste reduction
- Ease of use
- Record maintenance
- Patience

Limitations of E-mail

- Hardware requirement
- Impermanent
- A hasty medium
- Hard to convey emotions

E-mail Format Types

Plain text (.txt)

- Can be read by anyone, regardless of the e-mail application they use.
- Does not offer text formatting options.
- Can have pictures or graphics attached, but not inset.

Rich text format (.rtf)

- Allows you to format a message for font, justification, stylized text and other options.
- Pictures or graphics (such as smileys) can be inserted.
- Cannot be read by every type of e-mail application.

Difference between cc and bcc

Both **Cc** and **Bcc** forward a copy of the message to everyone you've listed. The main difference between **Cc** (*i.e.*, **Carbon copy**) and **Bcc** (**Blind carbon copy**) is that, with the latter, the recipients do not get to know each other.

3. Please note all the e-mail addresses mentioned here are fictitious. Any resemblance to real e-mail addresses is co-incidental.

Using BCC protects the privacy of your recipients and helps reduce spam. When you send an email to several people using the normal "To:" or "Cc;" fields, all of their email addresses are clearly displayed in the message. If the email is forwarded, the email addresses of all previous recipients may also be forwarded. However, when you use BCC to send email to more than one person, the email addresses of your recipients will not be displayed in the emails.

Spam vs. Junk Mail

Spam is defined as irrelevant or inappropriate messages sent on the Internet to a large number of people. **Junk mail** is the unwanted or unsolicited advertising or promotional material received through mail or email. These two terms typically are used interchangeably.

In the early days of the Internet it was possible to send email to everyone on the system. (Nowadays, we have lot of spam filters that save us from a lot of spam). These system-wide emails were referred to as "SPAM", meaning *Send People Alotta Mail*. Spammers (*i.e.*, the people sending the spam) send their messages to hundreds, thousands or even MILLIONS of email addresses at once with the hopes that at least a few people will respond.

10.9.2 Protocols used in Email

Basically, a protocol refers to standard set of rules used at each end of a communication channel, in order to properly transmit information. In order to deal with your email you must use a mail client to access a mail server. The mail client and mail server can exchange information with each other using a variety of protocols. The protocols mainly used with email accessing are *IMAP, POP3, SMTP* and *HTTP*. Let us briefly talk about these.

IMAP — **IMAP** (*Internet Message Access Protocol*) is a standard protocol for accessing e-mail from local server. IMAP is a client/server protocol in which e-mail is received and held for the user by Internet server. As this requires only a small data transfer this works well even over a slow connection such as a modem. Only if the user requests to read a specific email message, then it will be downloaded from the server. You can also create and manipulate folders or mailboxes on the server, delete messages etc.

POP3 — The **POP** (*Post Office Protocol 3*) protocol provides a simple, standardized way for users to access mailboxes and download messages to their computers.

When using the POP protocol, all the eMail messages get downloaded from the mail server to user's local computer. The user can choose to leave copies of his/her eMails on the server as well. The advantage is that once messages are downloaded, the user can cut the Internet connection and read his/her eMail at own leisure without incurring further communication costs. On the other hand he/she might have transferred a lot of message (including spam or viruses), which may prove dangerous for data on PC.

SMTP — The **SMTP** (*Simple Mail Transfer Protocol*) protocol is used when you send email to another email users(the recipient). The SMTP protocol is used by the Mail Transfer Agent (MTA) to deliver the sent eMail to the recipient's mail server. The SMTP protocol can only be used to send emails, not to receive them. Depending on the network / ISP settings, user may only be able to use the SMTP protocol under certain conditions

HTTP — The **HTTP protocol** is not a protocol dedicated for email communications, but it can be used for accessing mailbox. Also called web based email, this protocol can be used to compose or retrieve emails from an email account. *Hotmail, Gmail, Ymail, Rediffmail* etc are good examples of using HTTP as an email protocol.

10.10 Chat

Chatting is the most fantastic thing on Internet. Chatting is like a text-phone. In telephonic conversations, you say something, people hear it and respond, you hear their responses on the spot and can reply instantly. In the same manner, in chatting, you type a message on your screen, which is immediately received by the recipient ; then the recipient can type a message in response to your message, which is received by you instantly.

If you want to chat, you need to log on to a server that provides chat facility. There are many sites that provide chat facility.

These sites mostly provide you chat facility via *chat rooms*.

CHAT

"Online textual talk is called Chatting."

Chat room A chat room is a website or a part of some online service that provides an online venue for communities of users with a common interest to communicate in real time (instantly).

10.10.1 Commonly used Type of Chat

Chat can be of different types. Given below are some commonly used types of chat.

(i) Instant Messaging (IM)

It is a type of communications service via *IM software* that enables one to create a kind of private chat connection with another individual in order to communicate in real time over the Internet, similar to a telephone conversation but using text-based, not voice-based, com- munication.

The most popular form of IM is *Instant Messengers* such as ***AOL Messenger***. Messenger software allows anyone who downloads their free software to choose a nickname and 'chat' with anyone else with the same messenger software directly. Even programs like *Skype* also offer such a facility.

(ii) ICQ ("I seek you")

ICQ offers chatting via ICQ software. It is used as a conferencing tool by individuals on the Net to chat, e-mail, perform file transfers, play computer games, and more.

When you download ICQ program (which is free) you are assigned an ICQ number. Two or more people using ICQ can have the same nickname, but no two people can have the same ICQ number.

(iii) IRC – Internet Relay Chat

Unlike IM and ICQ, IRC (Internet Relay Chat) is not owned by any company. To use IRC, you need an *IRC client program*. IRC has many networks that are completely separate from one another. If you want to chat with someone on IRC, you need to make sure that you are connected to an *IRC server* via your PC's *IRC client program*.

Once you connect to an IRC server, there are often thousands of rooms (called ***channels***) on nearly any topic, and tens of thousands of different people to chat with.

(iv) Web based Chat

Web based chat is also like IRC but it is different from it in the sense that it's on a specific website and no program is really needed to install on computer. *Yahoo Chat* is a great example of a web based chat.

0.11 Voice Over Internet Protocol (VoIP)

VoIP (Voice over Internet Protocol) is a technology that makes audio and video calls possible from any internet-connected device having a microphone and speakers. You must have made calls using apps like *WhatsApp, Telegram, FaceBook Messenger, Skype* etc. These software/apps use VoIP to make such calls over Internet. On the Internet, communication is the responsibility of a protocol called **IP** (Internet Protocol). VoIP allows for the transmission of voice along with other data being communicated over IP, hence the name – **Voice over Internet Protocol**. VoIP takes audio/voice input (which is of analog type) and converts them into digital information so that it can be transferred over Internet along with other data on IP.

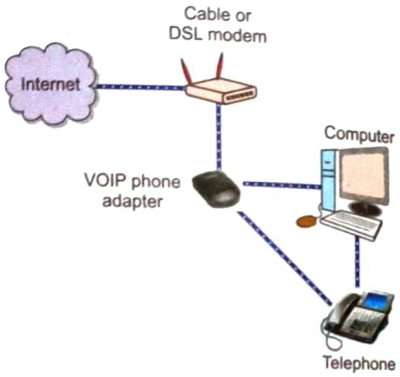


FIGURE **VoIP**

Examples of many VoIP based software/apps are *Skype, Google Duo, Meet, Zoom, WhatsApp, telegram, FaceBook Messenger* etc.

There are *two* major reasons to use VoIP :

- Lower Cost
- Increased functionality as it can use any Internet enabled devices

Advantages

1. It does not cost extra as long as you have a device with an Internet connection and are using it for personal use. Of course there are paid versions available for professional requirements.
2. *Easily available.* Many VoIP based software such as *Skype*, are easily downloadable.

Disadvantages

1. VoIP is possible if both the caller and the receiver have the right software and hardware to speak to one another
2. There is no guarantee of call's audio and video quality when using the Internet.
3. It uses Internet Protocol's packets so if packets get dropped along the way then voice quality drops.
4. Even with its high quality, it cannot match up the voice quality of a normal telephone line.
5. It may not work for emergency service numbers like Police or Fire Services' regular numbers.

Check Point 10.1

1. What is Internet ?
2. How is WWW different from the Internet ?
3. What is the role of a web server ?
4. What are the full form of these ?
   (*i*) WWW (*ii*) URL (*iii*) IMAP
   (*iv*) POP3 (*v*) SMTP (*vi*) HTTP
   (*vii*) VoIP
5. Name some E-mail protocols ?
6. What is VoIP ?

LET US REVISE

- *Internet is a network of many small networks that exist all over the world.*
- *The World Wide Web is a set of programs, standards and protocols that allows the multimedia files to be created, displayed and linked on the Internet.*
- *A server is a computer that has server software loaded on it. Its main job is to share resources for clients.*
- *A web server is a specific protocol type of server, looking only for HTTP requests instead of many different types of resource requests.*
- *A website or a site is a group of related web pages hosted on a webserver and are linked together and share a common interface and design.*
- *A document that uses HTTP is called a web page.*
- *The top-level web page of a web site is called home page.*
- *Sending and receiving messages electronically is called e-mail.*
- *An email address has two parts : username and domain name separated by @("at") symbol.*
- *Advantages of e-mail are : low cost, waste reduction, speed, ease of use, record maintenance etc.*
- *Online textual talk is called chatting.*

Objective Type Questions OTQs

Multiple Choice Questions

1. A web-site is a collection of
   (*a*) HTML documents (*b*) Graphic files
   (*c*) audio and video files (*d*) all the above
2. The first page that you normally view at a web site is its :
   (*a*) Home page (*b*) Master page (*c*) First page (*d*) Banner page
3. When you visit a website, it will open the page set as _____ of the website.
   (*a*) Web Site (*b*) Home page (*c*) Email (*d*) ISP ?
4. The first page on a website that allows you to navigate to other pages via menu system or links, is known as
   (*a*) front page (*b*) primary page (*c*) home page (*d*) first page
   (*e*) main page
5. IP stands for
   (*a*) Internet Protocol (*b*) Intranet Protocol
   (*c*) Internet Provider (*d*) Internet Package
6. URL stands for
   (*a*) Universal Resource Locator (*b*) Uniform Resource Location
   (*c*) Uniform Resource Locator (*d*) None of the above.
7. Which of these is not a part of URL ?
   (*a*) IP address (*b*) Port Number (*c*) Domain Name (*d*) None of these

8. Which portion of the URL below records the directory or folder of the desired resource ?

   http://www.somestore.com/firstfloor/shoes.htm

   (*a*) http (*b*) firstfloor
   (*c*) shoes.htm (*d*) www.somestore.com

9. In URL, http://www.pacein.com/pr/main.htm, which component identifies the protocol or type of the server ?

   (*a*) http (*b*) www.pacein.com
   (*c*) /pr/main.htm (*d*) none

10. In URL, http://www.pacein.com/pr/main.htm, which component identifies the web-site ?

    (*a*) http (*b*) www.pacein.com
    (*c*) /pr/main.htm (*d*) none.

11. In URL, http://www.pacein.com/pr/main.htm, which component identifies the path of a web page ?

    (*a*) http (*b*) www.pacein.com
    (*c*) /pr/main.htm (*d*) none.

12. URLs are of two types :

    (*a*) Absolute & Relative (*b*) Static & Dynamic
    (*c*) Absolute & Dynamic (*d*) None of the above

13. An email can be sent via,

    (*a*) PC (*b*) Tablet
    (*c*) Smart phone (*d*) All of the above

14. In an email address, the characters following '@' character represent

    (*a*) User name (*b*) E-mail Recipient
    (*c*) Domain name (*d*) None of these

15. _____ field of email-compose box allows to send same message to multiple recipients without getting them know about other recipients.

    (*a*) cc (*b*) To (*c*) bcc (*d*) Message

16. Chatting on Internet is

    (*a*) Talking face to face (*b*) Talking on phone
    (*c*) Online textual talk (*d*) None of these

17. A _____ room or online forum allows users to communicate through typed messages which are displayed almost instantly.

    (*a*) Mail (*b*) Buddy (*c*) Chat (*d*) Message.

18. Online textual talk is called _____ .

    (*a*) Video Conferencing (*b*) Telephony
    (*c*) Text Phone (*d*) Chat

19. Electronic mail cannot be used for

    (*a*) Sending documents
    (*b*) Sending bulk data (say over 2GB)
    (*c*) Sending attachments
    (*d*) Sending messages

Fill in the Blanks

1. The Internet is most often used for the exchange of _____ and _____.
2. Web pages on a site are linked together through a system of _____ .
3. The address of location of the document on the world wide web is called its _____.
4. The _____ is a computer based worldwide communication network.
5. The _____ is a component of the Internet that allows exchange of multimedia based and hyperlinked content.
6. Full form of WWW is _____ _____ _____.
7. A digital document hosted on a website is a _____ _____.
8. The landing page of a website is _____ _____.
9. A webpage whose content is generated every time it is loaded is called a _____ webpage.
10. A webpage whose content is fixed and appears the same every time it is loaded, is called a _____ webpage.
11. A _____ _____ is a specialised computer which serves the requests made by web browsers.
12. Full form of URL is _____ _____ _____.
13. Full form of cc in the content of email is _____ _____.
14. Full form of bcc in the context of email is _____ _____ _____.
15. Full form of IP is _____ _____.
16. Full form of VoIP is _____ _____ _____ _____.
17. Website store the browsing activity through _____.

True/False Questions

1. The Internet and www are the same thing.
2. The Internet is a component of www.
3. WWW is a part of the Internet.
4. The exchange of multimedia based hypertext is carried out on www.
5. A website consists of many home pages.
6. Dynamic webpages are written in plain HTML.
7. The dynamic code of dynamic webpages is written in the form of **scripts.**
8. The dynamic websites are faster than static websites.
9. Websites can't store basic information or user's browser activity.
10. Cookies are stored on the webserver.
11. Cookies are stored on the user's computer.
12. You cannot place audio calls on the Internet.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *What is Internet ?*

Solution. Internet is a computer-based world wide information network. The Internet is composed of a large number of smaller interconnected networks.

On Internet, each connected individual can communicate with anyone else on the Internet, can publish ideas, and can sell products with a mini- mum overhead cost.

2. *Mention some uses of Internet.*

Solution.

(*i*) Businesses use Internet to carry out electronic commerce.

(*ii*) Internet is used for voice and video confe- rencing and other forms of communication.

(*iii*) E-mail on Internet has greatly speeded up the communication among companies and among individuals.

(*iv*) Internet is also used for communication, entertainment, finding information, reaching customers and for providing many types of services.

3. *What do you understand by term URL ?*

Solution. **URL.** The full form of URL is Uniform Resource Locator. A URL refers to an address on the Internet. Billions of documents/multimedia files can be accessed on Internet through their URLs.

4. *In what ways is the World Wide Web different from other servers on the Internet ?*

Solution. The *World Wide Web* is an Internet system *i.e.*, a set of *programs*, *standards*, and *protocols* that allows the multimedia files to be created, displayed and linked on the Internet.

Before WWW, Internet was mainly used for obtaining textual information. The other servers on Internet : *ftp*, *gopher*, *mail*, *news* etc. cater to textual information but WWW, which uses http server, caters to multimedia information. Thus, it enables the users to receive hypertext *i.e.*, multimedia information on their web-browsers.

5. *Differentiate between a web browser and web server.*

Solution. A WWW client is called a **web browser** and a WWW server is called a **web server**.

A web browser navigates through the *world wide web* and displays web pages. The web browser requests a page from a web server. The web server makes available the desired page to web browser.

6. *Explain the following terms : (i) Web page (ii) home page.*

Solution. (*i*) **Web page.** A document using http and that resides on a web-site is known as a web page.

(*ii*) **Home page.** It is the top-level web page of a web site. This is the page that gets displayed first of all when a web-site is opened.

7. *Identify domain name(s) and URL(s) from the following list :*

(*a*) yoga@yoga.com (*b*) doYogaEveryday.com

(*c*) www.doYogaEveryday.com/old/index.html

(*d*) www.PerformingArts_schools.org/backlink.htm

(*e*) 192.168.0.1

(*f*) http://cbse. nic. in

Solution. (*b*) and (*f*) are domain names. (*c*) and (*d*) are URLs.

8. *Discuss some advantages and limitations of E-mail.*

Solution. *Advantages :*

(*i*) Low cost (*ii*) Speed

(*iii*) Waste Reduction (*iv*) Ease of Use

(*v*) Record Maintenance

Limitations :

(*i*) Hardware requirement (*ii*) Impermanent

(*iii*) A hasty medium (*iv*) Hard to convey emotion.

9. *Discuss the structure of an email message in brief.*

Solution. In an email-message, apart from the message text, there are some other fields, which are used for some specific purposes as listed below :

(*i*) *To :* In this field, the e-mail address of the recipient is given.

(*ii*) *CC :* In this field, the e-mail addresses of the *Carbon Copy* recipients of the mail, are mentioned.

(*iii*) *Bcc :* In this field, the email addresses of the *blind carbon copy* recipients are mentioned.

10. *What is Chatting ? Is it necessary to be online (i.e., connected to Internet) while chatting ?*

Solution. Chat is online textual conversation wherein the message typed by the sender is instantly made available to the recipient and vice versa.

Yes, it is necessary to be online while chatting.

11. *Which of the following entires is different from others ? Why ?*

Safari, Chrome, Netscape, FaceBook

Solution. **FaceBook.** It is a social networking site while others are web browsers.

12. *What is VoIP ? How does it work ?*

Solution. VoIP is a technology that allows one to make audio and video calls over a Broadband Internet connection. Full form of VoIP is ***Voice over Internet Protocol.***

It works as follows :

(*a*) The caller speaks into a microphone that is connected to a sound card.

(*b*) The sound card has an *'Analogue-to-Digital converter'* that translates the caller's speech into a digital stream.

(*c*) That data is given a destination address and off it goes like any other data stream.

(*d*) The receiving computer has the right VoIP software to convert the incoming data back into speech.

GLOSSARY

Chat	Online textual conversation.
E-mail	Sending and receiving messages electronically.
Home Page	Top-level web page on web-site that gets displayed upon opening a web-site.
Hyperlink	A link on a web page that points to a file or document or a web page or a web-site.
Hypertext	A combination of text, audio and video tracks, and hyperlinks.
HTTP	Hypertext Transfer Protocol. A protocol used by WWW to create display and link hypertext files.

Server	A powerful computer that provides services to other computers.
URL	Uniform Resource Locator. An internet address in a special format.
VoIP	Voice over Internet Protocol ; A technology to place over the Internet.
Web Browser	A software that navigates through World Wide Web and display web pages.
Web page	A document that uses HTTP.
Web site	A location on a net server.
WWW	World Wide Web. An Internet system used for creating, displaying, linking hypertext or multimedia files.

Assignment

1. What is WWW ? How does it function ?
2. What is a server ? What is a web server ?
3. Enlist some services provided by a web server.
4. Name some web server software prominently in use.
5. Differentiate between a web page and a home page.
6. How is a web site different from that of a web portal ?
7. What are the components of a website ?
8. What are the components of a webpage ?
9. Discuss some elements of a website.
10. The first page on a website that allows you to navigate to other page via menu system or links, is known as
    (*a*) front page (*b*) primary page
    (*c*) home page (*d*) first page
    (*e*) main page
11. What is a web site ? How is it related to a web page and a home page ?
12. What do you understand by a (*i*) Web browser (*ii*) Web server ?
13. What is a URL ? What are its components ? Examplify.
14. How is an absolute URL different from a relative URL ?
15. What do you understand by http ? How is it related to world wide web ?
16. What are the *two* parts of an email address ? Which character is used to separate these two parts ?
17. Explain the structure of an email address.
18. Explain the structure of an email message. Also differentiate between *cc:* and *bcc:* fields.
19. What are plugins ? How do you install these on a web browser ?
20. What are cookies ? What is their utility ?
21. Whai is VoIP ? How is it useful ?
22. List some advantages/disadvantages of VoIP ?

11 Societal Impacts

Outline

11.1 Introduction
11.2 Digital Footprint
11.3 Net and Communications Etiquettes
11.4 Ethical Essues
11.5 Open Source Philosophy
11.6 Copyright and other Licenses
11.7 Cyber Crime
11.8 Cyber Law and IT Act
11.9 E-Waste Management
11.10 Health Concerns with Technology Usage

11.1 Introduction

We are now living an era called the information age where we see that most our activities are technology–influenced, be it making an online payment, creating or development of own piece of art or information (such as writing articles or clicking photographs and so forth). With the reach of technology to our day to day life, there has been a paradigm shift, and it has also raised specific issues and problems related to society, ethics and law. In this chapter, we shall talk about topics about this very domain such as intellectual property rights, plagiarism, cybercrime, cyberlaw, e-waste management etc.

11.2 Digital Footprint

DIGITAL FOOTPRINTS

> **Digital Footprints** are the records and traces individuals' activities as they use the Internet. Digital footprints are permanently stored.

Digital footprints are the records and traces individuals leave behind as they use the Internet.

Your interactions on social media, your friend circle on social media sites, sites you *visit, online purchases, locations visited through Facebook check-ins* etc. all make up your **digital footprints.**

A common comparison to digital footprints is the footprints you leave behind you when walk along a beach. Every step you take leaves an impression on the sandy surface that allows another person to see the marks that your activity has made and offers the possibility for them to follow your trail. Digital footprints are also referred to as "digital tattoos".

Unlike a beach footprint which will eventually fade away, the Internet – and any electronic 'event' in general – doesn't work the same way. It is safe to assume that anything you do, publish or post online may be there permanently and won't be 'washed away'.

Mistakes aren't as easy as they used to be because once we post anything online, it stays forever and cannot be undone, Digital Footprints last forever, and colleges and jobs will look back at them to see if you are what you portray and how you conduct yourself actually.

11.2.1 Managing Digital Footprint

In this age, your digital reputation is equally important, which is controlled by your digital footprint.

To manage your digital footprint, you can follow the guidelines given below :

1. **Know what your digital footprint is.** Look at all the social networking sites and forums that you belong to, and search your name to know what information about you is available.
2. **E-behave responsibly.** You should be smart and sensible enough to know which sites you are visiting, which emails you are sending, or what links you open. Also, make sure to never share your location when online. It is best to wait until you are home to share pictures of an event or activity on social media.
3. **Keep your digital footprint clean.** Carefully go through your social media handles, past browsing history on YouTube and other public sites and do the following :
   (*a*) remove any photos, content, and links that may be inappropriate
   (*b*) remove any details about you that reveal too much information like your phone number, school, college name, address etc.
4. **Control the visibility of your information.** Every web browser, social networking site offers options in their settings to control the visibility and access of your information. Rather than making every browsing activity and post of yours ***Public***, you should be selective about who you authorize to access your information. It should be limited to only your known circle or the people you can trust upon (*private*), *e.g., Friends, or Friends of Friends, or your contacts* etc.
   Thus, you should use the privacy features of your browser and of the various websites you frequent to reduce the visibility of your information.

5. **Allow Comments Moderation.** Since many comments on public websites can be publicly seen, monitor and moderate comments associated with you to maintain a positive digital footprint.
   Consider using the "**block comments**" feature or setting your social networking profile to "private" so that only designated individuals can view your information.
6. **Think before you post.** All the above mentioned steps are to control what has been done earlier. But if you are cautious and think before you post anything online, your digital footprint will be clean.

DIGITAL FOOTPRINT	
Be Careful About	**Be Smart About**
❖ What you share online ❖ Where you share ❖ With whom you share	❖ Sites you visit ❖ Emails you open ❖ Links you click

11.3 Net and Communications Etiquettes

The word **netiquette,** derives from the combination of words — **'net'** (internet) and **'etiquette'**. It refers to online manners while using Internet or working online. While online, you should be courteous, truthful and respectful of others.

Following lines list basics rules of netiquettes.

1. **Refrain from personal abuse.** If you disagree to something, you may express robust disagreement, but **never** call them names or threaten them with personal violence.
2. **Never spam.** That is, don't repeatedly post the same advertisement for products or services.
3. **Write clearly and concisely.** On a site that has many non-native English speakers, avoid using slang they may not understand.
4. **Always post correct content in respectful language.** Remember that your posts are public. They can be read by your friends, your siblings, your parents, or your teachers.
5. **In a discussion forum, stick to the topic.** Don't post about football in a hair-care forum or about hair care in a gardening forum!
6. **Never expect other people to do your homework for you.** Never ask for or expect that your complete homework solution will be made available to you. Also, while asking for help, include details of what attempts you've made to solve the problem. It will save time and also show people that you are making an effort to help yourself.
7. **Do not post copyrighted material to which you do not own the rights.** It is plagiarism. If you need to use the copy righted material, then follow these rules :
   (*a*) Ask permission from the copyright holder.
   (*b*) Enclose the copyrighted information that you are using and cite its source.

Failing this will put you and the site in legal trouble.

11.3.1 Email Etiquettes

It is important that whether for business or personal use that we should follow the basics of email etiquettes. Following lines enlist some important email etiquette that everyone needs to be aware of and follow.

1. **Be concise and to the point.** Do not make an e-mail longer than it needs to be. A long e-mail can be very discouraging to read.
2. **Use proper spelling, grammar & punctuation.** It is important for conveying the message properly. Improper spelling, grammar and punctuation give a bad impression.
3. **Use proper structure & layout.** Since reading from a screen is more difficult than reading from paper, the structure and lay out is very important for e-mail messages. Use short paragraphs and blank lines between each paragraph. When making points, number them or mark each point as separate to keep the overview.
4. **Do not write in CAPITALS (Case Sensitivity).** IF YOU WRITE IN CAPITALS IT SEEMS AS IF YOU ARE SHOUTING. This can be highly annoying and might trigger an unwanted response. Therefore, try not to send any email text in capitals.
5. **Add an e-mail disclaimer.** Disclaimers in your internal and external mails can help protect you from liability if you inadvertently forwarded a virus by e-mail.
6. **Handle abbreviations and emoticons with care.** In business e-mails, use of abbreviations (such as BTW for by the way) and emoticons (smiley faces) is generally inappropriate.
7. **Gender Sensitivity.** If you are writing to a person unknown to you, your email should be gender neutral. That is, you should carefully not write or replace the *gendered nouns* with ***gender-neutral nouns***, *e.g.*, as shown in some samples below :

Gendered noun	Gender-neutral noun
man, woman	person, individual
mankind	people, human beings, humanity
man-made	machine-made, synthetic
the common man	the average (or ordinary) person
chairman	chair, chairperson, coordinator
mailman	mail carrier, letter carrier, postal worker
policeman	police officer
steward, stewardess	flight attendant
congressman	congress person, legislator, representative
Dear Sir:	Dear Sir or Madam:, Dear Editor:, Dear Service Representative:, To Whom it May Concern:

11.4 Ethical Essues

These days, we can easily say that our society is information society and our era is information era. As we all know that ***information*** *is the means to acquire knowledge.* In other words, we can say that *information forms the intellectual capital* for a person or body. However, there are many ethical issues involved with the usage and availability of information.

Some common ethical issues are :

(*i*) Intellectual property rights
(*ii*) Digital property rights
(*iii*) Plagiarism

11.4.1 Intellectual Property Rights

As mentioned earlier, information makes intellectual property. Any piece of information is produced or created with a lot of efforts and it consumes a lot of time. The cost factor is also involved with the creation or production of information. Though once produced, it becomes very easy to duplicate it or share it with others. But this very thing makes information difficult to safeguard unlike tangible property.

NOTE

Intellectual property rights are the rights of the owner of information to decide how much information is to be exchanged, shared or distributed. Also it gives the owner a right to decide the price for doing (exchanging / sharing/distributing) so.

The creator/producer of the information is the real owner of the information. And the owner has every right to protect his/her intellectual property. To protect one's intellectual property rights one can get information copyrighted or patented or use trademarks.

The ethical issue involved with it is that information must not be exchanged without the consent of its owner.

The intellectual property rights must be protected, for it :

- encourages individuals and businesses to create new software and new software applications, as well as improving existing applications,
- ensures new ideas and technologies are widely distributed,
- promotes investment in the national economy.

11.4.1A Digital Property Rights

DIGITAL PROPERTY

" **Digital property** (or **digital assets**) refers to any information about you or created by you that exists in digital form, either online or on an electronic storage device. "

Digital property (or **digital assets**) refers to any information about you or created by you that exists in digital form, either online or on an electronic storage device. All of your digital property comprises what is known as your ***digital estate.***

Examples of digital property include : *any online personal accounts, such as email and communications accounts, social media accounts, shopping accounts, photo and video sharing accounts, video gaming accounts, online storage accounts, and websites and blogs that you may manage ; domain names registered in your name ; intellectual property, including copyrighted materials, trademarks, patents and any software or code (such as software tools created by you or games or apps created by you) you may have written and own etc.*

Digital property rights lie with the owner. Legally a person who has created it or the owner who has got it developed by paying legally is the legal owner of a digital property. Only the owner can use and decide who all and in what form can his/her digital asset may be used by other, whether by making payments or by buying it or by obtaining its license or usage rights etc. But this is not the case generally; there are many threats to digital properties.

Threats to Digital Properties

Let us briefly talk about common threats to digital properties :

1. **Digital software penetration tools.** Although one needs to buy usage rights or license to use a digital property, there are many software penetration tools such as *cracks* and *keygens*, tools created by hackers to penetrate your software's registration system and enable unauthorized users to freely access your software without actually paying for it.
2. **Stealing and plagiarizing codes of your digital properties.** Sometimes other developers somehow get hold of your software's source code and then create plagiarized versions of your code and use it in their own software. In other words, they steal your software's source code and use it to build their own versions of it, and then sell it under their own company name.

Digital Property Rights Protection

As there are multiple types of threats to digital properties, there are many ways you can ensure protection of your digital properties.

Let us talk about these protective measures :

1. **Anti-Temper Solutions.** There are many anti-tamper solution available today which ensure that your digital property is tamper-proof. These anti-temper solutions use a host of advanced technologies to prevent hackers from hacking, reverse-engineering or manipulating your digital properties such as *utility tools, software, apps, video games* and so forth.
2. **Legal Clauses.** Add legal clause in the clauses of use of your software/digital properties. You must include a transparent clause in your software's *Terms of Service* that prohibits the scraping of your software's source code for reuse. This is a sound legal backup for you.
3. **Limit the sharing of software code.** You should share your software code only with trusted individuals who are part of development team. You should also use a Digital Rights Management (DRM) solution to protect your software from being scraped for source code using decompilers etc.

11.4.2 Plagiarism

Simply put, *Plagiarism* means *stealing*. Surprised? If you look into an English dictionary to find the meaning of word plagiarism, it will give somewhat like "the unauthorized use or close imitation of the language and thoughts of another author and the representation of them as one's own original work."

PLAGIARISM

> **Plagiarism** is stealing someone else's intellectual work and representing it as your own work without citing the source of information.

Thus, **Plagiarism** is stealing someone else's intellectual work (can be an idea, *literary work* or *academic work* etc.) and representing it as your own work without giving credit to creator or without citing the source of information.

Any of the following acts would be termed as Plagiarism :

- Using some other author's work without giving credit to the author.
- Using someone else's work in incorrect form than intended originally by the author/creator.

- Modifying/lifting someone's production such as *music-composition* etc. without attributing it to the creator of the work.
- Giving incorrect or incorrect source of information *i.e.*, wrongful citation.
- Failure in giving credit or acknowledging the contribution of others in a collaborative effort, to which you are also part of.

How not to Plagiarize ?

As most universities[1] put in their student-handbook. 'To avoid plagiarism :

You must give credit whenever you use

- another person's idea, opinion, or theory;
- quotations of another person's actual spoken or written words ; or
- Paraphrase of another person's spoken or written words.

Plagiarism is Offence

'If plagiarism involves copying not only ideas but also a substantial portion of a copyrighted work without attribution and without permission, it would amount to both copyright infringement and the violation of the 'special right' of the author to be credited.

Copyright infringement and the violation of an author's right to be credited are both civil wrongs and criminal offences. A civil suit may be instituted, and criminal charges may also be filed'[2].

Both civil suit and criminal charges are punishable offences and amount to fine and penalties.

11.5 Open Source Philosophy

Broadly the term '*open source software*' is used to refer to those categories of software / programs whose licenses do not impose much conditions. Such software, generally, give users freedom to run/use the software for any purpose, to study and modify the program, and to redistribute copies of either the original or modified program (without having to pay royalties to previous developers).

There are many categories of software that may be referred to as open source software. Following subsection is going to talk about the same.

11.5.1 Terminology

Before we talk about various terms and definitions pertaining to '**Open**' world, you must be clear about *two* terms which are often misunderstood or misinterpreted. These terms are :

- Free software and
- Open source software

Free Software

Free Software means the software is freely accessible and can be freely used, changed, improved, copied and distributed by all who wish to do so. And no payments are needed to be made for **free software**.

1. Found in most universities' guidelines for students/student-handbooks
2. www.mxmindia.com (interview of *Nandita Saikia*)

The definition of **Free Software** is published by *Richard Stallman's Free Software Foundation*. Here is the key text[3] of that definition :

> *"Free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer." Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software :*
>
> - *The freedom to run the program, for any purpose (freedom 0).*
> - *The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.*
> - *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
> - *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.*
>
> *A program is free software if users have all of these freedoms.*

Open Source Software

Open Source Software, on the other hand, can be **freely used** (in terms of making modifications, constructing business models around the software and so on) but it ***does not have to be free of charge***. Here the company constructing the business models around *open source software* may receive payments concerning support, further development. What is important to know here is that in *open source software, the source code is freely available* to the customer.

11.5.2 Philosophy of Open Source

Open source software is officially defined by the **open source definition** at *http://www.opensource.org/docs/definition_plain.html*.

It states that :

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria :

Free Redistribution	No restriction on the re-distribution of the software whether as a whole or in part.
Source Code	The program must include source code, and must allow distribution in source code as well as compiled form.
Derived Works	The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
Integrity of the Author's Source Code	The integrity of the author'source code must be maintained. Any additions / modifications should carry a different name or version number from the original software.

3. Excerpt courtesy *Free Software Foundation*. This keytext is available at *www.gnu.org/philosophy/free-sw.html*

No Discrimination Against Persons or Groups	The license must not discriminate against any person or group of persons.
No Discrimination Against Fields of Endeavor	The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.
Distribution of License	The rights attached to the program must apply to all to whom the program is redistributed.
License must not be Specific to a Product	There must not be any restriction on the rights attached to the program, *i.e.*, there should not be a condition on the program's being part of a particular software distribution.
The License must not Restrict other Software	The license must not place restrictions on other software that is distributed along with the licensed software. *For example*, the license must not insist that all other programs distributed on the same medium must be open-source software.
Licence must be Technology Neutral	No provision of the license may be predicated on any individual technology or style of interface.

A software which is **free** as well as **open** belongs to category **FOSS** (*Free and Open Source Software*).

NOTE

The terms Free and Open represent a differing emphasis on importance of ***freedom*** (*free software*) or ***technical progress*** (open source software).

11.5.3 Definitions

After understanding the difference between the terms free and **open**, let us now proceed to our discussion on terminology and definitions pertaining to open source software.

OSS and FLOSS	OSS refers to *open source software*, which refers to software whose source code is available to customers and it can be modified and redistributed without any limitation. An OSS may come free of cost or with a payment of nominal charges that its developers may charge in the name of development, support of software. FLOSS refers to *Free Libre and Open Source Software* or to *Free Livre and Open Source Software*. The term FLOSS is used to refer to a software which is both ***free software*** as well as ***open source software***. Here the words **libre** (a Spanish word) and **livre** (a Portuguese word) mean ***freedom***.
GNU	GNU[4] refers to ***G**NU's **N**ot **U**nix*. GNU Project emphasizes on freedom. The GNU project was initiated by *Richard M. Stallman* with an objective to create an operating system. With time, GNU project expanded and now it is not limited to only an operating system. Now, it offers a wide range of software, including applications apart from operating system.

4. GNU is recrusive acronym for GNU's Not Unix. A recursive acronym is the one that uses its abbreviation in full form *e.g.*, VISA is also recursive acronym – VISA International Service Association.

FSF	FSF is *Free Software Foundation*. FSF is a non-profit organization created for the purpose of supporting free software movement. *Richard Stallman* founded FSF in 1985 to support GNU project and GNU licences. Now a days, it also works on legal and structural issues for the free software community.
OSI	OSI is *Open Source Initiative*. It is an organization dedicated to cause of promoting open source software. *Bruce Perens* and *Erics Raymond* were the founders of OSI, that was founded in February 1998. OSI specifies the criteria for open source software and properly defines the terms and specifications of *open source software*. Open source doesn't just mean access to the source code. The distribution terms of open source software must comply with the *Open Source Definition* by OSI.
Freeware	The term *freeware* is generally used for software, which is available free of cost and which allows copying and further distribution, but not modification and whose source code is not available. Freeware should not be mistaken for open software or for free software. Freeware is distributed in binary form (ready to run) without any licensing fee. In some instances the right to use the software is limited to certain types of users, for instance, for private and non-commercial purposes. One example is Microsoft Internet Explorer, which is made available as freeware.
W3C	W3C is acronym for *World Wide Web Consortium*. W3C is responsible for producing the software standards for world wide web. The W3C was created in October 1994, to lead the world wide web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. The World Wide Web Consortium (W3C) describes itself as follows : The World Wide Web Consortium exists to realize the full potential of the Web. The W3C is an industry consortium that seeks to promote standards for the evolution of the Web and interoperability between WWW products by producing specifications and reference software. Although industrial members fund W3C it is vendor-neutral, and its products are freely available to all.
Proprietary Software	***Proprietary software*** is the software that is *neither open nor freely available*. Its use is regulated and further distribution and modification is either forbidden or requires special permission by the supplier or vendor. Source code of proprietary software is normally not available.
Shareware	***Shareware*** is software, which is made available with the right to redistribute copies, but it is stipulated that if one intends to use the software, often after a certain period of time, then a license fee should be paid. Shareware is not the same thing as *free and open source software* (FOSS) for *two* main reasons : (*i*) the source code is not available and, (*ii*) modifications to the software are not allowed. The objective of shareware is to make the software available to try for as many users as possible. This is done in order to increase prospective users' will to pay for the software. The software is distributed in binary form and often includes a built-in timed mechanism, which usually limits functionality after a trial period of usually one to three months.
Copylefted Software	Copylefted software is free software whose distribution terms ensure that all copies of all versions carry more or less the same distribution terms. This means, for instance, that copyleft licenses generally disallow others to add additional requirements to the software) and require making source code available. This shields the program, and its modified versions, from some of the common ways of making a program proprietary.

11.6 Copyright and other Licenses

The Licenses are permissions given to use a product or someone's creation. *Copyright* is a related term, which you must know. *Copyright* defines the ownership rights. A **copyright** is a collection of rights that automatically vest to the creator of an original creative work such as *a literary work, a design, song, movie or software* etc. A copyright holder can give licenses to use its work in a specific way.

Let us talk about these terms in details.

Here, you should know about a related term, **Copyleft**. **Copyleft** is a license that gives rights opposite to copyright. The **Copyleft** offers users the right to freely distribute and modify the original work, but only under the condition that the derivative works be licensed with the same rights. Its symbol is also flipped copyright symbol.



(*a*) Copyright symbol



(*b*) Copyleft symbol

Figure 11.1

LICENSES

"Licenses are the permissions given to use a product or someone's creation by the copyright holder."

COPYRIGHT

"A copyright is a legal term to describe the rights of the creator of an original creative work such as a literary work, an artistic work, a design, song, movie or software etc."

Since Open Source software movement was started against the proprietary software licenses, let us now talk about various types of software licenses available in this domain.

11.6.1 Licenses and Domains of Open Source Technology

As per Open Source Initiative, "*Open source licenses are licenses that comply with the Open Source Definition — in brief, they allow software to be freely used, modified, and shared.*"

Open-source licenses make it easy for others to contribute to a project without having to seek special permission. It also protects you as the original creator, making sure you at least get some credit for your contributions. It also helps to prevent others from claiming your work as their own.

Broadly used open source licences are being given below for your reference :

1. GNU General Public License (GPL)

The GNU General Public Licence (GPL) is probably one of the most commonly used licenses for open-source projects. The GPL grants and guarantees a wide range of rights to developers who work on open-source projects. Basically, it allows users to legally copy, distribute and modify software. This means, with GPL, a user can :

Copy the software	Copy the software as many times as needed. There's no limit to the number of copies one can make.
Distribute the software however you want	There is no restriction of distribution methods and styles – can be in copied form or printed form or web-link form.
Charge a fee to distribute the software	After modifying the software, you can even charge for your software, explaining why you are charging them but the software should still be under GNU GPL.
Make whatever modifications to the software you want	You are free to make any kind of modifications to the GNU GPL software. The only catch is that the other project must also be released under the GPL.

2. GNU Lesser General Public License (LGPL)

There is another GNU license : the **Lesser General Public Licence (LGPL)**. It offers lesser rights to a work than the standard GPL licence. The LGPL is used to license *free software* so that it can be incorporated into both *free software* and *proprietary software*. The LGPL and GPL licenses differ with one major exception ; with LGPL the requirement that you have to release software extensions in open GPL has been removed.

Mostly, LGPL is used by libraries. LGPL is also called GNU libraries and formally called the Library GPL.

3. BSD License

BSD licenses represent a family of permissive free software licenses that have fewer restrictions on distribution compared to other free software licenses such as the *GNU General Public License*. There are *two* important versions of BSD licence :

the New BSD License/ Modified BSD License	The New BSD License ("3-clause license") allows unlimited redistribution for any purpose as long as its copyright notices and the license's disclaimers of warranty are maintained. The license also contains a clause restricting use of the names of contributors for endorsement of a derived work without specific permission.
the Simplified BSD License / FreeBSD License	The Simplified BSD license is different from New BSD License in the sense that the latter omits the non-endorsement clause.

4. MIT License

The MIT License is the shortest and probably broadest of all the popular open-source licenses. Its terms are very loose and more permissive than most other licenses.

The basic provisions of the license are :

- You can use, copy and modify the software however you want. No one can prevent you from using it on any project, from copying it however many times you want and in whatever format you like, or from changing it however you want.
- You can give the software away for free or sell it. You have no restrictions on how to distribute it.
- The only restriction is that it be accompanied by the license agreement. It basically says that anyone can do whatever they want with the licensed material, as long as it's accompanied by the license.

NOTE

The MIT License is the least restrictive open source license.

5. Apache License

The Apache License, grants a number of rights to users. These rights can be applied to both *copyrights* and *patents*. The Apache License offers :

Rights are perpetual	Once granted, you can continue to use them forever.
Rights are worldwide	If the rights are granted in one country, then they're granted in all countries.
Rights are granted for no fee or royalty.	There is up-front usage fee, no per-usage fee or any other basis either.
Rights are non-exclusive.	You are not the sole-licensee; other can also use the licensed work.
Rights are irrevocable	No one can take these rights away once they're granted.

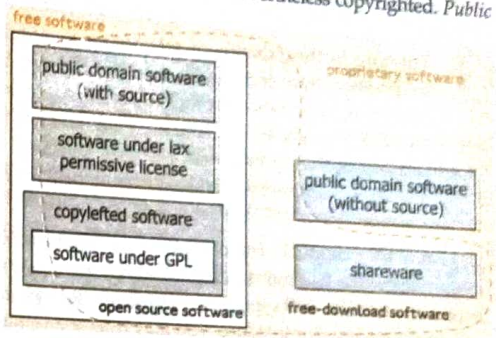
Redistributing code requires giving proper credit to contributors to the code and the same license (Apache) would remain with the software extension.

Public Domain Software vs. Proprietary Software

Public-domain software is free and can be used without restrictions. The term public-domain software is often used incorrectly to include freeware, free software that is nevertheless copyrighted. *Public domain software* is, by its very nature, outside the scope of copyright and licensing.

On the contrary, there is *Proprietary software*, which is neither free nor available for public. There is a proper license attached to it. User has to buy the licence in order to use it.

Consider the diagram (Fig. 11.2) originally made by *Chao-Kuei*[5] that describes the categories of software.



Figre 11.2 Categories and domains of software.

11.7 Cyber Crime

Cybercrime is any criminal offense that is facilitated by, or involves the use of, electronic communications or information systems, including any electronic device, computer, or the Internet. The term, **cybercrime**, is a general term that covers crimes like *phishing, credit card frauds, illegal downloading, industrial espionage, child pornography, cyber bullying, cyber stalking, cyber terrorism, creation and/or distribution of viruses, spam* and so on.

CYBER CRIME

"Any criminal offense that is facilitated by, or involves the use of, electronic communications or information systems, including any electronic device, computer, or the Internet is referred to as **Cyber Crime**."

Some cybercrimes common these days are being discussed below :

(i) Hacking

While working online, you often enter and create information related to you. You often enter information related to payments and even about your home address etc. while shopping. This information must be safe and must not fall into wrong hands. One should be careful while working online as there are many ways through which thieves can obtain your personal information.

HACKING

"Hacking refers to gaining unauthorised access to a network or computer or digital files, with an intention to steal or manipulate data or information or to install malware."

One such way of doing it is ***hacking***. **Hacking** a computer means, gaining unlawful entry in it. So a **hacker** is someone who gains unauthorised access to your network or computer or digital files, with an intention to steal or manipulate data or information or to install malware. Hackers exploit your computer/network security and employ techniques like, ***spoofing, phishing, social engineering*** etc. to capture user's personal or financial details.

5. and available under GNU GPL v2.

Spoofing

It refers to is a fraudulent or malicious practice in which communication is sent from an unknown source disguised as a trusted source known to the receiver.

In spoofing attack, a hacker or malicious individual impersonates another user or device on a network, duping users or systems into believing they are legitimate or authentic.

Phishing

Phishing is the practice of attempting to acquire sensitive information from individuals over the internet, by means of deception. Information typically targeted by phishing schemes includes passwords, user-names, bank account information, and social security numbers. The term 'phishing' is a play on 'fishing' — hackers use various forms of 'bait' in order to catch a victim.

PHISHING

> "**Phishing** is the practice of attempting to acquire sensitive information from individuals over the internet, by means of deception."

It is the criminally fraudulent process of attempting to acquire sensitive information such as usernames, passwords, credit card information, account data etc. In Phishing, an imposter uses an authentic looking email or web-site to trick recipients into giving out sensitive personal information. For instance, you may receive an email from your bank (which appears genuine to you) asking to update your information online by clicking at a specified link. Though it appears genuine, you may be taken to a fraudulent site where all your sensitive information is obtained and later used for cyber-crimes and frauds.

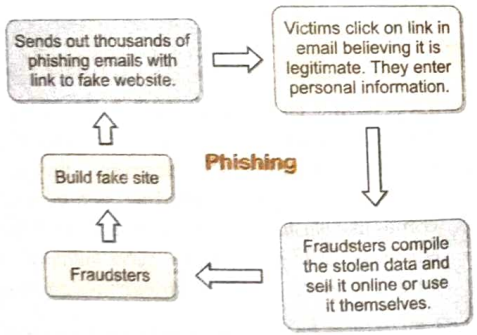


Figure 11.3 How Phishing attacks are carried out.

Social Engineering/Pretexting

They pose as a legitimate business or government officials to obtain your personal infor- mation from financial institutions, telephone companies, and other sources.

(ii) Cyber Trolls and Bullying

Cyber **troll** refers to a person who purposely posts opposing-, sarcastic-, demeaning- or insulting- comments about something or someone with an aim of targeting a person online. The provocative messages posted this way are also called **troll**. So the word *troll* can refer to a person also who is doing it and it may refers to the derogatory comments posted by a troll. Trolling is a cybercrime and is closely related to cyber bullying. In fact, it is a form of **cyber bullying**.

CYBER TROLLS

> "Derogatory messages or comments posted online targeting people are called **cyber trolls**."

Cyber Bullying

Cyberbullying occurs when someone uses the Internet, a cell phone, email, instant messaging, chat rooms, or social networks, such as *Facebook*, *Twitter* etc., to harass, demean, embarrass, or intimidate someone else. It is commonly a crime committed by teens too, as their growing access to electronic communication makes it possible to make fun of or ostracize others.

The problem spreads like wildfire as the bully can hide behind the anonymity of a login identity, while encouraging other kids to join in the "fun" of harassing the victim.

Cyberbullying is a crime, garnering such criminal charges as harassment, libel, assault, and even terrorism. In addition to criminal charges, cyberbullies may be held responsible for the damage they do in a civil lawsuit, where they may be ordered to pay for it.

CYBER BULLYING

> "Harassing, demeaning, embarrassing, defaming or intimidating someone using modern technologies like Internet, cell phones, instant messengers, social networks etc., is called **Cyber Bullying**."

(iii) Cyber Stalking

This is a kind of online harassment wherein the victim is subjected to a barrage of online messages and emails. Typically, these stalkers know their victims and instead of resorting to offline stalking, they use the Internet to stalk. A cyber stalker relies upon the anonymity afforded by the Internet to allow them to stalk their victim without being detected.

Cyber stalkers often do this to trouble their victims :

- They collect all personal information about the victim such as name, family background, telephone numbers of residence and work place, daily routine of the victim, address of residence and place of work, date of birth etc.
- The stalker may post this information on any filthy/obscene or illegal websites posing as if the victim is posting this information.
- People of all kind from nook and corner of the World, who come across this information, start calling the victim at his/her residence and/or work place for many filthy/obscene reasons.
- Some stalkers subscribe the e-mail account of the victim to innumerable obscene or illegal sites because of which victim starts receiving such kind of unsolicited e-mails.
- Some stalkers keep on sending repeated e-mails asking for various kinds of favors or threaten the victim.
- Stalkers follow their victim from board to board. They "hangout" on the same social networking site as their victim, many times posting notes to the victim, making sure the victim is aware that he/she is being followed.
- Stalkers will almost always make contact with their victims through email having friendly or threatening content. The stalker many times uses multiple names when contacting the victim.

(iv) Scams

Any fraudulent business practice that extracts money from an unsuspecting, ignorant person is called a scam. These days, the Internet has become another primary source of scams. Scams committed over the Internet are called **online scams**.

An example on online scam is being given below :

"Mrinalini wanted to gift her friend a customized bracelet whose picture she saw of a photo sharing website that listed the design and cost of customization. Mrinalini, happily transferred an advanced payment of ₹2450 to the account mentioned on the site and was promised a date of delivery. But to her shock, the gift never arrived. Even after repeated calls, she only had a promise that it would be delivered soon and suddenly that website account was deleted from the photo sharing website and even the numbers went dead."

This is one example of the common cyber scam that may happen to unsuspecting customers.

SCAM

> "Any fraudulent business practice that extracts money from an unsuspecting, ignorant person is called a scam."

Measures to avoid Online Scams

Important things to keep in mind while using the Internet to avoid scam include the following :

(*i*) Never enter personal information or any financial information (banking information or credit/debit card information) on unsecure websites, *i.e.*, the sites that do not employ HTTPS and do not have padlock sign.

(*ii*) Never reply to emails from any unknown or unreliable source.

(*iii*) Never click on any links that you have received in your email, even if you know the sender. Rather open a browser window and type the url yourself than clicking on the link in the email.

(*iv*) Never respond to an e-mail or advertisement claiming you have won something.

(v) Illegal Downloads

Illegal downloading refers to obtaining files for which you don't have the right to use or download from the Internet. It is downloading a paid digital item, without making any payment and using an illegal way to download it.

For example, If you are downloading a movie which is not available for free download, this is illegal download. Similarly, downloading a copy of the licensed software bypassing the legal measures is also illegal download.

Most items that are protected under copyright law are available against a payment. Violating this is known as illegal download. For example, a movie or a photograph or a video etc. is copyrighted in the favour of the creator/owner/producer.

A product is protected by copyright law cannot be downloaded, copied, reproduced or resold without their permission.

(vi) Child Pornography

Child pornography is defined as any visual or written representation (including images, movies and/or texts) that depict or advocate sexual activity (including sexual molestation or exploitation) of anyone under the age of 18. The law also includes some child nudity, simulated sex involving children and any material that is computer-doctored to look like child porn.

Information Technology Act, 2000 & Indian Penal Code, 1860 provides protection from child pornography. Child is the person who is below the age of 18 years.

According to the new (amended) Information Technology Bill, Section 67 has been amended – that **not only creating and transmitting obscene material in electronic form but also to browse such sites is an offence.**

11.8 Cyber Law and IT Act

Cyberlaw is a generic term which refers to all the legal and regulatory aspects of Internet and the World Wide Web. Anything concerned with or related to or emanating from any legal aspects or issues concerning any activity of netizens and others, in Cyberspace comes within the

ambit of Cyberlaw. The growth of Electronic Commerce has propelled the need for vibrant and effective regulatory mechanisms which would further strengthen the legal infrastructure, so crucial to the success of Electronic Commerce. All these regulatory mechanisms and legal infrastructures come within the domain of Cyberlaw.

Cyberlaw is important because it touches almost all aspects of transactions and activities on and concerning the Internet, the World Wide Web and Cyberspace.

India's IT Act and IT (Amendment) Act, 2008

In India the cyber laws are enforced through **Information Technology Act, 2000** (*IT Act 2000*) which was notified on 17 October 2000. It is based on the United Nation's Commission for International Trade related laws (UNCITRAL) model law.

IT ACT 2000's prime purpose was **to provide legal recognition to electronic commerce** and to facilitate filing of electronic records with the Government, *i.e.*, to provide the legal infrastructure for e-commerce in India.

The Act was later amended in December 2008 through the **IT (Amendment) Act, 2008**. It provided additional focus on Information Security. It has added several new sections on offences including Cyber Terrorism and Data Protection. **The Information Technology Amendment Act, 2008** (IT Act 2008) came into force from October 27, 2009 onwards. Major amendments of IT ACT (2008) included :

Digital Signatures	Authentication of electronic records by digital signatures gets legal recognition.
Electronic governance	E-Documents get legal recognition. Documents required as per law by any arm of the government may be supplied in electronic form.
Offences and Penalties	The maximum penalty for any damage to computers or computer systems is a fine up to ₹1 crore.
Amendments to other laws	Other related acts such as the Indian Penal Code, 1860, the Indian Evidence Act, 1872, the Bankers' Books Evidence Act, 1891, the Reserve Bank of India Act, 1934 were to be amended to align them with the IT Act.

11.9 E-Waste Management

Electronic waste, e-Waste, e-Scrap, or **Waste Electrical and Electronic Equipment** (WEEE) describes discarded electrical or electronic devices. "Electronic waste" may also be defined as discarded computers, office electronic equipment, entertainment device electronics, mobile phones, television sets and refrigerators. This includes used electronics which are destined for reuse, resale, salvage, recycling, or disposal.

Of all the different types of waste, electronic waste has the characteristics of :

(*a*) the fastest growing segment of waste

(*b*) most valuable due to its basic composition

(*c*) very hazardous if not handled carefully

11.9.1 E-Waste Disposal Process

E-Waste is categorized by the government of India under the broad class of hazardous waste. Within e-Waste, there are several categories such as Large and small household appliances, electrical and electronic toys and sporting equipment, tools, computers and related equipment and so forth.

Composition of e-waste. Electrical and Electronic equipment contains metallic and non metallic elements, alloys and compounds such as Copper, Aluminum, Gold, Silver, Palladium, Platinum, Nickel, Tin, Lead, Iron, Sulphur, Phosphorous, Arsenic etc.

The E-waste management involves proper recycling and recovery of the disposed material. The recycle and recovery includes the following unit operations.

1. **Dismantling.** Removal of parts containing dangerous substances (CFCs, Hg switches, PCB); removal of easily accessible parts containing valuable substances (cable containing copper, steel, iron, precious, metal containing parts).
2. **Segregation of ferrous metal, non-ferrous metal and plastic.** This separation is normally done in a shredder process.
3. **Refurbishment and reuse.** Refurbishment and reuse of e-waste has potential for those used electrical and electronic equipments which can be easily refurbished to put to its original use.
4. **Recycling/recovery of valuable materials.** Ferrous metals in electrical are furnaces, non-ferrous metals in smelting plants, precious metals in separating works.
5. **Treatment/disposal of dangerous materials and waste**. Shredder light fraction is disposed off in landfill sites or sometimes incinerated (expensive), chlorofluoro-carbons (CFCs) are treated thermally, Printed Circuit Board(PCB) is incinerated or disposed off in underground storages, Mercury(Hg) is often recycled or disposed off in underground landfill sites.

NOTE

IMPORTANT. Before discarding your laptop, desktop, or smartphone (*i.e.*, the e-waste), make sure to permanently delete all data and information from its storage. Otherwise this information may get stolen and may pose bigger threat through its misuse.

11.9.2 Benefits of e-Waste Recycling

The e-Waste disposal and proper recycling is very much necessary and important for the benefit of people, environment and the nation. The key benefits are :

1. **Allows for recovery of valuable precious metals.** Most consumer electronics contain valuable materials like copper, gold and zinc that can and should be recycled. Virgin Materials are significantly more costly than recycled materials for manufacturing.
2. **Protects public health and water quality.** E-waste contains a variety of toxic substances, which may include lead, mercury and cadmium. When e-waste is disposed into landfills, these toxins can be released into the atmosphere or leak in through the land and have negative health and environmental effects.
3. **Creates Jobs.** Recycling e-waste domestically creates jobs for professional recyclers and refurbishers and creates new markets for the valuable components that are dismantled.
4. **Toxic Waste.** Mining produces toxic waste, which are linked with crop devastation and human health crisis due to water contamination.
5. **Saves landfill space.** E-waste is a growing waste stream. Recycling these items will help conserve landfill space.

11.10 Health Concerns with Technology Usage

Although technology has improved our lives enormously yet it should be used in moderate amounts. Excessive use of technology, such as *smart phones, computers, online gaming, social media* etc. leads to many health related problems. In the following lines we are discussing major health concerns related to excessive technology usage :

1. Impact on Hearing

Studies have proven that listening to music that loud for more than 15 minutes cause hearing damage over time. Also, it has been said that using headphones increases the bacteria levels in your ears over 700 times when used for more than an hour. This shocking statistic came from a study way back in 1992 when experts measured bacteria on 20 headsets.

2. Impact on Bones and Joints

Use of technology has affected our postures. Most of the times, we sit in the same postures and make similar, repetitive movements, *e.g.*, thumb movements on mobile phones. Slouching, or using your joints and muscles in repetitive movements all cause strain on our muscles and joints. A **Repetitive Strain Injury (*RSI*)** is an injury or disorder of the muscles, nerves , tendons, ligaments and joints.

NOTE

A Repetitive Strain Injury (*RSI*) is an injury or disorder of the *muscles, nerves, tendons, ligaments* and *joints.*

3. Eye Problems

Constant exposure to smart phone, laptops and computer screens impacts our vision and may lead to other eye related problems. The blue light that comes from our phones and computers is very damaging on the retina, even more than UV light; this may even lead to vision loss. **Computer Vision Syndrome (CVS)** is a technology related health condition affecting eyesight.

NOTE

Computer Vision Syndrome (CVS) is a technology related health condition affecting eyesight.

4. Sleep Issues

Excessive smartphone, computer and tablet use can disrupt our sleep. Bright lights from these devices block melatonin secretion, the hormone that regulates sleep and this leads to smaller sleep cycles and disrupted sleep. Sleep is so essential for overall health that it impacts our normal *thinking and behavioural patterns, memory and attention span.*

Check Point 11.1

1. What is digital footprint ?
2. What is IPR ?
3. What is plagiarism ?
4. What is FOSS ?
5. How is freeware different from free software ?
6. What are open source licences ?
7. What is hacking and spoofing ?
8. How is e-waste disposed and recycled ?

5. Mental Health Issues

Excessive use of technology leads to isolation as people don't get time to physically socialise. It sometimes also leads to anxiety and depression as by looking at picture pefect social media profile of others, people often tend to think that their "connections" have "perfect rosy lives" while they are not.

Excessive use of technology and Internet leads to addiction. People keep obsessively looking through emails and messages. They start feeling stress if they don't get some likes or replies on their posts etc. This problem is formally termed as ***Internet addiction disorder.***

Following infobox talks about ***Internet addiction Disorder*** in details.

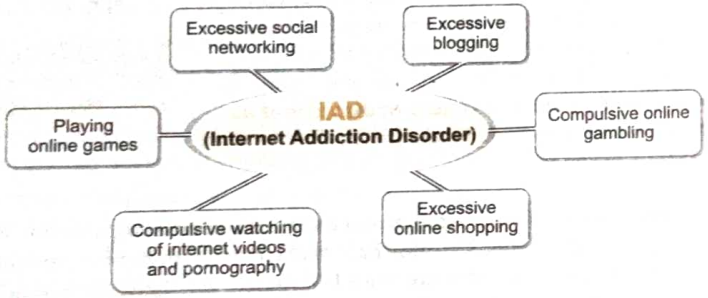
Internet Addiction

Internet addiction is a specific type of disorder, an impulse control disorder, whereby one uses Internet impulsively and compulsively. When someone uses Internet excessively on social media, blogs, online gaming, porn etc., this poses a great to the person's mental health. Such a condition is termed as **Internet addiction** by doctors.

Persons who have Internet addiction exhibit feelings of restlessness, moodiness, depression, or irritability when attempts are made by the family to cut down use of the Internet.

IAD

"When a person can't find a balance between their time online and their time offline, it considerably affects their mental health, this condition is called **Internet addiction** or **Internet Addiction Disorder** (IAD)."



Internet addiction results in many problems in the affected person, such as :

- One suffers from personal, family, academic, financial, and occupational issues, just like these happen in other types of addictions.
- Real life relationships (both personal and professional) get disrupted.
- Sufferer becomes socially awkward.
- Sufferer starts lying about the time spent on the Internet and avoids interaction with people around.
- Sufferer's changed habits and behaviour leads to loss of trust in the people or relations around him, and it leads to more loneliness.

How to Overcome Internet Addiction?

To overcome Internet addiction, one needs the help of a qualified doctor, counsellor and of course the family support. Other than doctor and counsellor's assistance, family support is crucial, for the family need to know how the sufferer may exhibit the withdrawal symptoms and how the family needs to react towards them.

When a person is trying to come out on Internet addiction, he/she may experience withdrawal symptoms like :

- **Mental and emotional symptoms.** Anger, depression, relief, mood swings, anxiety, fear, irritability, sadness, loneliness, boredom, restlessness, procrastination.
- **Physical symptoms.** An upset stomach, eating irregularities, (such as skipping meals), severe headaches, backaches, dry eyes, ignoring personal hygiene, and sleep disturbance.

Thus, for Internet addiction, a person needs timely help of qualified doctor, counsellor and family.

LET US REVISE

- *A digital footprint is a trail of data you create while using the Internet. It includes the websites you visit, emails you send, and information you submit to online services.*
- ***Intellectual property rights** are the rights of the owner of information to decide how much information is to be exchanged, shared or distributed ; and to decide the price for doing (exchanging/sharing/distributing) so.*
- ***Plagiarism** is stealing someone else's intellectual work and representing it as your own work without giving credit.*
- ***Digital property** (or **digital assets**) refers to any information about you or created by you that exists in digital form, either online or on an electronic storage device.*
- *OSS refers to Open Source Software, which refers to software whose source code is available to customers and it can be modified and redistributed without any limitation. An OSS may come free of cost or with a payment of nominal charges that its developers may charge in the name of development, support of software.*
- *FLOSS refers to Free Libre and Open Source Software or to Free Livre and Open Source Software. The term FLOSS is used to refer to a software which is both **free software** as well as **open source software**. Here the words **libre** (a Spanish word) and **livre** (a Portuguese word) mean **freedom**.*
- *Phishing is the practice of attempting to acquire sensitive information from individuals over the internet, by means of deception.*
- *Hacking refers to gaining unauthorised access to a network or computer or digital files, with an intention to steal or manipulate data or information or to install malware.*
- *Licenses are the permissions given to use a product or someone's creation by the copyright holder.*
- *A copyright is a legal term to describe the rights of the creator of an original creative work such as a literary work, an artistic work, a design, song, movie or software etc.*

Objective Type Questions OTQs

Multiple Choice Questions

1. A software that can be freely accessed and modified is called _____ .
   (a) Synchronous Software (b) Package Software
   (c) Open Source Software (d) Middleware
2. Which of the following is an advantage of 'open source' software ?
   (a) You can edit the source code to customise it.
   (b) You need to be an expert to edit code.
   (c) You have to pay.
   (d) Can sometimes be too generic for specialist purposes.
3. Which of the following is a disadvantage of 'open source' software ?
   (a) High quality software with lots of features
   (b) Not as customizable
   (c) May not have been tested as much as proprietary software, so might have bugs.
   (d) You can edit the source code to customise it

4. Which of the following is an advantage of 'proprietary' software ?
   (a) It is usually free.
   (b) Thoroughly tested because people are paying to use it.
   (c) Not as customizable
   (d) Can sometimes be too generic for specialist purposes
5. Which of the following is a disadvantage of 'proprietary' software ?
   (a) You need to be an expert to edit code.
   (b) You have to pay for this type of software.
   (c) It's licensed.
   (d) It is launched after proper testing.
6. The generally recognized term for the government protection afforded to intellectual property (written and electronic) is called _____ .
   (a) computer security law (b) aggregate information
   (c) copyright law (d) data security standards
7. Which of the following would be a creative work protected by copyright ?
   (a) A list of all Indian President names (b) A portrait of your family
   (c) A song you wrote (d) The name of your pet dog
8. Which of the following is not a type of cybercrime ?
   (a) Data theft (b) Forgery
   (c) Damage to data and systems (d) Installing antivirus for protection
9. Which of the following is not done by cyber criminals ?
   (a) Unauthorized account access (b) Mass attack using Trojans as botnets
   (c) Email spoofing and spamming (d) Report vulnerability in any system
10. What is the name of the IT law that India is having in the Indian legislature ?
    (a) India's Technology (IT) Act, 2000
    (b) India's Digital Information Technology (DIT) Act, 2000
    (c) India's Information Technology (IT) Act, 2000
    (d) The Technology Act, 2008
11. What is meant by the term 'cyber-crime' ?
    (a) Any crime that uses computers to jeopardise or attempt to jeopardise national security
    (b) The use of computer networks to commit financial or identity fraud
    (c) The theft of digital information
    (d) Any crime that involves computers and networks
12. What is an example of e-waste ?
    (a) a ripened banana (b) an old computer
    (c) old clothes (d) empty soda cans
13. The rights of the owner of information to decide how much information is to be shared/exchanged/distributed, are collectively known as _____ (IPR).
    (a) Intelligent Property Rights (b) Intellectual Property Rights
    (c) Interactive Property Rights (d) Instance Property Rights

14. Stealing someone else's intellectual work and representing it as own, is called _____ .
    (a) Intellectual steal (b) Pluckism
    (c) Plagiarism (d) Pickism
15. The information/art/work that exists in digital form is called _____ .
    (a) e-work (b) e-asset (c) digital property (d) e-property
16. Every activity you perform on the Internet is saved for how long ?
    (a) one month (b) one year
    (c) as per my setting (d) forever
17. The digital trail which gets created as a person's Internet usage using computers, smartphones, gaming consoles etc. is called _____
    (a) Internet data (b) Internet trail
    (c) Digital footprint (d) e-footprint
18. Gaining unauthorised access to a network or computer or digital files with malicious intentions, is called _____
    (a) Cracking (b) Hacking
    (c) Banging (d) Phishing
19. Legal term to describe the rights of a creator of original creative or artistic work is called _____
    (a) Copyright (b) Copyleft
    (c) GPL (d) none of these

Fill in the Blanks

1. OSS stands for _____ .
2. Any fraudulent business practice that extracts money from an unsuspecting, ignorant person is called a _____ .
3. _____ is stealing someone else's intellectual work and representing it as your own without giving credit.
4. Any work/information that exists in digital form either on Internet or on an electronic device, is known as _____ property.
5. Discarded electrical or electronic devices are known as _____ .
6. The least restrictive open source license is _____ license.
7. The original code written by programmers for a software is known as _____ .
8. _____ means no price is to be paid for the software.
9. _____ _____ means freedom to use the software.
10. IAD means _____ _____ _____ .
11. The _____ _____ is the digital trail of your activity on the Internet.
12. The _____ are the permissions given to use a product or someone's creator by the copyright holder.
13. _____ is a license that gives right opposite to copyright.
14. A _____ _____ _____ is an injury or disorder of the muscles, nerves, tendons, ligaments and joints.
15. _____ _____ _____ is a technology related health condition affecting eyesight.

True/False Questions

1. Open Source Software can be used for commercial purposes.
2. It is okay to copy and paste information from the Internet into your report then organize it.
3. Shareware software allows you to try the software before you buy it.
4. Freeware is copyrighted software that is freely available to use.
5. Cyber-laws are incorporated for punishing all types of criminals only.
6. Deceptively attempting to acquire sensitive information of someone else using online means, is a cybercrime.
7. Freeware and free software mean the same thing.
8. Excessive use of Internet and social media is termed as a disorder.
9. Digital footprint can be deleted.
10. Digital footprint remains forever.
11. It is safe to make all one posts public on social media.
12. Hacking is performed to help find the security loopholes.
13. If you post something mean about someone, you can just delete it and your activity will be undone.
14. Hacking is a cybercrime.
15. Copyright is the right of the creator of creative/artistic work.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *What is digital footprint ?*

   Solution. A digital footprint is the record or trail left by the things one does online. The social media activity, the information on personal website, the browsing activities, online subscriptions, any photo galleries and videos uploaded by a user — essentially, any activity carried out on the Internet makes the digital footprint of a user.

2. *Why is it important to have positive digital footprint ?*

   Solution. It is very important to have a clean and secure digital footprint because :

   - It gives us a digital persona by defining our online behaviour.
   - The digital footprint is often used by universities before approving admissions to a student.
   - The digital footprint is also used by future employers, and law enforcement offices, to find people with positive and clean digital footprint.
   - The digital footprint should not provide personal information as it could be misinterpreted or misused for theft of identity.

3. *What are intellectual property rights ?*

   Solution. **Intellectual property rights** are the rights of the owner of information to decide how much information is to be exchanged, shared or distributed. Also it gives the owner a right to decide the price for doing (exchanging/sharing/ distributing) so.

4. *Why should intellectual property rights be protected ?*

   Solution. The intellectual property rights must be protected because protecting them

   - encourages individuals and businesses to create new software and new software applications, as well as improving existing applications,
   - ensures new ideas and technologies are widely distributed,
   - promotes investment in the national economy.

5. *What do you understand by plagiarism ? Why is it a punishable offence ?*

   Solution. Plagiarism is the act of using or stealing someone else's intellectual work, ideas etc. and passing it as your own work. In other words, plagiarism is a failure in giving credit to its source.

   Plagiarism is a fraud and violation of Intellectual property rights. Since intellectual property holds a legal entity status, violating its owner's right is a legally punishable offence.

6. *What is digital property ? Give some examples of digital properties.*

   Solution. Digital property (or digital assets) refers to any information about you or created by you that exists in digital form, either online or on an electronic storage device.

   Examples of digital property include : **any online personal accounts (email/social media accounts/ shopping accounts/video gaming accounts, online storage accounts) and personal websites and blogs ; domain names registered in your name; intellectual properties etc.**

7. *Describe the terms free software and open source software. Write examples of one Proprietary and one OSS Software.*

   Solution. *Free software* is the software free of cost, which can be copied, modified and redistributed as well but whose source code is not available. *Open source software*, on the otherhand, is the software, whose source code is available and which can be copied, modified and redistributed as well. There may or may not be charges payable for open source software.

   Open Source Software : Linux ; *Proprietary Software* : Microsoft Windows 8.

8. *Expand the following terms : (i) OSS (ii) SDLC (iii) GNU (iv) FLOSS*

   Solution.

   (*i*) **OSS.** Open Source Software. (*ii*) **SDLC.** System Development Life Cycle.
   (*iii*) **GNU.** GNU is Not Unix. (*iv*) **FLOSS.** Free Libre/Livre and Open Source Software.

9. *Mr. Jayanto Das is confused between Shareware and Open source software. Mention at least two points of differences to help him understand the same.*

   Solution. **Shareware** is software, which is made available with the right to redistribute copies, but it is available for limited time, often after a certain period of time, then a license fee should be paid.

   Shareware is not the same thing as *free and open source software* (FOSS) for *two* main reasons : (*i*) the source code is not available and, (*ii*) modifications to the software are not allowed.

   OSS refers to ***open source software***, which refers to software whose source code is available to customers and it can be modified and redistributed without any limitation. An OSS may come free of cost or with a payment of nominal charges that its developers may charge in the name of development, support of software.

10. *Would you suggest open source software (OSS) for an organization or sector where the **performance** is the factor of utmost importance, such as Military ?*

    *You must be aware that military has different software needs than the commercial sector because of its unique mission and environment. While **commercial sector** chooses software on the basis of factors like : application choice, ease of use, service and support, price, reliability and performance, **the military** does the same depending upon factors like : reliability, long-term supportability, security, scalability and performance of the software.*

Keeping in mind the above scenario, answer the above question with a proper justification. Give example of a software, if you are recommending one.

Solution. The open source software comes with characteristics like : availability of open source, no license restriction on type of usage, freedom to modify, redistribute etc. But all these characteristics are not enough to be suggested as a preferred software. Not all the OSS provide long-term support or are secure-enough or scalable. Therefore, we can not blindly recommend any OSS software. The same applies to proprietary software as well.

Therefore, we can recommend only those software to Military that have solid support-base in the form of *community-development-groups* etc. and that have features like *security, scalability* and *performance* along with it. Thus, the OSS software suite like LAMP (*Linux, Apache, MySQL, PhP*) that has all the above mentioned features can be recommended to Military.

11. *List a table of differences between Free software and Open source software.*

Solution.

Free Software	Open Source Software
❖ Freedom to run program for any purpose	❖ It has distribution of license.
❖ Freedom to study about program	❖ Availability of source code.
❖ Freedom to distribute copies of software	❖ Free distribution
❖ Freedom to modify/improve program and release improvements to public	❖ Integrity of Authors Source Code.

12. *Compare freeware and Shareware.* [CBSE Sample Paper II, 12]

Solution. Freeware is computer software that is available for use at no cost or for an optional fee. Freeware is generally proprietary software available at zero price and is not free software. The author usually restricts one or more rights to copy, distribute and make derivative works of the software.

Shareware is usually offered as a trial version with certain features only available after the license is purchased, or as a full version, but for a trial period. Once the trial period has passed, the program may stop running until a license is purchased. Shareware is often offered without support, updates, or help menus, which only become available with the purchase of a license. The words "free trial" or "trial version" are indicative of shareware.

13. *Differentiate between open source and open data.* [CBSE Sample Paper 2019-20]

Solution. The term '**open source**' is applicable to software, which means source code of a software is freely available and user can make changes in it and reuse it.

The term '**open data**' is applicable to data that is freely available to everyone to use without any licensing or copyright requirements.

14. *As a citizen of India, what advise you should give to others for e-wate disposal ?*

Or

[CBSE Sample Paper 2019-20]

How should one dispose off the E-waste ?

Solution. E-wastes should not be given to unauthorized vendors/buyers. The respective pollution control boards in different states, authorize agencies to collect e-waste from generators. The dealers should have valid consent and authorization. This authorization is given based on the competency of the recycler, infrastructure and other factors as decided by the regulatory authorities.

15. *What are the freedoms that a free software must provide ?*

Solution. According to the Free Software Definition free software must fulfill *four* freedoms :

(*i*) The freedom to run the program, for any purpose

(*ii*) The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.

(*iii*) The freedom to redistribute copies.

(*iv*) The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

16. *Nivedita has recently shifted to new city and new school. She does not know many people in her new city and school. But all of a sudden, someone is posting negative, demeaning comments on her social networking profile, school site's forum etc.*

She is also getting repeated mails from unknown people. Everytime she goes online, she finds someone chasing her online.

(*a*) *What is this happening to Nivedita ?* (*b*) *What action should she taken to stop them ?*

Solution. (*a*) Nivedita has become a victim of cyber bullying and cyber stalking.

(*b*) She must immediately bring it into the notice of her parents and school authorities. And she must report this cyber crime to local police with the help of her parents.

17. *Describe following Cybercrimes : (a) Cyber Bullying (b) Cyber Stalking*

Solution. Cyber bullying refers to act of online harassment of someone by using online tools such as Internet, email, instant messages, chat rooms or social networking sites etc. Cyber trolling, which means posting of sarcastic, demeaning or insulting comments about someone, is also considered form of cyber bullying.

Cyber Stalking refers to online stalking where someone uses Internet, chat rooms, social networking sites, emails etc. to stalk his/her victim. Cyber stalker follows the victim online everywhere and keeps posting/sending something which are unsolicited.

GLOSSARY

Digital Footprint	Online activity trail of users on the Internet.
FLOSS	Free Libre (or Livre) Open Source Software. Software that are free as well as open.
Freeware	Software that are available at no cost but cannot be modified.
Free Software	Software available free of cost and also can be copied and redistributed but no source code is available.
OSS (Open Source Software)	Software whose source code is available and which can be modified, copied and redistributed.
Proprietary Software	Software that are neither free nor open.
Shareware	Software for which licence fee is payable after some time limit.
Digital Property	Any information about you or created by you that exists in digital form, either online or on an electronic storage device.
Plagiarism	Stealing someone else's intellectual work and representing it as your own.
Intellectual Property	Creative creations of mind such as patents, literary and creative arts, copyrights, trademarks tec.
Phishing	Attempting to acquire sensitive information from individuals over the internet, by means of deception.
Scam	Any fraudulent business practice that extracts money from an unsuspecting, ignorant person.

Assignment

1. What is digital footprint ?
2. Why is it so important to carefully think of digital footprint ?
3. What is intellectual property ? What do you understand by intellectual property rights ?
4. What is plagiarism ?
5. How can you avoid plagiarism while referring to someone's else's creation ?
6. What is digital property ?
7. What are the threats to digital properties ?
8. How can you protect your digital properties ?
9. Expand the terms : OSI, FLOSS, FSF, GNU, GPL W3C, OSS.
10. What are open source based software ?
11. Compare and Contrast
    (*i*) Free software and Open source software
    (*ii*) OSS and FLOSS
    (*iii*) Proprietary software and Free software
    (*iv*) Freeware and Shareware
    (*v*) Freeware and Free software.
12. What is public domain software ? Name some common public domain software licenses.
13. What is GPL software license ? How is it different from LGPL software license ?
14. These days it is equally important to safeguard digital properties :
    (*a*) What are digital properties ?
    (*b*) Why is above given statement/fact important ?
    (*c*) What are possible threats to digital properties ?
    (*d*) How can digital properties be safeguarded ?
15. Define these terms :
    (*i*) Phishing (*ii*) Scam
    (*iii*) Hacking (*iv*) Spoofing
16. What is cyber crime ? Give examples.
17. Can you give an example of crime which is not cyber crime ?
18. What is Internet addiction ? What are some symptoms of the Internet addiction ?
19. What is cyber bullying ?
20. What is copyright ?
21. What are some types of software licenses ?
22. What are netiquettes ? Why are these important ?
23. What is hacking ? Who is a hacker ?
24. What are some health concerns related to the excessive use of technology ?
25. How does excessive use of technology impact the mental health of a user ?
26. How does excessive use of technology impact the physical health of a user ?

12 Data Protection

Outline

12.1 Introduction
12.2 Threats to Data
12.3 Data Protection Solutions
12.4 Firewall – An Important Solution for Computer Security

12.1 Introduction

We live in a world at a time which is often called information age where information is freely available from any part of the world. The Internet has revolutionized the world and people are getting used to having access to whatever information they want anytime, anywhere and from a wider and wider range of computing devices.

Unfortunately, this also has resulted in increased security concerns. Now, even if you ensure not to use any external media on your computers but if you are connected to the Internet, your computer and data on it is still vulnerable.

This chapter talks about the threats that are associated with online access and the measures to prevent and counter them, if ever they occur.

12.2 Threats to Data

A **threat** is a potential violation of security. When a *threat* is actually executed, it becomes **attack**. Those who execute such actions, or cause them to be executed are called *attackers*.

Some common threats the average computer user faces every day are being given below :

- Viruses : *Worms* *Trojans*
- Spyware
- Adware
- Spamming
- PC Intrusion :
  - *Denial of service* *Sweeping* *Password Guessing*
- Phishing

> **NOTE**
>
> Malware is a general term used to refer to viruses, worms, spyware, adware etc. In other words, it is unwanted software that someone else wants to run on your computer. Malware infects your computer, making it behave in a way, which you do not approve of.

12.2.1 Computer Viruses

Computer viruses are malicious codes/programs that cause damage to data and files on a system. Viruses can attack any part of a computer's software such as boot block, operating system, system areas, files and application-program-macros etc.

Two other similar programs also cause virus like effects. These are :

(a) ***Worms.*** A worm is a self-replicating program which eats up the entire disk space or memory. A worm keeps on creating its copies until all the disk space or memory is filled.

(b) ***Trojan Horses.*** A Trojan horse is a program that appears harmless (such as a text editor or a utility program) but actually performs malicious functions such as deleting or damaging files.

DAMAGE CAUSED BY VIRUSES

Viruses can do the following if left unchecked :

- **Damage or delete files.** Some viruses may delete or damage random documents or specific files that are crucial to your operating system—for example, operating system files. This damage can range from rendering useless just a few files to affecting your entire computer, possibly requiring you to reinstall your operating system and start from scratch.
- **Slow down your computer.** Viruses can run in the background, without being seen, and may cause your computer to run extremely slow.
- **Invade your email program.** Some forms of viruses may wreak even more havoc by spreading themselves to the contacts in your address book.

> **NOTE**
>
> Not all computer problems are caused by viruses. Even though your computer, or a particular program, may not be running properly, this could be caused by other things, such as a bug (which is simply an error in the application's code), or misconfiguration of software or hardware.

12.2.2 Spyware

Spyware is a software which is installed on your computer to spy on your activities and report this data to people willing to pay for it. It tracks the user's behavior and reports information back to a central source. These are used to spy on some one either for legal or illegal purpose.

Spyware mostly get installed on your PC without your consent. Typically, spyware finds its way onto PCs by "piggybacking" onto a file, or gets downloaded from the Internet when you visit a particular website. Pests such as spyware can often lurk silently on your computer until someone or something sets them off, or until they are found and properly removed.

> **NOTE**
>
> Do you know that roughly 32% computers of the world are infected with some types of malware.

DAMAGE CAUSED BY SPYWARE

Spyware can act like a peeping tom or, at worse, a geeky thief.

For example, it :

- **Compromises your data, computing habits, and identity.** Spyware can monitor information about your computing habits, such as what websites you visit, or record your keystrokes, which in the end can lead to identity theft. For example, spyware can record the keystrokes that you use while keying in a credit card number and send this number to a "cyberthief."
- **Alters PC settings.** Some forms of spyware can also alter computer settings like your web browser home page setting or the placement of your desktop icons. This doesn't do much damage to your PC, but it's really annoying.
- **Slows down your PC.** Spyware can rob your PC or system speed and Internet access efficiency. This can become a big problem when you're trying to use the programs on your PC, watch videos online, or download large files.

12.2.3 Adware

These are the programs that deliver unwanted ads to your computer (generally in Pop-Ups form). They consume your network bandwidth. Adware is similar to spyware—however, it may be installed with your consent. So it is advised that you thoroughly read installation agreements before you allow installation of a software.

DAMAGE CAUSED BY ADWARE

Adware comes complete with the following disadvantages :

- **Adware tracks information just like spyware.** Adware tracks information about your data and computing habits to produce targeted advertising, such as pop-up ads, on your computer screen.
- **Displays arrays of annoying advertising.** When infected with adware, you will likely see frequent pop-up ads appear out of nowhere. This may even happen every time you open your web browser.
- **Slows down your PC.** The adware software working in the background and the bombardment of ads can slow your PC to a crawl.

12.2.4 Spamming

Spamming refers to the sending of bulk-mail by an identified or unidentified source. In non-malicious form, bulk-advertising mail is sent to many accounts. In malicious form (*e.g.*, e-mail bombing), the attacker keeps on sending bulk mail until the mail-server runs out of disk space.

NOTE

You can't get a virus simply by reading email. It is activated only when you click a link in it or open an attachment.

DAMAGE CAUSED BY SPAMMING

Spamming comes complete with the following disadvantages :

- **Spam reduces productivity.** The billions of spam messages circulating across the Internet can disrupt email delivery, degrade system performance, and reduce overall productivity.
- **Spam eats up your time.** Deleting spam emails seems like the simple solution, but it eats up a significant amount of productivity.
- **Spam can lead to worse things.** Spam messages may contain offensive or fraudulent material and can even be used to spread viruses.

12.2.5 PC Intrusion

Every PC (personal computer) connected to the Internet is a potential target for hackers. Computers are under constant attack from cyber vandals. PC Intrusion can occur in any of the following form.

(*i*) ***Sweeper Attack.*** This is another malicious program used by hackers. It sweeps *i.e.*, deletes all the data from the system.

(*ii*) ***Denial of Services.*** This type of attack eats up all the resources of a system and the system or applications come to a halt. Example of such an attack is flooding a system with junk mail.

(*iii*) ***Password Guessing.*** Most hackers crack or guess passwords of system accounts and gain entry into remote computer systems. And then they use it for causing damages in one or another form.

12.2.6 Eavesdropping

Eavesdropping is a passive attack in which an attacker gains access to the communication-medium through which some communication is taking place and then listens to the communication and gets information about the content of the message.

EAVESDROPPING

"Unauthorised monitoring of other people's communications is called **Eavesdropping**"

Eavesdropping can be carried out through all communication devices and media of today–telephone systems, emails, instant messaging, other Internet services (*e.g.*, chat rooms, social networking websites etc.), mobile devices etc. Eavesdropping activities do not affect normal operation of transmission and communication ; thus both the sender and the recipient can hardly notice that the data has been stolen, intercepted or defaced.

For example, while sending emails, if the email message is not encrypted and digital signature has not been used, then the attacker can exploit these security loopholes. Because of these security lapses, the attacker can launch a *Man-in-the-Middle attack* on the network and intercept

the message being transmitted. The attacker can then deface the message and send it to the recipient. The recipient is then deceived into believing the defaced message is the real message and may act as per the defaced message and may provide personal or sensitive information.

Similarly sending or providing confidential information over insecure protocols like HTTP makes the information more prone to eavesdropping attack.

12.2.7 Phishing and Pharming

It is the criminally fraudulent process of attempting to acquire sensitive information such as usernames, passwords, credit card information, account data etc. In Phishing, an imposter uses an authentic looking email or web-site to trick recipients into giving out sensitive personal information. For instance, you may receive an email from your bank (which appears genuine to you) asking to update your information online by clicking at a specified link. Though it appears genuine, you may be taken to a fraudulent site where all your sensitive information is obtained and later used for cyber-crimes and frauds.

Pharming (pronounced "farming") is an attack in which a hacker attempts to redirect a website's traffic to another, bogus website. Through pharming attack, the attacker points you to a malicious and illegitimate website by redirecting the legitimate URL. Even if the URL is entered correctly, it can still be redirected to a fake website.

In this the attacker convinces you that the site is real and legitimate by spoofing or looking almost identical to the actual site down to the smallest details. You may enter your personal information and unknowingly give it to someone with malicious intent.

12.2.8 Cookies

A cookie, also known as a web cookie or a browser cookie, is a small piece of data sent from a website and stored in a user's web browser (in a text file) while a user is browsing a website. Some cookies disappear after user closes his browser while others, known as tracking cookies, remain saved and load the next time user visits the same websites.

These cookies help track user's browsing sessions and load information faster, but create some security and privacy concerns as well. These security and privacy concerns are :

1. ***Session Data.*** When you visit a website on a regular basis, such as your email or online bank, you may not have to enter your username and password to get in. That's because the information is being pulled from a tracking cookie. While these cookies encrypt the information they store, if somebody found out the encryption key and acquired your cookies, he could discover your passwords. While the odds of this happening are extremely small, the risk does exist.

2. ***Tracking Information.*** When you visit certain websites with advertisements, those ads create cookies that store and track your online patterns. You may have noticed that if you go to a clothing store's website, for example, you'll see ads for that store when you click away to other websites. That's because tracking cookies have relayed this information back to the advertisers, who use it to target their ads. Sometimes, your information may be sold to other companies. Some people view this as an invasion of privacy.

3. ***Public Computers.*** While the same general threats exist for traffic cookies saved on public or shared computers as those saved on your personal computer, the much larger amount of people who have access to these computers makes saving traffic cookies more risky. When you finish using a public or shared computer, delete the cookies to ensure that the next people who use the same computer can't access the information.

12.3 Data Protection Solutions

The entire computer security is based on a system of actions and safeguards that are designed to protect a computer system from deliberate (or accidental) access and/or damage by these threats. In this section, we are going to categorize the solutions to these threats in *two* ways :

- Active Protection
- Preventive Measures

Active Protection

Installing and properly using an antivirus software that includes internet security – which includes protection against threats such as viruses, Spyware, and PC intrusion – is vital for proper protection against the hackers, intruders, and other wrongdoers.

Preventive Measures

Even though security programs may actively detect and eliminate any threats your PC encounters, you should always help to prevent these issues from ever arising.

Let us now move on to the real solutions to the threats that we have discussed so far.

12.3.1 Solutions to Viruses, Adware and Spyware

There are thousands of viruses, Spyware and other Malware currently *"in the wild,"* and more appear each week, so protecting your systems against infection by viruses is an essential part of information security. Protecting yourself against viruses involves the following safeguards :

Active Protection

- Use Anti-Virus and Anti-Spyware software.

You need these programs help to detect and eliminate any malware that sneaks its way onto your PC. If a virus somehow is found on a system on your network, you should do the following :

- Scan all your systems for evidence of the virus.
- Disconnect any infected systems immediately from your network.
- Restore the infected systems from a clean backup.
- Notify your antivirus vendor so it can ensure its signature database is up-to-date.

- Download updates regularly.

New viruses and other malware emerge everyday, and your security software needs to know about them in order to provide full protection. This is done through *signature file updates*, so that your anti virus is laden with all new emerging viruses.

To keep you antivirus, anti-spyware program updated, the best option is to set the program to *automatically update* when needed, so you don't have to worry – you'll always be fully protected.

- Run frequent full-system scans.

Even though most security software programs actively scan your PC for malware, you should also perform a full system scan at least once a month.

Preventive Measures

- Keep your system up-to-date.

Malware often takes advantage of security holes in operating systems and software programs. You should always install any available updates for your operating system, such as Windows, and any common software you use.

- Use caution when downloading files on the Internet.

Only download files from reputable websites by looking for signs, such as a privacy statement, full contact information, and SSL encryption of sensitive information, typically indicated by a padlock in the lower-right corner of your web browser.

- Be careful with email.

Email is a very convenient and useful communication method; however, it is also used by hackers, spammers, and criminals to get what they want. Follow these guidelines :

- Don't download or open unsolicited email attachments.
- Watch for Phishing scams. Be careful of an email that asks you to verify your personal details. Don't click on any link provided in the email. Rather type the address of concerned bank or site in browser and find out the truth.
- Check for security alerts.
- Review your web browser settings regularly *e.g.*, disable running of scripts and cookies etc.
- Disconnect from the Internet when you're away.

COOKIES

> "Cookies are small files created on client computers when these systems browse certain Web sites. These cookies can contain information about the user."

- Disable Cookies, if possible

If you are particular about the safety of your personal information, then you may disable cookies on your browser. For this, just click the Help menu of your browser and search for how to block cookies.

12.3.2 Solutions to Spam, Eavesdropping

Spam has become the bane of the Internet, and still there is no real solution in sight. Spam is usually defined as "unsolicited e-mail" and resembles the flyers from stores that clog your newspapers each morning, but it's much more than that. Protecting yourself against Spam involves the following safeguards :

Active Protection

- Use Anti-Spam software.

Following are *two* of main methods used by anti-spam software to get rid of spam :

(*i*) **Sender filtering.** This method allows only messages from your approved sender list to reach your inbox – all other mail is quarantined for later review. *Sender filtering* is done on the basis of *digital certificates* and *digital signatures*.

- ***Digital Certificates***, *specially formatted digital information issued to website*, are used to verify the identity of the message sender to the recipient. Digital certificates are issued by a certificate authority (CA) that is trusted by both the sender and recipient.
- ***Digital signatures*** are a way of authenticating the identity of creators or producers of digital information. A digital signature is like a handwritten signature and can have the same legal authority in certain situations, such as buying and selling online or signing legal contracts.

DIGITAL CERTIFICATES

> "Digital Certificates, specially formatted digital information issued to website, are used to verify the identity of the message sender."

DIGITAL SIGNATURES

> "Digital signatures are a way of authenticating the identity of creators or producers of digital information."

(*ii*) **Keyword filtering.** This method filters out email messages that contain certain keywords or phrases, which are defined by you or others.

Preventive Measures

- Keep your email address private.

Be careful whom you give your email address to. Before giving your address out on an online form, check if there is a website privacy policy. This policy typically informs you of how they handle your personal information. Signing up for free offers seen online or by email may dramatically increase your chances of receiving spam messages.

- Use encrypted connection always especially if you have to provide sensitive information. Encrypted connections are made possible through protocols like Hypertext Transfer Protocol Secure (HTTPS) and Secure Shell (SSH) and offer better security to data being transmitted.
- Install personal firewall on computers connected to the Internet so as to keep a check on incoming and outgoing information and connections.
- Always avoid conducting online transactions or using online banking services on public networks or public Internet facilities (*e.g.*, public WiFi).
- Install protection software such as **Internet security software** that also provides intrusion prevention system to detect and prevent further attacks by eavesdroppers.

12.3.3 Solutions to PC Intrusion

The combination of identification, authentication and authorization can control access to a system. This combination is very useful especially in network security. Various techniques used for network security are given below :

Active Protection

- Authorization

Asking the user a legal login-id performs authorization. If the user is able to provide a legal login-id, he/she is considered an *authorized user*.

- Authentication

Authentication is also termed as **password-protection** as the authorized user is asked to provide a valid password, and if he/she is able to do this, he/she is considered to be an *authentic user*.

- Firewall

A system designed to prevent unauthorized access to or from a private network is called ***Firewall.*** Firewalls are a mechanism to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets.

Preventive Measures

- Use proper File access permissions when sharing files on the Internet.

File access permissions refer to privileges that allow a user to read, write or execute a file.

- If a user has **Read permission** for a file, he/she can view and read the file.
- If a user has **Write permission** for a file, he/she can edit and write into the file.
- If a user has **Execute permission** for a file, he/she can execute the file.

File permissions are given for *three* sets of users : *owner*, *group* and *others*.

- **Owner** – the user who has created the file.
- **Group** – the group of users who are working with the owner as a group.
- **Others** – all other users.

So, you can decide upon the file permissions, so that an unknown user does not get write or execute permissions at all.

- Disconnect from the Internet when away

Using "always on" Internet connections such cable and DSL increases your chances of some infections and intrusions as your PC is always connected to the Internet. This doesn't mean you should switch back to dial-up Internet– however, you may want to disconnect from your "always on" connection when you don't plan on using it for a long period of time.

12.3.4 Solutions to Phishing and Pharming Attack

Phishing is the fishing for confidential information. It is a scam that encompasses fraudulently obtaining and using an individual's personal or financial information. ***Pharming*** refers to the redirection of an individual to an illegitimate Web site through technical means. To counter these evil twins is a task where you need alertness and carefulness.

You can use following safeguards to counter these attacks :

Active Protection

If, somehow, you have become a victim of these attacks, then follow these guidelines :

- Take the computer offline

Disconnecting from the internet reduces the probability of infecting other devices in the same network with malware.

- Backup all files on the hard drive

If backing up data is routinely done, it is only necessary to backup new files. Focus on capturing all sensitive data and irreplaceable files such as videos and family photos.

- List the information given to phishing scammers

Depending on what was leaked, one may need to change passwords, or cancel credit cards etc. BUT here you need to be very very cautious. DO NOT USE THE COMPROMISED COMPUTER TO CONTACT AGENCIES because using the same device may open users to the risk of a freshly installed ***keylogger***, which can record the new password as well.

- Run anti-virus software.

Do not trust free anti-virus programs. They can be malware in disguise. It is advised to take the infected computer to a professional who specializes in malware removal.

- Contact credit agencies to report any possibilities of identity theft

Reach out to credit bureaus that operate and manage the credit cards, as it will sound the alarm to potential creditors.

Preventive Measures

Phishing, and to a lesser extent, ***pharming***, rely on tricking users rather than advanced technology, the best way to handle these threats is through vigilance.

- Don't open emails from unknown sources or click on links embedded in suspect messages.
- Check the security guidelines of websites such as PayPal so that you can distinguish between legitimate and bogus emails.

- Also, rather than clicking on the link embedded in an email, you can type the general link in your web browser (*e.g.* http://www.paypal.com).
- Most important, **when in doubt, do not click.**

Physically protecting systems	Using software to protect systems
❖ Keeping important computers such as servers or mainframes in locked rooms. ❖ Posting security guards. ❖ Security locks, smart cards. ❖ Keeping sensitive data on stand- alone machines instead of networks. ❖ Using alarm systems and video cameras.	❖ Data encryption. It is a method in which data is 'scrambled' before being transmitted through a network. Only the authorised recipient has the 'key'. ❖ Firewalls. A software to block access from outside (and to prevent data being sent out in some cases). ❖ Activity or Audit logs to track who has been doing what on the system. ❖ User IDs and Passwords – to authenticate users. ❖ Access rights – to ensure information safety.

12.4 Firewall – An Important Solution for Computer Security

An Internet firewall is a device or software that is designed to protect your computer from data and viruses that you do not want. A firewall is so called because of the real firewalls used to secure buildings. A physical firewall is a set of doors that closes in a building so as to contain a fire to one area, preventing the entire building from being destroyed. Likewise an Internet firewall is designed to shut off access to your operating system or to other computers that are connected to your network. Firewalls can be implemented in *two* forms :

FIREWALL

"A **firewall** is a network security system, either hardware- or software-based, that controls incoming and outgoing network traffic based on a set of rules."

1. Software Firewall

A software firewall is a special type of computer software running on a computer. It protects your computer from outside attempts to control or gain access, and, depending on your choice of software firewall, it could also provide protection against the most common Trojan programs or e-mail worms.

2. Hardware Firewall

It is physical piece of equipment designed to perform firewall duties. A hardware firewall may actually be another computer or a dedicated piece of equipment which serve as a firewall. Hardware firewalls can be effective with little or no configuration, and they can protect every machine on a local network.

Firewalls keep out malevolent hackers and people who intended to do damage and take over other peoples' servers. Firewalls really serve no other purpose. Firewalls seek to limit the access to a server or computer and let in only the people who need to be there.

NOTE

Confidentiality of information ensures that only authorized users get access to sensitive and protected data.

Check Point 12.1

1. Both email attachments and downloaded files can spread malware.
   (*a*) True (*b*) False
2. What is a firewall ?
   (*a*) A wall that is reinforced and cannot catch on fire.
   (*b*) A program that protects against viruses.
   (*c*) A filter for an Internet connection that monitors outgoing and incoming activity.
3. A strong password should contain :
   (*a*) Both uppercase and lowercase letters.
   (*b*) A word that is easy to remember, such as the name of a pet.
   (*c*) At least 8 characters, and a combination of letters, numbers, and characters.
4. Which type of program can send out information about your web browsing activities or other personal details ? (*a*) Cookies (*b*) Spam (*c*) Spyware (*d*) Trojan

LET US REVISE

- *A **threat** is a potential violation of security. When a threat is actually executed, it becomes **attack**.*
- *Computer viruses are malicious codes/programs that cause damage to data and files on a system.*
- *A worm is a self-replicating program, which eats up the entire disk space or memory.*
- *A trojan horse is a program that appears harmless (such as a text editor or a utility program) but actually performs malicious functions such as deleting or damaging files.*
- *Spyware is a software which is installed on your computer to spy on your activities and report this data to people willing to pay for it.*
- *Adware are the programs that deliver unwanted ads to your computer.*
- *Malware is a general term used to refer to viruses, worms, spyware, adware etc.*
- *Spamming refers to the sending of bulk-mail by an identified or unidentified source.*
- *Phishing is the criminally fraudulent process of attempting to acquire sensitive information pertaining to a user.*
- *To keep a computer protected, one should use updated software, be cautious while handling mails and surfing Internet, scanning computer regularly etc.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. A worm is...
   (*a*) A slimy pink thing that lives in dirt.
   (*b*) Pieces of malicious code that make copies of themselves and spread through computers without human interaction.
   (*c*) An unsolicited email message.
2. If you receive an email claiming to need your username and/or password, what should you do ?
   (*a*) Report it as phishing/spam through your email provider
   (*b*) Delete the message
   (*c*) Reply to the message with your email and password
3. What should your response be if you receive an email stating that you have won a lottery or received some inheritance from an unknown person ?
   (*a*) Send them email asking for further procedure
   (*b*) Ignore it
   (*c*) Mark it as spam and block it
4. A computer _____ is a malicious code which self-replicates by copying itself to other programs.
   (*a*) program (*b*) virus (*c*) application (*d*) worm
5. Which of the following is a type of program that either pretends to have, or is described as having, a set of useful or desirable features but actually contains a damaging code.
   (*a*) Trojans (*b*) Viruses (*c*) Worm (*d*) Adware
6. Which of the following is the type of software that has self-replicating software that causes damage to files and system ?
   (*a*) Trojans (*b*) Viruses (*c*) Worm (*d*) Adware
7. Which of the following tracks your internet browsing habits and sends you popups containing advertisements related to the sites and topics you've visited ?
   (*a*) Trojans (*b*) Viruses (*c*) Worm (*d*) Adware

8. What is the broad term covering computer viruses, worms, trojan, adware, etc. ?
   (*a*) Malware (*b*) Spyware (*c*) Worm (*d*) Adware
9. The attack that focuses on capturing small packets from the network transmitted by other computers and reading the data content in search of any type of information is _____ .
   (*a*) Phishing (*b*) Eavesdropping (*c*) Scams (*d*) PC intrusion
10. Which of the following is/are source of spreading viruses from one computer to another ?
    (*a*) Email (*b*) Infected data
    (*c*) Infected programs (*d*) All of the above
11. _____ are the attempts by individuals to obtain confidential information from you through an original looking site and URL.
    (*a*) Phishing scams (*b*) Spoofing (*c*) Eavesdropping (*d*) Pharming
12. _____ is an attempt where a hacker tries to divert network traffic to a bogus site.
    (*a*) Phishing scams (*b*) Spoofing (*c*) Eavesdropping (*d*) Pharming attack

Fill in the Blanks

1. _____ is a general term used to refer to viruses, worms, spyware, adware etc.
2. A _____ is a self-replicating program which eats up the entire disk space or memory.
3. _____ _____ is a program that appears harmless (such as a text editor or a utility program) but actually performs malicious functions.
4. _____ is a software which is installed on your computer to spy on your activities and report this data to people willing to pay for it.
5. _____ refers to the sending of bulk-mail by an identified or unidentified source.
6. Unauthorised monitoring of other people's communications is called_____.
7. A _____ is a small piece of data sent from a website and stored in a user's web browser (in a text file) while a user is browsing a website.
8. _____ is a system designed to prevent unauthorized access to or from a private network.

True/False Questions

1. Computer viruses are healthy codes/programs written to test the security health of a system.
2. A worm is a self-replicating program.
3. A trojan horse is a utility program that does harm when its date gets expired.
4. Spyware is healthy software which keeps an eye of activities happening on your computer.
5. Adware are the programs that deliver unwanted ads to your computer.
6. Malware is a special type of virus.
7. Spamming refers to the sending of bulk-mail by an identified or unidentified source.
8. Phishing involves authentic looking sites trying to collect your sensitive transactional and identity information.
9. To keep a computer protected, one should use updated software, be cautious while handling mails and surfing Internet, scanning computer regularly etc.
10. Firewall can be implemented in software as well as in hardware .

Solved Problems

1. *Describe why authentication is important for file protection.*

   Solution. Authentication is the process of determining whether someone is a legal user. It is the process of identifying an individual, usually based on a username and password. Authentication merely ensures that the individual is who he or she claims to be, but says nothing about the access rights of the individual. It is used a primary step for file protection from unauthorised users.

2. *Viruses, pharming and phishing are all examples of potential Internet security issues. Explain what is meant by each of these three terms.*

   Solution. **Virus** refers to a computer program/software that replicates/copies itself. It can delete or alter files/data stored on a computer and can even make the computer "crash"/run slow.

   Pharming is a way to trick unsuspecting user by inserting malicious code/software installed on a user's hard drive/actual web server. This code redirects user to a fake website (without his/her knowledge). Pharming takes place generally to obtain personal/financial information/data.

   Phishing is another way to trick unsuspecting user where legitimate-looking emails are sent to a user. As soon as recipient opens/clicks on link in the email/attachment, the user is directed to a fake website (without his/her knowledge). Like *Pharming*, *Phishing* also takes place generally to obtain personal/financial information/data.

3. *What is anti-virus software* ?

   Solution. There are a number of malicious software ('malware') programs that can cause damage to computers. These include viruses, worms, Trojan horses (Trojans), malware, spyware etc.

   Anti-virus software is designed to detect and block attacks from malware. This software when loaded, resides in memory and checks every operation if it is malicious or not. If it finds any suspicious activity, it blocks that operation and saves our computer.

4. *Internet security is a major issue for many people. The following is a list of five typical security issues:*
   - pharming
   - phishing
   - viruses
   - spyware

   For each one, describe the security issue and suggest a way of protecting against it.

   Solution.

Security Issue	Description of Security Issue	Method of Protection
Pharming	It is actually a code installed on the hard drive of a user's computer or on actual web server ; code redirects user to a bogus/fake website without user knowing.	❖ Use filters to authenticate websites. ❖ User should be alert and look for pharming clues which indicate being directed to a bogus site.
Phishing	In phishing, creator sends legitimate-looking (fake) email in the hope of gaining personal/ financial information ; fake email replicates a well known company *e.g.*, a bank.	❖ ISPs can filter/block out phishing emails. ❖ User should be aware of opening links in emails.
Spyware	It is a type of software that gathers information by monitoring key presses on a user's keyboard or activity and relays the information back to person who sent the spyware.	❖ Use dropdown boxes. ❖ User should be alert and look for clues when using their computer.
Viruses	It refers to program or coding that replicates itself / corrupts the system/alters or deletes data.	❖ Use anti-virus software. ❖ Do not use disks/software from unknown sources. ❖ Do not open emails from unknown senders.

5. *What is the need for secure passwords ?*

Solution. When more than one person uses a network, it is important to have user IDs and passwords. Only someone with a valid login and password can access that network. It also helps the network manager trace unusual activity to a specific user.

A weak password makes it easy for someone to try to guess your login details. A good password will have a mix of upper case and lower case letters, numbers and special characters. A good combination of these makes a strong password and difficult to crack it. Hence a strong password keeps our system secure.

6. *What measures should you take to keep data secure ?*

Solution. Measures that can be taken to keep data secure include :

(*i*) Making regular backups of files (backup copies should be stored in fireproof safes or in another building).

(*ii*) Protecting against viruses by running anti-virus software.

(*iii*) Using a system of strong passwords so that access to data is restricted.

(*iv*) Safe storage of important files stored on removable disks, *e.g.,* locked away in a safe location.

GLOSSARY

Phishing	Tricking users to give login ids and password by authentic looking site addresses.
Virus	A malicious code written to harm the data and applications.
Worm	Self replicating program that eats up all the memory/space.

Assignments

1. What is a virus ? What is anti-virus software ?
2. How is backup utility useful ? Is it necessary to take backup of data ?
3. What is a computer virus ? How can it affect your computer ?
4. What are different type of threats to computer security ?
5. What type damages can be caused by viruses to your computer ?
6. What are Malware ? What type of damages can they cause to your computer ?
7. What is a Spam ? Why has it become a big Internet issue ?
8. What do you understand by PC intrusion ?
9. What measure would you take to avoid :
   (*i*) Virus attack (*ii*) Spyware (*iii*) Adware (*iv*) Spam (*v*) PC intrusion ?
10. What are denial-of-service and Sweeper attacks ?
11. What is phishing ?
12. How is pharming similar to and different from phishing ?
13. What is (*i*) Authentication, (*ii*) Authorization ? Why are these two used together ?
14. What is the significance of a firewall in a computer's security scheme ?
15. What is Eavesdropping ? What security measures can you take up to prevent it ?

Appendix A

Installing MySQL Python Connectors

Prerequisites for installing MySQL Connector Python

Before you install MySQL connector on Python, you must ensure that :

(*i*) Python is installed on your computer and you are connected to the Internet.

(*ii*) Python is on system's PATH. You often check the box to include Python in system's PATH variable. If it is not, then on Windows, you can see the properties of Computer icon on desktop (right click and then click *Properties*).

Then from *Advanced system settings* → *Advanced* → *Environment variables*. Edit the PATH variable and without deleting anything add **; <pythonpath>** ; to it, *e.g.,* if python's path on your computer is **"C:\Python37-32\python.exe"** then add **; C:\ Python37-32\python.exe;** to the PATH variable.

(*iii*) You need root or administrator privileges to perform the installation process.

Installing mysql connector and pymysql on standard Python installation, using pip

MySQL Connector Python is available on *pypi.org* wherefrom it will be downloaded and installed on your computer through **pip** command. The pip command can install on any operating system platform including Windows, MacOs, Linux, and Unix and Ubuntu. The procedure is just the same but for Linux and mac you need to add **sudo apt** before the **pip** command, just the way you did in Django installation. The command to install *MySQL Connector python* and *pymysql* is :

```
pip install mysql-connector-python
pip install pymysql
```

Installing mysql Connector on Anaconda Python Installation

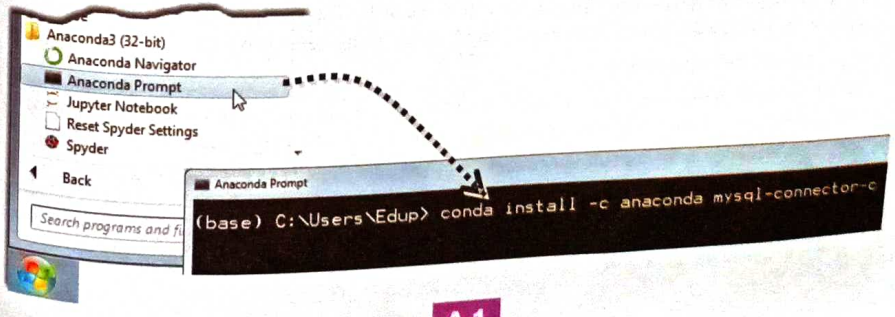
The library **pymysql** is by default installed on Anaconda.

To install mysql connector on Anaconda installation, you need to first download the installer from the link **anaconda.org/anaconda/mysql-connector-c/files, *Files* tab** as per your OS.

Then start Anaconda prompt. On Windows you do it by command :

Start menu → Anaconda → Anaconda Prompt

On the *conda prompt*, issue command : **conda install -c anaconda mysql-connector-c** and it will be installed.



Appendix B

New Style : String Templates with { } formatting

The new style of creating SQL query strings involves the use of the **.format( )** method of the str type. Let us first understand this with a normal string and then we shall apply it on SQL query strings. The general form for using **.format( )** is :

```
template.format(p0, p1, ..., k0 = v0, k1 = v1, ...)
```

The **template** is a string containing a mixture of one or more format codes embedded in constant text. The format method uses its arguments to substitute an appropriate value for each format code in the template.

The arguments to the **.format( )** method are of two types. The list starts with zero or more positional arguments p_i, followed by zero or more keyword arguments of the form $k_i v_i$ where each k_i is a name with an associated value v_i.

Consider following example. In this example, the format code "{0}" is replaced by the first positional argument (49), and "{1}" is replaced by the second positional argument, the string "okra"

These are place holders

Values tuple V. Values are substituted from here

```
"We have {0} hectares planted to {1}." .format (49, "okra")
```

The above string template will yield following string

```
'We have 49 hectares planted to okra.
```

The values are picked from the values tuple V from left to right, *i.e.*, the first value of tuple will be placed in the first { } placeholder, second value in the second { } placeholder, and so on.

You may even skip the numbers 0, 1 in { }. That is, following string will also yield the same string as above.

```
"We have {} hectares planted to {} ." .format (49, "okra")
```

In the next example, we supply the values using **named arguments**. The arguments may be supplied in any order. The keyword names must be valid Python names, *e.g.*,

Values tuple V. Values are substituted from here

```
"{monster} has eaten {city}".format(city ='Tokyo', monster ='Tsunami')
```

In the above string template **{monster}** and **{city}** are named arguments. Named arguments will replace the value of the arguments as per the names not as per the order. So even though in values tuple above, the first value is **city ='Tokyo'**, it will not replace the first placeholder**{monster}**, rather it will replace the placeholder having its name *i.e.*, **{city}**. So the string generated will be :

```
'Tsunami has eaten Tokyo
```

Based on above information, you can create your SQL query strings. Again, make sure to enclose a string value place holder in quotes, *i.e.*, as '{ },]. Now consider following example :

```
st = "select * from student where marks > {} and section = '{}' ".format(70,'B')
```

The above query string **st** stores :

Place holder enclosed in quotes for string value

```
"select * from student where marks > 70 and section = 'B' "
```

And it will yield the same query result as earlier.

Answers to Objective Type Questions (OTQs)

CHAPTER 1 : PYTHON PANDAS – I

Multiple Choice Questions

1. (c)	2. (b)	3. (d)	4. (c)	5. (b)	6. (b)
7. (a)	8. (b)	9. (a)	10. (c), (d)	11. (d)	12. (b)
13. (c)	14. (a)	15. (b)	16. (a)	17. (c)	18. (d)
19. (a)	20. (c)	21. (c)	22. (d)	23. (d)	24. (c)
25. (b)	26. (c)				

Fill in the Blanks

1. Pandas	2. series	3. DataFrame	4. NaN	5. dtype	6. len( )
7. count( )	8. value	9. size	10. size, value	11. loc	12. at
13. iat	14. inplace				

True / False Questions

1. T	2. F	3. T	4. T	5. F	6. T
7. T	8. F	9. T	10. F	11. T	12. F
13. F	14. F				

CHAPTER 2 : PYTHON PANDAS – II

Multiple Choice Questions

1. (b)	2. (d)	3. (c), (d)	4. (a), (d)	5. (b), (c)	6. (c)
7. (c)	8. (d)	9. (d)	10. (a)	11. (c)	12. (c)
13. (c)	14. (c)	15. (b)	16. (d)	17. (c)	18. (b)
19. (b)	20. (c)	21. (c)	22. (a), (b)	23. (b)	24. (c)
25. (c)					

Fill in the Blanks

1. iterrows( ), iteritems( )	2. add or radd( )	3. +
4. mode( )	5. mean( )	6. median( )
7. quantile( )	8. quantile( )	9. quantile( )
10. isnull( )	11. concat( )	12. merge( )
13. sort_values( )	14. sort_index	

True / False Questions

1. F	2. T	3. F	4. T	5. F	6. F
7. T	8. T	9. F	10. F	11. F	12. F
13. T	14. T	15. T	16. T	17. T	18. F
19. F	20. T				

CHAPTER 3 : PLOTTING WITH PYPLOT

Multiple Choice Questions

1. (d)	2. (d)	3. (b), (d)	4. (c)	5. (c)	6. (a)
7. (b)	8. (b)	9. (c)	10. (b)	11. (b), (d)	12. (b)
13. (d)	14. (d)	15. (d)	16. (a)	17. (b)	18. (b)
19. (c)	20. (a)	21. (c)	22. (c)	23. (b)	24. (c)
25. (b)	26. (d)	27. (c)	28. (b)	29. (c)	30. (c)

Fill in the Blanks

1. histogram	2. matplotlib	3. plot( )	
4. barh( )	5. scatter( )	6. hist( )	
7. markers	8. linewidth	9. linestyle	
10. width	11. xticks( )	12. savefig( )	
13. orientation	14. showmeans	15. notch	16. loc

True/False Questions

1. T	2. F	3. T	4. F	5. F	6. T
7. T	8. T	9. F	10. T	11. T	12. F
13. F	14. F	15. T			

CHAPTER 4 : IMPORTING/EXPORTING DATA BETWEEN CSV FILES/MYSQL AND PANDAS

Multiple Choice Questions

1. (f)	2. (d)	3. (d)	4. (d)	5. (b)	6. (b)
7. (a)	8. (c)	9. (b)	10. (b)	11. (a), (b)	12. (c)
13. (a)	14. (b)	15. (b)			

Fill in the Blanks

1. comma separated values	2. , (comma)	3. read_csv	4. to_csv
5. sep	6. na_rep	7. read_sql( )	8. to_sql( )

True / False Questions

1. F	2. T	3. F	4. F	5. F	6. T
7. T	8. F	9. F	10. F		

CHAPTER 5 : MYSQL SQL REVISION TOUR

Multiple Choice Questions

1. (a)	2. (c)	3. (d)	4. (a)	5. (b)	6. (a)
7. (c)	8. (a)	9. (a)	10. (c), (d)	11. (a)	

Fill in the Blanks

1. FROM	2. DISTINCT	3. asterisk (*)	4. WHERE	5. AND	6. IN
7. NOT IN	8. LIKE				

True / False Questions

1. F	2. T	3. F	4. F	5. F	6. F
7. T					

CHAPTER 6 : MYSQL FUNCTIONS

Multiple Choice Questions

1. (b)	2. (c)	3. (d)	4. (a), (c)	5. (b)	6. (b)
7. (b)	8. (c)	9. (c)	10. (a)	11. (c)	12. (a)

Fill in the Blanks

1. single row	2. multiple rows	3. mid( )	4. Day( )	5. Dayname( )
6. Rtrim( )		7. Curdate( ), Sysdate( )		

True / False Questions

1. F	2. F	3. T	4. F	5. F

CHAPTER 7 : QUERYING USING SQL

Multiple Choice Questions

1. (d)	2. (c), (d)	3. (d)	4. (d)	5. (a)	6. (c)
7. (b)	8. (d)	9. (c)	10. (b)	11. (d)	12. (a)
13. (a)	14. (b)				

Fill in the Blanks

1. ORDER BY	2. DESC	3. SUM	4. AVG	5. MAX
6. MIN	7. COUNT	8. GROUP BY		

True / False Questions

1. F	2. T	3. T	4. F	5. F	6. T
7. T	8. T	9. T	10. T		

CHAPTER 8 : JOINS AND SET OPERATIONS

Multiple Choice Questions

1. (c)	2. (b)	3. (a)	4. (b)	5. (c)	6. (a)
7. (a)	8. (b)	9. (b)	10. (a)	11. (b)	12. (b)
13. (c)	14. (c)	15. (c)			

Fill in the Blanks

1. join
2. equi-join
3. Left
4. Right
5. Inner
6. INTERSECT
7. UNION ALL
8. UNION
9. MINUS
10. INNER

True / False Questions

1. F
2. F
3. T
4. T
5. F
6. T
7. T
8. T
9. T
10. F

CHAPTER 9 : INTRODUCTION TO COMPUTER NETWORKS

Multiple Choice Questions

1. (a)
2. (a)
3. (b)
4. (b)
5. (a)
6. (b)
7. (a)
8. (b)
9. (a)
10. (b)
11. (b)
12. (d)
13. (b)
14. (d)
15. (c)
16. (a)
17. (c)
18. (c)
19. (d)
20. (b)

Fill in the Blanks

1. WAN
2. Wide Area Network
3. node/host
4. LAN
5. MAN
6. hub
7. bridge
8. router
9. gateway
10. reapter
11. bus
12. star
13. star, bus
14. mesh
15. PAN

True/False Questions

1. F
2. T
3. F
4. F
5. F
6. T
7. T
8. T
9. T
10. F
11. T
12. T
13. F
14. T
15. F

CHAPTER 10 : INTRODUCTION TO INTERNET AND WEB

Multiple Choice Questions

1. (d)
2. (a)
3. (b)
4. (c)
5. (a)
6. (c)
7. (a)
8. (b)
9. (a)
10. (b)
11. (c)
12. (a)
13. (d)
14. (c)
15. (c)
16. (c)
17. (c)
18. (d)
19. (b)

Fill in the Blanks

1. messages, information
2. hyperlinks
3. URL
4. Internet
5. WWW
6. World Wide Web
7. web page
8. home page
9. dynamic
10. static
11. web server
12. Uniform Resource Locator
13. carbon copy
14. Blind carbon copy
15. Internet Protocol
16. Voice over Internet Protocol
17. cookies

True/False Questions

1. F
2. F
3. T
4. T
5. F
6. F
7. T
8. F
9. F
10. F
11. T
12. F

CHAPTER 11 : SOCIETAL IMPACTS

Multiple Choice Questions

1. (c)
2. (a)
3. (c)
4. (b)
5. (b)
6. (c)
7. (c)
8. (d)
9. (d)
10. (c)
11. (d)
12. (b)
13. (b)
14. (c)
15. (c)
16. (d)
17. (c)
18. (b)
19. (a)

Fill in the Blanks

1. Open Source Software
2. scam
3. Plagiarism
4. digital
5. e-waste
6. MIT
7. source code
8. Freeware
9. Free software
10. Internet Addiction Disorder
11. Digital Footprint
12. Licenses
13. Copyleft
14. Repetitive Strain Injury
15. Computer Vision Syndrome

True/False Questions

1. T
2. F
3. T
4. T
5. F
6. T
7. F
8. T
9. F
10. T
11. F
12. F
13. F
14. T
15. T

CHAPTER 12 : DATA PROTECTION

Multiple Choice Questions

1. (b)
2. (a)
3. (c)
4. (d)
5. (a)
6. (c)
7. (d)
8. (a)
9. (b)
10. (d)
11. (a)
12. (d)

Fill in the Blanks

1. Malware
2. worm
3. Trojan horse
4. Spyware
5. Spamming
6. Eavesdropping
7. cookie
8. Firewall

True/False Questions

1. F
2. T
3. F
4. F
5. T
6. F
7. T
8. T
9. T
10. T

informatics practices

a text book for class XII

Informatics Practices is a major subject at Senior Secondary level of CBSE with an aim of creating a league of students well-versed with computer practices that apply information science tools and methods in various fields. The subject 'Informatics Practices' focuses on concepts of Information Science applications and the basics of a Programming Language Python with an aim to use it for the management of information effectively.

Written strictly according to CBSE syllabus of subject code 065, this book covers :

- Data Handling
- Database Query using SQL
- Computer Network
- Societal Impacts

This text book has been so designed that it contains adequate examples and practical questions so as to make the learning complete.

Keeping in sync with earlier editions, this book comes with a student-friendly presentation style so as to make learning process easier and fun to learn. This textbook – complete with numerous examples, OTQs (Objective Type Questions), solved problems, glossary and assignments – promises to guide you for the preparation for the *final examination*.

Publisher

ISBN 978-81-7700-237-9

Published by
DHANPAT RAI & CO. (P) LTD.
Educational & Technical Publishers
4576/15, Agarwal Road, Darya Ganj, New Delhi-110002
Ph.: 2324 7736, 37, 38 • E-mail: dhanpatrai@gmail.com