AIM: Hadoop Installation on Single Node

OBJECTIVE:

1. To Learn and underst and the concepts of Hadoop
2. To learn and understand the Hadoop framework for Big Data
3. To understand and practice installation and configuration of Hadoop.

SOFTWARE REQUIREMENTS:

1 Ubuntu 14.04 / 14.10
2 Java 1.7
3 Hadoop 2.9.0

THEORY:

Introduction

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models.It is designed to scaleup from single servers to thousands of machines, each offering local computation and storage.
Due to the advent of new technologies, devices, and communication likes social networking sites, the amount of data produced by mankind is growing rapidly every year.The amountof data produced by us from the beginning of time till 2003 was 5 billion gigabytes. Thesame amount was created in every two days in2011, and in every ten minutes in2013.This rate is still growing enormously. Though all this information produce dismeaningful and can be useful when processed, it is being neglected

BigData

Big data means really a big data, it is a collection of large data sets that cannot be processed using traditional computing techniques. Big data is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks. Big data involves the data produced by different devices and applications. Given below are some of the field sthat come under the umbrella of Big Data.

Hadoop

Doug Cutting, MikeCafarella and team took the solution provided by Google and started an Open Source Project called HADOO Pin2005 and Doug name ditafterhisson's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop runs applications using the MapReduce algorithm, where the datails processed in parallel on different CPU nodes. In short, Hadoop frame work is capabale enough to develop applications capable of running on clusters of computers and they could perform complete statistical analysis for huge amounts of data.

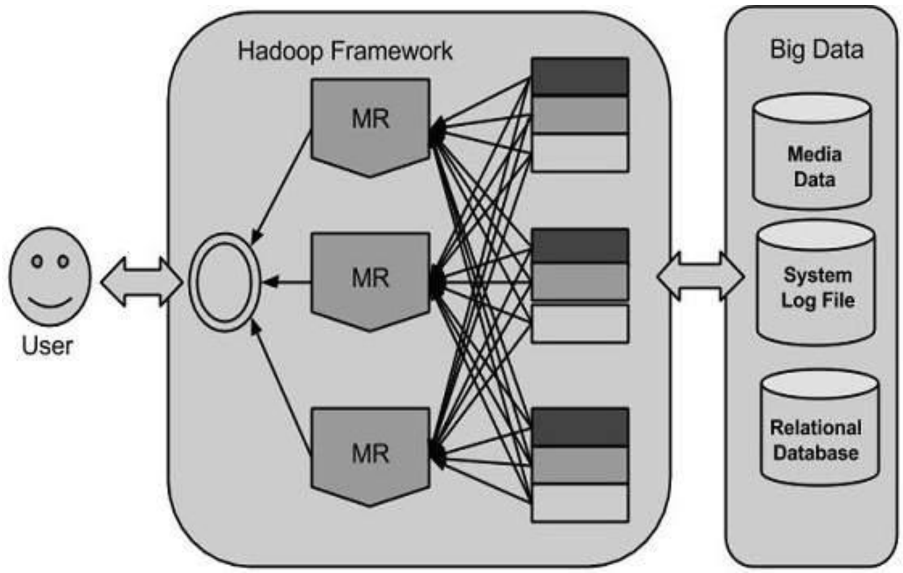


Figure 1.1: Hadoop Framework

Hadoop is an Apache open source framework written in java that allows distributed processing of large data sets across clusters of computers using simple programming models. A Hadoop frame-worked application works in an environment that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

Hadoop framework includes following four modules:

- **HadoopCommon:**These are Java libraries and utilities required by other Hadoop modules.These libraries provides file system and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.
- **Hadoop YARN:**This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System(HDFS™):** A distributed file system that provides high-through put access to application data.
- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

Installing Java

Hadoop framework is written in Java!!

Update the source list
student@student:~$**sudo apt-get update**

The OpenJDK project is the default version of Java
that is provided from a supported Ubuntu repository.

student@student:~$sudo apt-get install default-jdk

student@student:~$**java -version** java
version "1.7.0_91"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)

Create User for Hadoop
student@student:~$sudoaddgrouphadoop
student@student:~$ sudoadduser --ingrouphadoophduser
student@student:~$ sudoadduserhdusersudo
student@student:~$ sudo apt-get installopenssh-server
student@student:~$ su – hduser

Installing SSH (secure shell)

sshhas two main components:

1. **ssh**: The command we use to connect to remote machines - the client.
2. **sshd**: The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first. Use this command to do that :

student@student:~$sudo apt-get install openssh-server

Create and Setup SSH Certificates

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine.

For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

student@student:~$**ssh-keygen -t rsa -P ""** student@student:~$cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.
We can check if ssh works:

student@student:~$**sshlocalhost**

Install Hadoop

student@student:~$**wget**http://mirrors.sonic.net/apache/hadoop/common/hadoop2.6.3/hadoop-2.6.3.tar.gz

student@student:~$**tar xvzf hadoop-2.6.3.tar.gz**

We want to move the Hadoop installation to the **/usr/local/hadoop** directory using the following command:

student@student:~$sudo mv hadoop-2.9.0 /usr/local/hadoop
student@student:~$sudochown -R student /usr/local

Setup Configuration Files

The following files will have to be modified to complete the Hadoop setup:

~/.bashrc
/usr/local/hadoop/etc/hadoop/hadoop-env.sh
/usr/local/hadoop/etc/hadoop/core-site.xml
/usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/hdfs-site.xml

1. ~/.bashrc:

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA_HOME** environment variable using the following command:

Now we can append the following to the end of **~/.bashrc**:

student@student:~$**gedit .bashrc**

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

student@student:~$**source .bashrc**

This command applies the changes made in the .bashrc file.

2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh

We need to set **JAVA_HOME** by modifying **hadoop-env.sh** file.

student@student:~$**gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh**

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64 Adding the above statement in the **hadoop-env.sh** file ensures that the value of JAVA_HOME variable will be available to Hadoop whenever it is started up.

3. /usr/local/hadoop/etc/hadoop/core-site.xml:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up. This file can be used to override the default settings that Hadoop starts with.

```
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
```

4.sudogedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
```

5. sudogedit /usr/local/hadoop/etc/hadoop/yarn-site.xml

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
```

```
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

6 sudogedit /usr/local/hadoop/etc/hadoop/mapred-site.xml

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

student@student:~$sudomkdir -p /usr/local/hadoop_tmp
student@student:~$sudomkdir -p student@student:~$/usr/local/hadoop_tmp/hdfs/namenode
student@student:~$sudomkdir -p student@student:~$/usr/local/hadoop_tmp/hdfs/datanode
student@student:~$sudochown -R hduser /usr/local/hadoop_tmp

Open the file and enter the following content in between the
Format the New HadoopFilesystem

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current**directoryunder **/usr/local/hadoop_store/hdfs/namenode** folder:

student@student:~$**hdfsnamenode -format**

DEPRECATED: Use of this script to execute hdfs command is
Note that **hadoopnamenode -format** command should be executed once before we start using Hadoop. If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

<u>Starting Hadoop</u>

Now it's time to start the newly installed single node cluster. We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

student@student:~$**start-all.sh**

We can check if it's really up and running:

student@student:~$**jps**
9026 NodeManager
7348 NameNode
9766 Jps
8887 ResourceManager
7507 DataNode

The output means that we now have a functional instance of Hadoop running on our VPS

2)ASSIGNMENT TITLE: Design a distributed application using MapReduce which processes a log file of a system.

(Virtual private server).

Hadoop Web Interfaces

Let's start the Hadoop again and see its Web UI:

Accessing HADOOP through browser

http://localhost:50070/

Verify all applications for cluster

http://localhost:8088/

CONCLUSION:

We studied installation of Hadoopinstallation and configuration.

OBJECTIVE:

1. To explore different Big data processing techniques with use cases.
2. To study detailed concept of Map-Reduced.

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. Java 1.7
3. Hadoop 2.9.0

PROBLEM STATEMENT: - Design a distributed application using MapReduce which processes a log file of a system. List out the users who have logged for maximum period on the system. Use simple log file from the Internet and process it using a pseudo distribution mode on Hadoop platform.

THEORY:

Introduction

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a program model for distributed computing based on java.

The MapReduce algorithm contains two important tasks, namely Map and Reduce.Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.

Under the MapReduce model, the data processing primitives are called mappers and reducers.
Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm:

MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

Mapstage: The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

Reduce stage: This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

Inserting Data into HDFS:

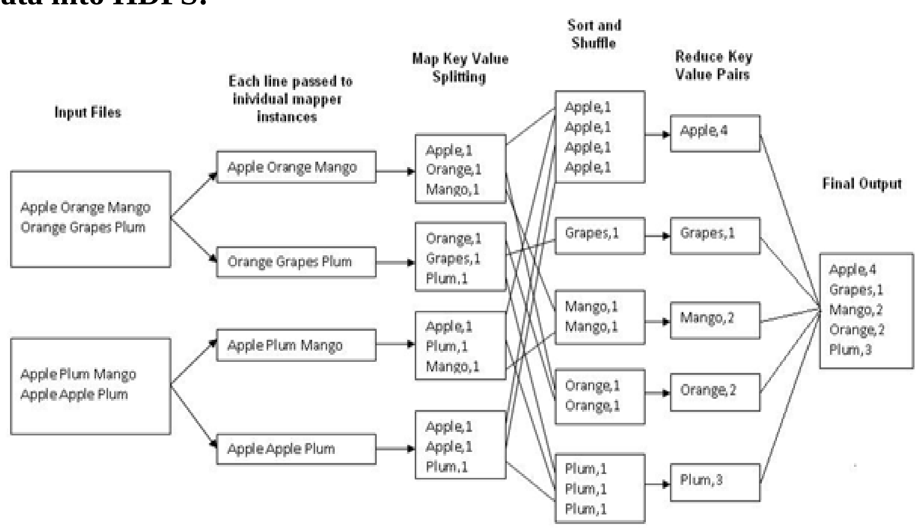


Figure 2.1: An Example Program to Understand working of MapReduce Program.

•The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

Steps for Compilation & Execution of Program:

student@student-HP-Compaq-4000-Pro-SFF-PC:~$ su - hduser

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ start-dfs.sh
hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ start-yarn.sh

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ jps

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudomkdir max

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudochmod -R 777 max/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudochown -R hduser max/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudocp /home/student/Desktop/max/* ~/max/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ cd max

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/max$ ls

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/max$ sudomkdir ~/input81

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/max$ sudocp sample.txt ~/input81/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/max$ $HADOOP_HOME/bin/hdfsdfs -put ~/input81 /

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/max$ javac Cyber.java -classpath hadoop-core-1.2.1.jar

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/max$ cd ..

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudochown -R hduser max/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ jar -cvf max.jar -C max/ .

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ $HADOOP_HOME/bin/hadoop jar max.jar Cyber /input2 /output2

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ $HADOOP_HOME/bin/hdfsdfs -cat /output81/part-00000

OUTPUT:
Ashok 6
Heena 6
Kunda 8
Pratap 7
Raman 7
Rohan 6

CONCLUSION: Thus we have learnt how to design a distributed application using MapReduce and process a log file of a system.

ASSIGNMENT TITLE: Design and develop a distributed application to find the coolest/hottest year from the available weather data.

OBJECTIVE:

1. To explore different Big data processing techniques with use cases.
2. To study detailed concept of Map-Reduced.

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. Java 1.7
3. Hadoop 2.9.0

PROBLEM STATEMENT: -Design and develop a distributed application to find the coolest/hottest year from the available weather data. Use weather data from the Internet and process it using MapReduce.

THEORY:
MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.

Map takes a set of data and converts it into another set of data, where individual elementsare broken down into tuples (key/value pairs).Secondly, reduce task, which takes the outputfrom a map as an input and combines thosedata tuples into a smaller set of tuples. As thesequence of the name MapReduce implies, thereduce task is always performed after the map
job. The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.
MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
Map stage: The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.Reduce stage: This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

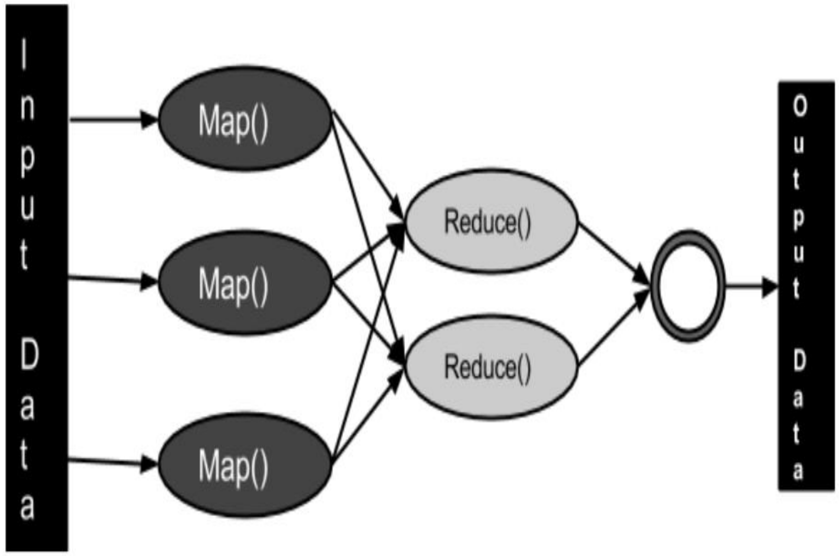


Figure 3.1 Working of Map and Reduce Method

Map Function – It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

Example – (Map function in Word Count)

Input	Set of data	Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN
Output	Convert into another set of data (Key,Value)	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

Reduce Function – Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

Example – (Reduce function in Word Count)

Input **(output of Map function)**	Set of Tuples	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)
Output	Converts into smaller set of tuples	(BUS,7), (CAR,7), (TRAIN,4)

Work Flow of Program:

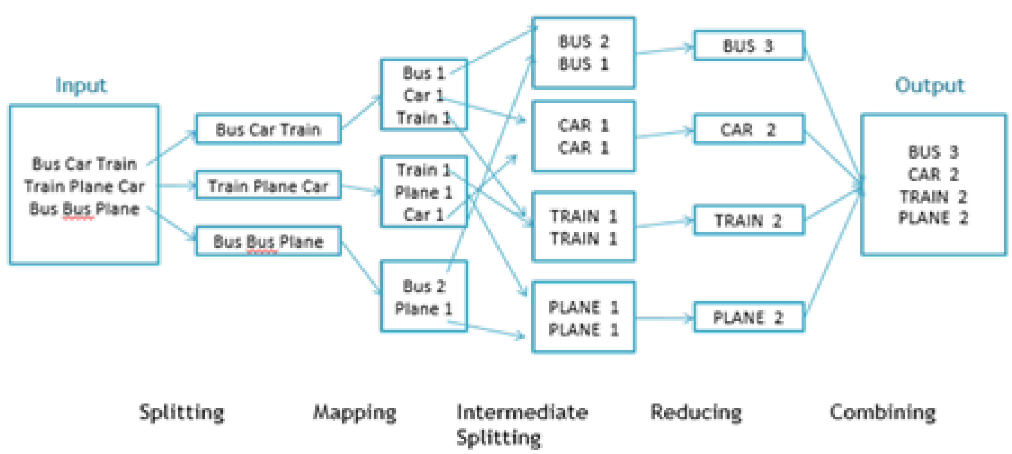


Figure 3.2: Workflow of MapReduce

Workflow of MapReduce consists of 5 steps

1. **Splitting** – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line (‘\n’).
2. **Mapping** – as explained above
3. **Intermediate splitting** – the entire process in parallel on different clusters. In order to group them in “Reduce Phase” the similar KEY data should be on same cluster.
4. **Reduce** – it is nothing but mostly group by phase
5. **Combining** – The last phase where all the data (individual result set from each cluster) is combine together to form a Result**.**

Steps for Compilation & Execution of Program:

student@student-HP-Compaq-4000-Pro-SFF-PC:~$ su - hduser

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudomkdir temp

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudochmod -R 777 temp/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudochown -R hduser temp/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudocp /home/student/Desktop/temp-std/* ~/temp/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ cd temp

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ ls

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ cd

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ start-dfs.sh

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ start-yarn.sh

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ jps

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ cd temp

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ sudomkdir ~/input11
hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ sudocp temp.txt ~/input11/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ $HADOOP_HOME/bin/hdfsdfs -put ~/input11 /

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ javac MyMaxMin.java -classpath hadoop-core-1.2.1.jar

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ ls

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~/temp$ cd ..

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ sudochown -R hduser temp/

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ jar -cvf temp.jar -C temp/ .

hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ $HADOOP_HOME/bin/hadoop jar temp.jar MyMaxMin /input11 /output11 hduser@student-HP-Compaq-4000-Pro-SFF-PC:~$ $HADOOP_HOME/bin/hdfsdfs -cat /output11/part-r-00000

OUTPUT:

Cold Day 20150101 -0.6
Cold Day 20150102 1.3
Cold Day 20150103 2.3
Cold Day 20150104 -1.3
.
.
.
.
.
Hot Day 20150803 36.5
Hot Day 20150804 36.5
Hot Day 20150805 37.3
Hot Day 20150806 37.7
Hot Day 20150807 37.8

CONCLUSION:

Thus we have learnt how to design a distributed application using MapReduce and process a Dataset.

OBJECTIVE:

1.To learn NoSQL Databases (Open source) such as Hive/ Hbase
2. To study detailed concept HIVE .

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. Java 1.7
4. Hadoop 2.9.0
5. HIVE

PROBLEM STATEMENT: -Write an application using HBase and HiveQL for flight information system which will include-
1) Creating, Dropping, and altering Database tables
2) Creating an external Hive table to connect to the HBase for Customer Information Table
3) Load table with data, insert new values and field in the table, Join tables with Hive
4) Create index on Flight information Table
5) Find the average departure delay per day in 2008.

THEORY:
Hive:

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

- Prerequisites: Core Java,
    - ❑ Database concepts of SQL,
    - ❑ Hadoop File system, and any of Linux operating system
- Features:
    - It stores schema in a database and processed data into HDFS.
    - It is designed for OLAP.
    - It provides SQL type language for querying called HiveQL or HQL.
    - It is familiar, fast, scalable, and extensible.

- **Architecture:**

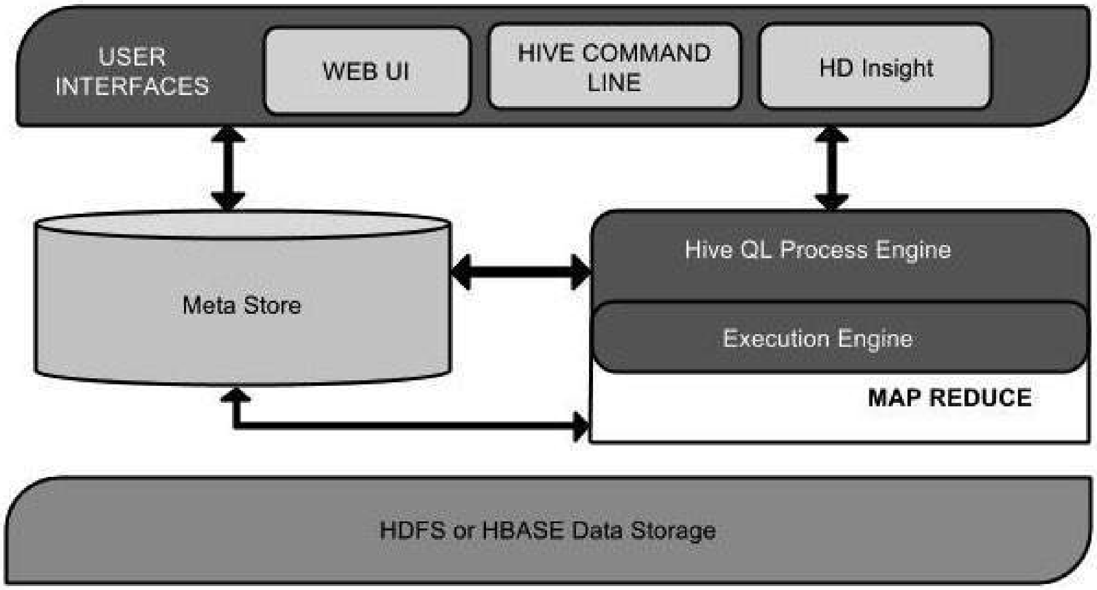


Figure 1: Hive Architecture

This component diagram contains different units. The following table describes each unit:

Unit Name	Operation
User Interface	Hive is a data warehouse infrastructure software that can create interaction between user and HDFS. The user interfaces that Hive supports are Hive Web UI, Hive command line, and Hive HD Insight (In Windows server).
Meta Store	Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.
HiveQL Process Engine	HiveQL is similar to SQL for querying on schema info on the Metastore. It is one of the replacements of traditional approach for MapReduce program. Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.

Execution Engine	The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine. Execution engine processes the query and generates results as same as MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Table 4.1: Hive components and their operations.

Sample database operations:

```
hive> create database db1;
OK
Time taken: 0.069 seconds
hive> use db1;
OK
Time taken: 0.012 seconds
hive> create table flight (fno int, year int, dest varchar(10),
      delay float);
OK
Time taken: 0.191 seconds
hive> alter table flight rename to air_flight;
OK
Time taken: 0.897 seconds
```

More alter commands:

```
hive> alter table air_flight add columns (source varchar(10));
OK
Time taken: 0.247 seconds
hive> alter table air_flight change source src varchar(15);
OK
Time taken: 0.244 seconds
```

```
hive> drop table flight;
OK
Time taken: 0.288 seconds
hive>
```

Start using Hive command line:

```
~$ jps
9669 NodeManager
9209 DataNode
9065 NameNode
10219 Jps
9549 ResourceManager
9391 SecondaryNameNode
$ hive

Logging initialized using configuration in jar:file:/usr/local/hive
e-common-1.2.1.jar!/hive-log4j.properties
hive> create database mydb;
OK
Time taken: 1.528 seconds
hive> use mydb;
OK
Time taken: 0.109 seconds
```

Create the table:

```
hive> create table flight (fno int, year int, dest varchar(10),
      delay float);
OK
Time taken: 2.082 seconds
hive> desc flight;
OK
fno                     int

year                    int

dest                    varchar(10)

delay                   float

Time taken: 0.877 seconds, Fetched: 4 row(s)
```

Table creating methodology:

```
hive> create table flight (fno int, year int, dest varchar(10), delay float)
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.691 seconds
hive>
```

Insert the values:

```
hive> insert into flight values (123, 2009, "Mumbai", 30.0);
Query ID = mitu_20180328120405_5ac7f04f-19ba-413f-827d-d311a611dd51
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Job running in-process (local Hadoop)
2018-03-28 12:04:20,804 Stage-1 map = 0%,  reduce = 0%
2018-03-28 12:04:21,845 Stage-1 map = 100%,  reduce = 0%
Ended Job = job_local1290390528_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://localhost:54310/user/hive/warehouse/mydb.db/flight/.hive-
staging_hive_2018-03-28_12-04-05_186_4559567806608736141-1/-ext-10000
Loading data to table mydb.flight
```

Insert queries:

- insert into flight values (123, 2009, "Mumbai", 30.0);
- insert into flight values (342, 2008, "Nagpur", 13.0);
- insert into flight values (232, 2008, "Aurangabad", 0.0);
- insert into flight values (103, 2009, "Kolhapur", 10.0);
- insert into flight values (200, 2008, "Jalgaon", 50.0);
- insert into flight values (112, 2009, "Amravati", 0.0);

Show table contents:

```
hive> select * from flight;
OK
123     2009    Mumbai  30.0
342     2008    Nagpur  13.0
232     2008    Aurangabad      0.0
103     2009    Kolhapur        10.0
200     2008    Jalgaon 50.0
112     2009    Amravati        0.0
Time taken: 0.661 seconds, Fetched: 6 row(s)
hive>
```

Loading a text data locally:

flight_data.txt

```
923,2009,Navi Mumbai,60.0
156,2009,Kolhapur,30.0
112,2009,Amravati,0.0
322,2008,Nagpur,0.0
132,2008,Aurangabad,10.0
170,2008,Jalgaon,40.0
```

```
hive> load data local inpath "flight_data.txt"
    > overwrite into table flight;
Loading data to table mydb.flight
Table mydb.flight stats: [numFiles=1, numRows=0, totalSize=138, rawDataSize=0]
OK
Time taken: 0.683 seconds
hive> select * from flight;
OK
923     2009    Navi Mumba      60.0
156     2009    Kolhapur        30.0
112     2009    Amravati        0.0
322     2008    Nagpur  0.0
132     2008    Aurangabad      10.0
170     2008    Jalgaon 40.0
Time taken: 0.038 seconds, Fetched: 6 row(s)
hive>
```

Creating a new table:

```
hive> create table nflight (fno int, year int, source varchar(10))
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.48 seconds
hive>
```

```
hive> select * from nflight;
OK
112     2007    Pune
322     2009    Pune
170     2009    Pune
Time taken: 0.166 seconds, Fetched: 3 row(s)
hive>
```

Joining the tables:

```
hive> select a.fno, a.year, a.dest, a.delay, b.source
    > from flight a join nflight b
    > on (a.fno = b.fno);
Query ID = mitu_20180328134721_284e7df1-a0b0-40d3-9b8c-8e345a7488
Total jobs = 1
2018-03-28 13:47:30,968 WARN  [main] util.NativeCodeLoader: Unabl
e-hadoop library for your platform... using builtin-java classes
e
Execution log at: /tmp/mitu/mitu_20180328134721_284e7df1-a0b0-40d
8821.log
```

```
Total MapReduce CPU Time Spent: 0 msec
OK
112     2009    Amravati        0.0     Pune
322     2008    Nagpur  0.0     Pune
170     2008    Jalgaon 40.0    Pune
Time taken: 16.548 seconds, Fetched: 3 row(s)
hive>
```

Creating an index:

•An Index is nothing but a pointer on a particular column of a table.
•Creating an index means creating a pointer on a particular column of a table.

```
hive> create index flight_index on table flight(fno)
    > as 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
    > WITH DEFERRED REBUILD;
OK
Time taken: 0.988 seconds
hive>
```

```
hive> show tables;
OK
flight
mydb__flight_flight_index__
nflight
values__tmp__table__1
values__tmp__table__10
values__tmp__table__11
```

- Find the average departure delay per day in 2008.

Example:select avg(delay) from flight where year = 2008;

```
hive> select avg(delay) from flight where year = 2008;
Query ID = mitu_20180328140022_2c343f4a-781a-47ce-88fd-1cbacd629087
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2018-03-28 14:00:24,656 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local843554894_0017
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 4564 HDFS Write: 3800 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
16.666666666666668
```

CONCLUSION:

Thus we have learnt how to designa application HBase and HiveQL for flight information system.

OBJECTIVE:

1. To understand and apply the Analytical concept of Big data using R/Python.
2. To study detailed concept R .

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. Java 1.7
3. R Studio 1.0.44-amd64.deb

PROBLEM STATEMENT: Perform the following operations using R/Python on the Amazon book review and facebook metrics data sets
1) Create data subsets
2) Merge Data
3) Sort Data
4) Transposing Data
5) Melting Data to long format
6) Casting data to wide format

THEORY:

R was initially written by **Ross Ihaka** and **Robert Gentleman** at the Department of Statistics of the University of Auckland in Auckland, New Zealand. R made its first appearance in 1993.

1. A large group of individuals has contributed to R by sending code and bug reports.
2. Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive

Features of R

As stated earlier, R is a programming language and software environment for statistical analysis, graphics representation and reporting. The following are the important features of R –

- R is a well-developed, simple and effective programming language which includes conditionals, loops, user defined recursive functions and input and output facilities

- R has an effective data handling and storage facility,
- R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
- R provides a large, coherent and integrated collection of tools for data analysis.
- R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

R - Data Reshaping

Data Reshaping in R is about changing the way data is organized into rows and columns. Most of the time data processing in R is done by taking the input data as a data frame. It is easy to extract data from the rows and columns of a data frame but there are situations when we need the data frame in a format that is different from format in which we received it. R has many functions to split, merge and change the rows to columns and vice-versa in a data frame.

1. **Import the dataset:**

read.csv()–Reads a csv file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

```
> d = read.csv("fb.csv")          Reads csv file
> dim(d)
[1] 500  19
> ncol(d)                         No. of columns
[1] 19
> nrow(d)                         No. of rows
[1] 500
> head(d)                         First six entries
  Page.total.likes   Type Category Post.Month
1           139441  Photo        2         12
2           139441 Status        2         12
3           139441  Photo        3         12
4           139441  Photo        2         12
5           139441  Photo        2         12
6           139441 Status        2         12
```

2. **Creating the subset:**

```
> sub = d[c('Category','comment','like','share')]
> head(sub)
  Category comment like share
1        2       4   79    17
2        2       5  130    29
3        3       0   66    14
4        2      58 1572   147
5        2      19  325    49
6        2       1  152    33
> write.csv(sub,"sub.csv")
```

Create subset

Store in csv file

subset( )–Return subsets of vectors, matrices or data frames which meet conditions.– Usage:subset(x, ...)## Default S3 method:subset (x, subset, ...)## S3 method for class 'matrix'subset (x, subset, select, drop = FALSE, ...)## S3 method for class 'data.frame'subset (x, subset, select, drop = FALSE, ...)

```
> sub2 = subset(sub, comment > 50)
> sub2
    Category comment like share
4          2      58 1572   147
143        2      60  859    90
169        1     144 1622   208
229        2      64  367    25
245        2     372 5172   790
289        1     103  469    33
380        3      51 1998   128
461        3     146 1546   181
481        2      56  360    99
```

Subset condition

3. Merge Datasets:

```
> first = read.csv("fb.csv")
> second = read.csv("newfb.csv")
> dim(first)
[1] 500  19
> dim(second)
[1] 134  19
> newdata = rbind(first, second)
> dim(newdata)
[1] 634  19
```

Shows the dimensions (rows, columns

Binds both data frames

4. Sort Datasets:

```
> d = read.csv("fb.csv")
> sub = d[c('Category','like','comment','share')]
> x = sub[order(-d$share),]
> head(x)
    Category like comment share
245        2 5172     372   790
169        1 1622     144   208
461        3 1546     146   181
4          2 1572      58   147
106        1  955      42   139
380        3 1998      51   128
>
```

Sort by share
- for descending

5. Transpose Datasets:

```
> d = read.csv("fb.csv")
> sub = d[c('Category','like','comment','share')]
> tran = t(sub)
> head(tran)
         [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
Category    2    2    3    2    2    2    3    3    2     3
like       79  130   66 1572  325  152  249  325  161   113
         [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18]
Category     2     2     2     2     2     2     3     1
like       233    88    90   137   577    86    40   678
         [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
```

Finds the transpose

6. **Melt data to long format:**

```
> d = read.csv("fb.csv")
> sub = d[c('Category','like','comment','share')]
> melt(data = sub, id.vars = "Category")
  Category variable value
1        2     like    79
2        2     like   130
3        3     like    66
4        2     like  1572
5        2     like   325
6        2     like   152
7        3     like   249
8        3     like   325
```

Melt the dataset

7. **Casting dataset:**

```
> d = read.csv("fb.csv")
> sub = d[c('Category','Post.Month','Post.Hour','Paid')]
> head(sub)
  Category Post.Month Post.Hour Paid
1        2         12         3    0
2        2         12        10    0
3        3         12         3    0
4        2         12        10    1
5        2         12         3    0
6        2         12         9    0
> cast(sub, Category ~ Post.Month, mean, value = 'Paid')
  Category         1         2         3         4
1        1 0.3333333 0.1666667 0.2580645 0.3181818
2        2        NA 1.0000000 0.0000000 0.6000000
3        3 0.1333333 0.2727273 0.0000000 0.4347826
```

CONCLUSION: Thus we have learnt how to Perform the different Data Cleaning and Data modeling operations using R **.**

ASSIGNMENT TITLE: Perform the following operations using R/Python on the Air quality and Heart Diseases data sets.

OBJECTIVE:

1. To understand and apply the Analytical concept of Big data using R/Python.
2. To study detailed concept RHadoop.

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. Java 1.7
3. R Studio 1.0.44-amd64.deb

PROBLEM STATEMENT: Perform the following operations using R/Python on the Air quality and Heart Diseases data sets

1) Data cleaning
2) Data integration
3) Data transformation
4) Error correcting
5) Data model building

THEORY:

Data cleaning or data preparation is an essential part of statistical analysis. In fact, in practice it is often more time-consuming than the statistical analysis itself

1. **The dataset:**

```
> airquality
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
```

2. **Errors and corrections:**

```
> mean(airquality$Ozone)
[1] NA
> mean(airquality$Ozone, na.rm = TRUE)
[1] 42.12931
> max(airquality$Solar.R)
[1] NA
> max(airquality$Solar.R, na.rm = TRUE)
[1] 334
>
```

NA – Error

Correct the error

3. **Check the summary:**

```
> summary(airquality)
     Ozone           Solar.R           Wind             Temp      
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00  
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88  
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00  
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
 NA's   :37       NA's   :7                                       
     Month            Day      
 Min.   :5.000   Min.   : 1.0  
 1st Qu.:6.000   1st Qu.: 8.0  
 Median :7.000   Median :16.0  
 Mean   :6.993   Mean   :15.8  
 3rd Qu.:8.000   3rd Qu.:23.0  
 Max.   :9.000   Max.   :31.0  
```

4. **Data Cleaning – Removing NAs:**

```
> air = airquality
> air$Ozone = ifelse(is.na(air$Ozone), median(air$Ozone,
 na.rm = TRUE), air$Ozone)
> summary(air)
     Ozone           Solar.R           Wind       
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700  
 1st Qu.: 21.00   1st Qu.:115.8   1st Qu.: 7.400  
 Median : 31.50   Median :205.0   Median : 9.700  
 Mean   : 39.56   Mean   :185.9   Mean   : 9.958  
 3rd Qu.: 46.00   3rd Qu.:258.8   3rd Qu.:11.500  
 Max.   :168.00   Max.   :334.0   Max.   :20.700  
                  NA's   :7                       
```


```
> air = airquality
> air$Ozone = ifelse(is.na(air$Ozone), median(air$Ozone,
 na.rm = TRUE), air$Ozone)
> summary(air)
     Ozone           Solar.R           Wind       
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700  
 1st Qu.: 21.00   1st Qu.:115.8   1st Qu.: 7.400  
 Median : 31.50   Median :205.0   Median : 9.700  
 Mean   : 39.56   Mean   :185.9   Mean   : 9.958  
 3rd Qu.: 46.00   3rd Qu.:258.8   3rd Qu.:11.500  
 Max.   :168.00   Max.   :334.0   Max.   :20.700  
                  NA's   :7                       
```


5. Data Transformation:

```
> head(air)
  Ozone Solar.R Wind Temp Month Day
1  41.0     190  7.4   67     5   1
2  36.0     118  8.0   72     5   2
3  12.0     149 12.6   74     5   3
4  18.0     313 11.5   62     5   4
5  31.5     205 14.3   56     5   5
6  28.0     205 14.9   66     5   6
> air$Solar.Danger = air$Solar.R > 100
> head(air)
  Ozone Solar.R Wind Temp Month Day Solar.Danger
1  41.0     190  7.4   67     5   1         TRUE
2  36.0     118  8.0   72     5   2         TRUE
3  12.0     149 12.6   74     5   3         TRUE
4  18.0     313 11.5   62     5   4         TRUE
5  31.5     205 14.3   56     5   5         TRUE
6  28.0     205 14.9   66     5   6         TRUE
```

Added a column

```
> brks = c(0,50,100,150,200,250,300,350)
> air$Solar.R = cut(air$Solar.R, breaks = brks,
include.lowest = TRUE)
> head(air)
  Ozone   Solar.R Wind Temp Month Day
1  41.0 (150,200]  7.4   67     5   1
2  36.0 (100,150]  8.0   72     5   2
3  12.0 (100,150] 12.6   74     5   3
4  18.0 (300,350] 11.5   62     5   4
5  31.5 (200,250] 14.3   56     5   5
6  28.0 (200,250] 14.9   66     5   6
```

```
> air1 = air
> air1$Month = gsub(5,"May",air1$Month)
> air1$Month = gsub(6,"June",air1$Month)
> air1$Month = gsub(7,"July",air1$Month)
> air1$Month = gsub(8,"Aug",air1$Month)
> air1$Month = gsub(9,"Sept",air1$Month)
> head(air1)
  Ozone   Solar.R Wind Temp Month Day
1  41.0 (150,200]  7.4   67   May   1
2  36.0 (100,150]  8.0   72   May   2
3  12.0 (100,150] 12.6   74   May   3
4  18.0 (300,350] 11.5   62   May   4
5  31.5 (200,250] 14.3   56   May   5
6  28.0 (200,250] 14.9   66   May   6
>
```

6. Data integration

We can join multiple vectors to create a data frame using the **cbind()**function. Also we can merge two data frames using **rbind()** function.

```
># make two vectors and combine them as columns in a data.frame
>sport<- c("Hockey", "Baseball", "Football")
>league<- c("NHL", "MLB", "NFL")
>trophy<- c("Stanley Cup", "Commissioner's Trophy",
+           "Vince Lombardi Trophy")
> trophies1 <- cbind(sport, league, trophy)
># make another data.frame using data.frame()
> trophies2 <- data.frame(sport=c("Basketball", "Golf"),
+                        league=c("NBA", "PGA"),
+                        trophy=c("Larry O'Brien Championship
                           Trophy",
+                               "Wanamaker Trophy"),
+                        stringsAsFactors=FALSE)
># combine them into one data.frame with rbind
>trophies<- rbind(trophies1, trophies2)
```

7. Data Model Building:

```
>air=airquality
>head(air)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> x=air$Ozone
> y=air$Solar.R
>model=lm(y ~ x)
> d=data.frame(x = 56)
>predict(model,d)
       1
198.0661
```

CONCUSION: Thus we have learnt how to Perform the different Data Cleaning and Data modeling operations using R **.**

ASSIGNMENT TITLE: Integrate R/Python and Hadoop and perform the following operations on forest fire dataset

OBJECTIVE:

1.To understand and apply the Analytical concept of big data using R/Python.
2. To study detailed concept RHadoop.

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. Java 1.7
3. R Studio 1.0.44-amd64.deb
4. Hadoop 2.9.0
5. Hive

PROBLEM STATEMENT: perform the following operations on forest fire dataset

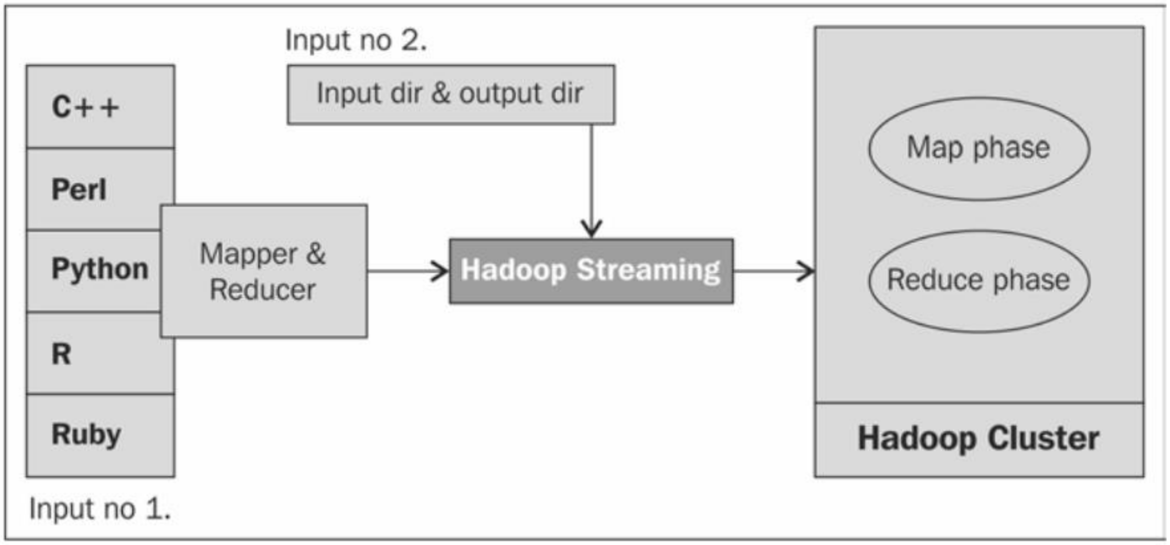
1) Text mining in RHadoop
2) Data analysis using the Map Reduce in Rhadoop
3) Data mining in Hive

THEORY:

Text mining, Data analysis/mining is an essential part of statistical analysis. In fact, in practice it is often more time-consuming than the statistical analysis itself.

Hadoop streaming: is a Hadoop utility for running the Hadoop MapReduce job with executable scripts such as Mapper and Reducer.

• This is similar to the pipe operation in Linux.
• With this, the text input file is printed on stream (stdin), which is provided as an input to Mapper and the output (stdout) of Mapper is provided as an input to Reducer; finally, Reducer writes the output to the HDFS directory.
• The main advantage of the Hadoop streaming utility is that it allows Java as well as non-Java programmed MapReduce jobs to be executed over Hadoop clusters.
• Also, it takes care of the progress of running MapReduce jobs.
• The Hadoop streaming supports the Perl, Python, PHP, R, and C++ programming languages.
• To run an application written in other programming languages, the developer just needs to translate the application logic into the Mapper and Reducer sections with the key and value output elements.



1) Text mining in RHadoop:

- Data analysis using the Map Reduce in Rhadoop

- Use dataset: forestfire.csv

$ cat forestfire.csv

month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0	0
Oct	tue	90.6	35.4	669.1	6.7	18	33	0.9	0	0
Oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0	0
mar	fri	91.7	33.3	77.5	9	8.3	97	4	0.2	0

- **Rscript of Mapper:** Mapper maps the input key/value pairs to a set of intermediate key/value pair.

$ gedit mapper.r

```
#!/usr/bin/Rscript
trimWhiteSpace<- function(line) gsub("(^ +)|( +$)", "", line)
splitIntoWords<- function(line) unlist(strsplit(line, "[[:space:]]+"))

## **** could wo with a single readLines or in blocks
con<- file("stdin", open = "r")
while (length(line <- readLines(con, n = 1, warn = FALSE)) > 0)
{
line<- trimWhiteSpace(line)
words<- splitIntoWords(line)
## **** can be done as cat(paste(words, "\t1\n", sep=""), sep="")

for (w in words)
```

```
{
        if (w=='sep' || w=='oct' || w=='jan' || w=='feb' || w=='mar' || w=='apr')
        cat(w, "\t1\n", sep="")
}
}
close(con)
```

- **Rscript of Reducer:**This stage is the combination of the **Shuffle**stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper.

- **$ gedit** reducer.r

```
#!/usr/bin/Rscript
trimWhiteSpace<- function(line) gsub("(^ +)|( +$)", "", line)
splitLine<- function(line)
{
        val<- unlist(strsplit(line, "\t"))
list(word = val[1], count = as.integer(val[2]))
}

env<- new.env(hash = TRUE)

con<- file("stdin", open = "r")
while (length(line <- readLines(con, n = 1, warn = FALSE)) > 0)
{
line<- trimWhiteSpace(line)
split<- splitLine(line)
word<- split$word
count<- split$count
        if (exists(word, envir = env, inherits = FALSE))
        {
        oldcount<- get(word, envir = env)
        assign(word, oldcount + count, envir = env)
   }
else assign(word, count, envir = env)
}
close(con)

for (w in ls(env, all = TRUE))
cat(w, "\t", get(w, envir = env), "\n", sep = "")
```

- **Execution of maaper.r and reducer.r**

$ **cat input.txt | Rscriptmapper.r | Rscriptreducer.r**

Output: mar 2
oct 2

2. Data analysis using the Map Reduce in Rhadoop

Steps: 1 //Copy dataset **forestfire.csv, mapper.r**and**reducer.r** into /home/hduser directory

$ suhduser
Passwd:
$ start-all.sh
$ jps
$ HADOOP_HOME/bin/hdfsdfs -mkdir /input11
$ HADOOP_HOME/bin/hdfsdfs -put **forestfire.csv** /input11/

Step: 2 // Execute hadoop-streaming2.9.0.jar JAR file

$ HADOOP_HOME jar /usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-2.9.0.jar -file mapper.r -mapper mapper.r -file reducer.r --reducer reducer.r -input /inp –output /out

$ HADOOP_HOME/bin/hdfsdfs-ls /out/
$ HADOOP_HOME/bin/hdfsdfs-cat /out/part-r-00000

3. Data mining in Hive:

- Problem Statement:

\- Find the maximum temperature per month, when there was a forest fire.

1. **Create the table in hive:**

```
hive> create table forest (month varchar(5), day varchar(5),
    > temp float, wind float, rain float, area float)
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 1.213 seconds
hive>
```

2. **Load the data :**

```
hive> load data local inpath "forest.csv"
    > overwrite into table forest;
Loading data to table mydb.forest
Table mydb.forest stats: [numFiles=1, numRows=0, totalSize=13656, ra
wDataSize=0]
OK
Time taken: 1.736 seconds
hive>
```

```
mitu@skillologies: ~
hive> select * from forest;
OK
 mont     day    NULL    NULL    NULL    NULL
 mar      fri    8.2     6.7     0.0     0.0
 oct      tue    18.0    0.9     0.0     0.0
 oct      sat    14.6    1.3     0.0     0.0
 mar      fri    8.3     4.0     0.2     0.0
```

3. **Execute the query:**

```
hive>
hive> select month, max(temp) from forest group by month;
Query ID = mitu_20180328163347_1d7e5569-b3ec-43e1-90ba-d584ac1e5e8a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size
: 1
```

```
Total MapReduce CPU Time Spent: 0 msec
OK
 apr    17.6
 aug    33.3
 dec    5.1
 feb    15.7
 jan    5.3
 jul    30.2
 jun    28.0
 mar    18.8
 may    18.0
 mont   NULL
 nov    11.8
 oct    21.7
 sep    30.2
Time taken: 1.923 seconds, Fetched: 13 row(s)
hive>
```

CONCUSION: Thus we have learnt how to perform the Text mining operation using R and Data mining operation using hive and hadoop.

ASSIGNMENT TITLE: Visualize the data using R/Python by plotting the graphs for assignment no. 6 and 7

OBJECTIVE:

- To understand and apply the visualize the data using R/Python by plotting the graphs

SOFTWARE REQUIREMENTS:

1. Ubuntu 14.04 / 14.10
2. Java 1.7
3. R Studio 1.0.44-amd64.deb

PROBLEM STATEMENT: perform the following operations on forest fire dataset
1) Text mining in RHadoop
2) Data analysis using the Map Reduce in Rhadoop
3) Data mining in Hive

THEORY:

1. R - Line Graphs

A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) value. Line charts are usually used in identifying the trends in data.

The **plot()** function in R is used to create the line graph.

- *Syntax:* *The basic syntax to create a line chart in R is −*

```
plot(v,type,col,xlab,ylab)
```

Following is the description of the parameters used −

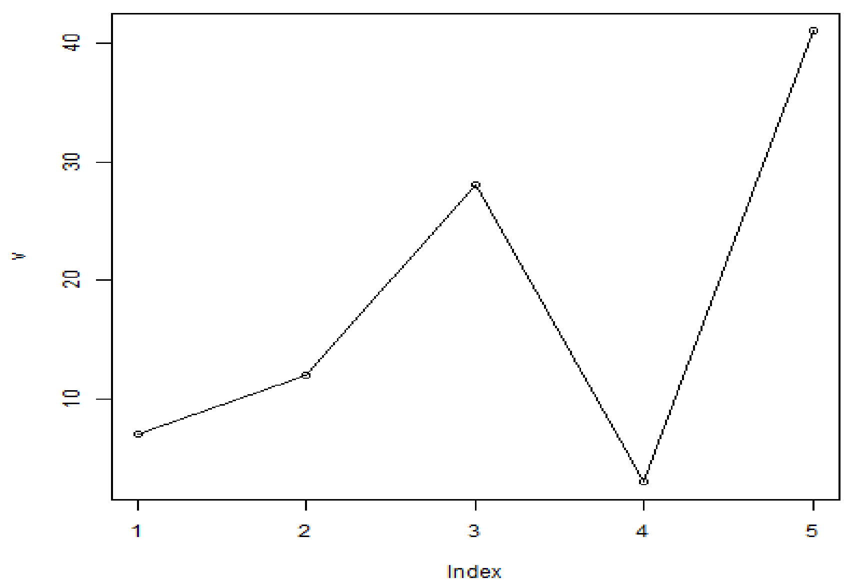
- **v** is a vector containing the numeric values.
- **type** takes the value "p" to draw only the points, "l" to draw only the lines and "o" to draw both points and lines.
- **xlab** is the label for x axis.
- **ylab** is the label for y axis.
- **main** is the Title of the chart.
- **col** is used to give colors to both the points and lines.

- Example

A simple line chart is created using the input vector and the type parameter as "O". The below script will create and save a line chart in the current R working directory.

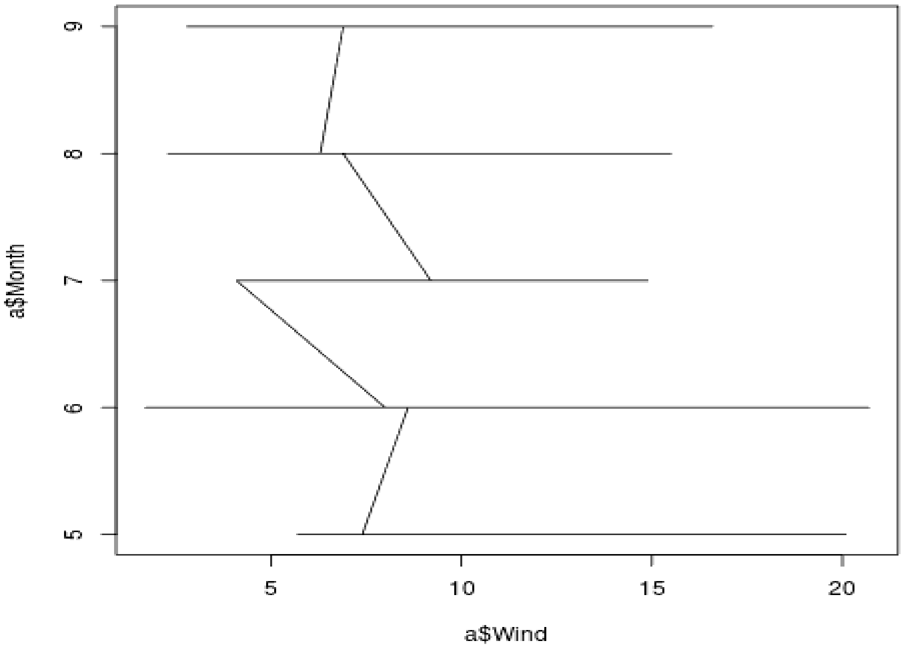
```
# Create the data for the chart.
v <-c(7,12,28,3,41)

# Give the chart file a name.
png(file ="line_chart.jpg")

# Plot the bar chart.
plot(v,type="o")

# Save the file.
dev.off()
```

When we execute the above code, it produces the following result −



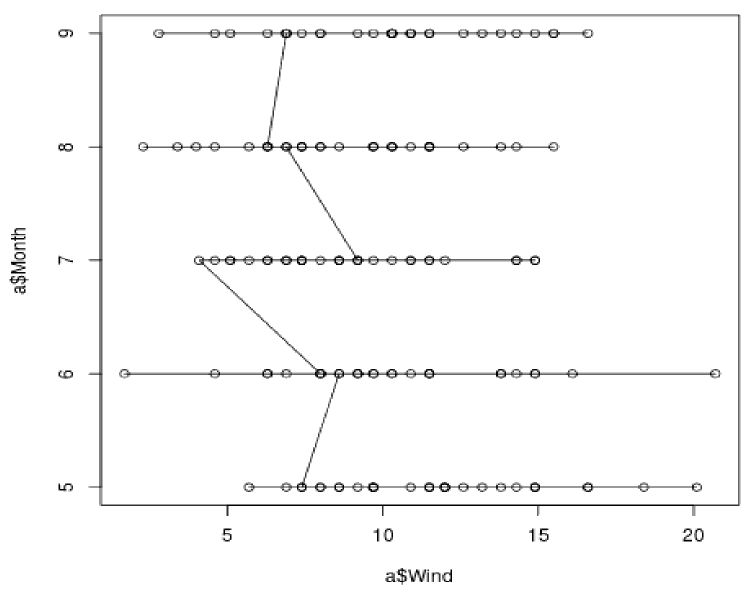
- R- Line Graph using data set

```
> a=read.csv("airquality.csv")
> png(file = "11.png")
> plot(a$Wind,a$Month,type='l')
> dev.off()
```



- Line graph with O option using data set:

```
> png(file = "11.png")
> plot(a$Wind,a$Month,type='o')
> dev.off()
```



2. R - Bar Charts:

A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot()** to create bar charts. R can draw both vertical and horizontal bars in the bar chart. In bar chart each of the bars can be given different colors.

Syntax: The basic syntax to create a bar-chart in R is −

barplot(H, xlab, ylab, main, names.arg, col)

Following is the description of the parameters used −

- **H** is a vector or matrix containing numeric values used in bar chart.
- **xlab** is the label for x axis.
- **ylab** is the label for y axis.
- **main** is the title of the bar chart.
- **names.arg** is a vector of names appearing under each bar.
- **col** is used to give colors to the bars in the graph.

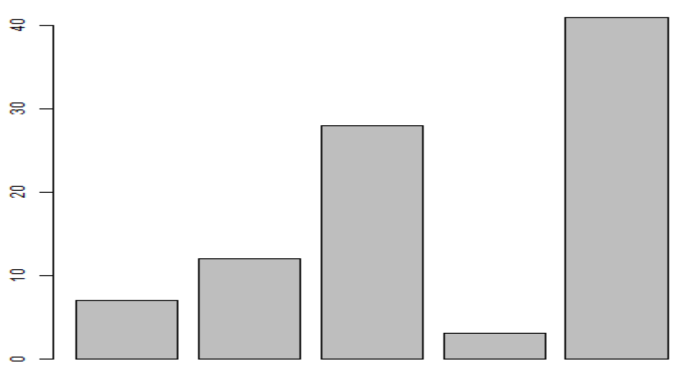
Example

A simple bar chart is created using just the input vector and the name of each bar.

The below script will create and save the bar chart in the current R working directory.

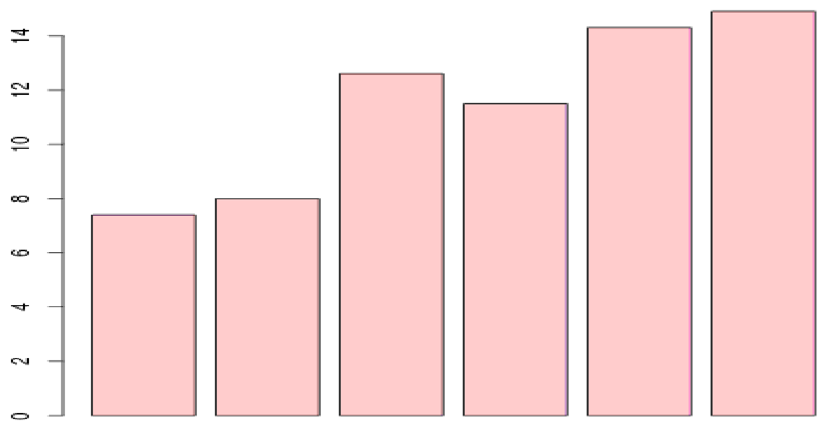
```
# Create the data for the chart.
H <-c(7,12,28,3,41)

# Give the chart file a name.
png(file ="barchart.png")

# Plot the bar chart.
barplot(H)

# Save the file.
dev.off()
```

When we execute the above code, it produces the following result −

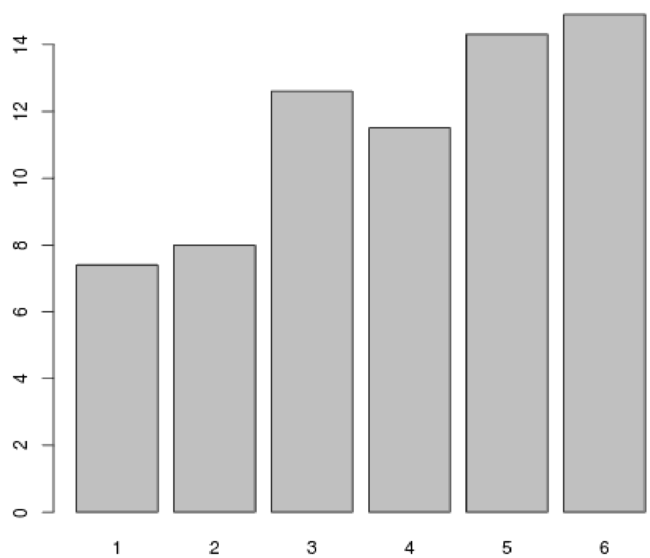


- Bar charts – (with color) using dataset:

```
> a1=head(a)
> png(file = "11.png")
> barplot(a1$Wind,a1$Month,col="pink")
> dev.off()
```




```
> png(file = "11.png")
> barplot(a1$Wind,a1$Month,names.arg=c(1,2,3,4,5,6))
> dev.off()
```



3. R – Boxplots

Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

Boxplots are created in R by using the **boxplot()** function.

- Syntax

The basic syntax to create a boxplot in R is −

```
boxplot(x, data, notch, varwidth, names, main)
```

Following is the description of the parameters used −

- **x** is a vector or a formula.
- **data** is the data frame.
- **notch** is a logical value. Set as TRUE to draw a notch.
- **varwidth** is a logical value. Set as true to draw width of the box proportionate to the sample size.
- **names** are the group labels which will be printed under each boxplot.
- **main** is used to give a title to the graph.

- Example

We use the data set "mtcars" available in the R environment to create a basic boxplot. Let's look at the columns "mpg" and "cyl" in mtcars.

```
input<-mtcars[,c('mpg','cyl')]
print(head(input))
```

When we execute above code, it produces following result −

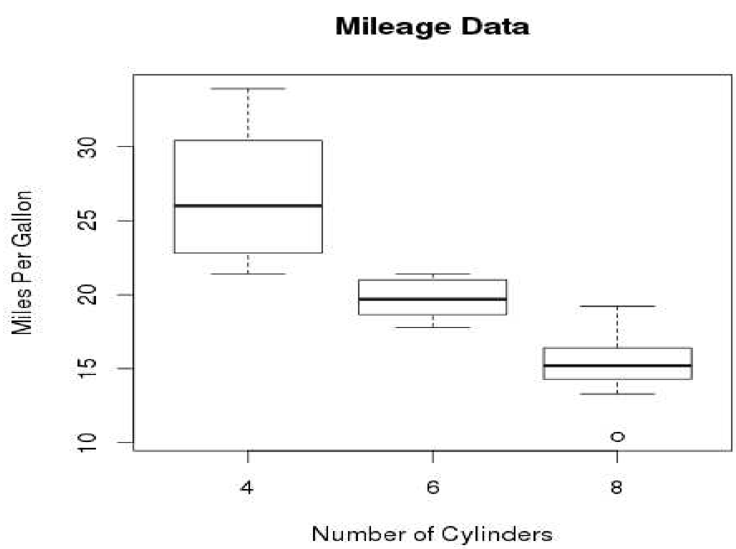
```
mpg  cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
```

- Creating the Boxplot

The below script will create a boxplot graph for the relation between mpg (miles per gallon) and cyl (number of cylinders).

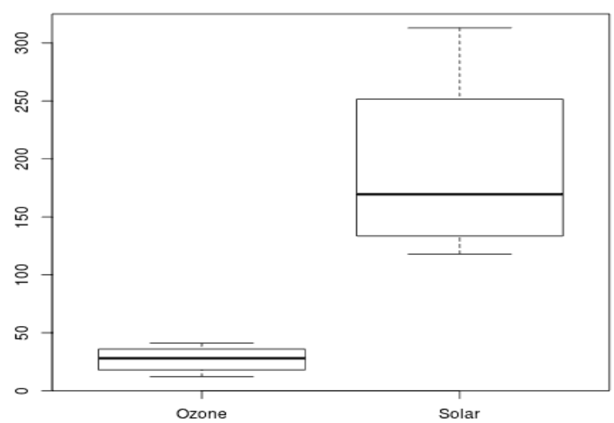
```
# Give the chart file a name.
png(file ="boxplot.png")

# Plot the chart.
boxplot(mpg ~cyl, data =mtcars,xlab="Number of Cylinders",
ylab="Miles Per Gallon", main ="Mileage Data")

# Save the file.
dev.off()
```

When we execute the above code, it produces the following result −



- Boxplot using dataset:

```
> png(file = "11.png")
> d=data.frame(Ozone= a1$Ozone,Solar=a1$Solar.R)
> boxplot(d)
> dev.off()
```



4. R - Histograms

A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.

R creates histogram using **hist()** function. This function takes a vector as an input and uses some more parameters to plot histograms.

-Syntax

The basic syntax for creating a histogram using R is −

```
hist(v,main,xlab,xlim,ylim,breaks,col,border)
```

Following is the description of the parameters used −

- **v** is a vector containing numeric values used in histogram.
- **main** indicates title of the chart.
- **col** is used to set color of the bars.
- **border** is used to set border color of each bar.
- **xlab** is used to give description of x-axis.
- **xlim** is used to specify the range of values on the x-axis.
- **ylim** is used to specify the range of values on the y-axis.
- **breaks** is used to mention the width of each bar.

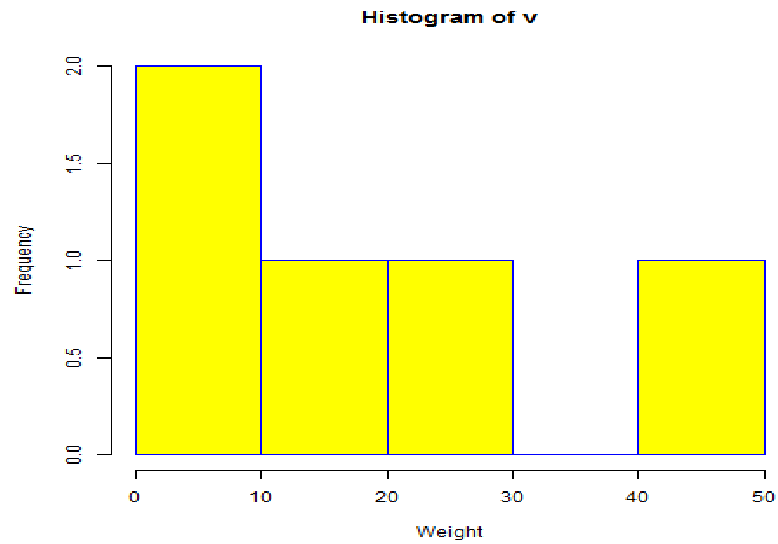
-Example

A simple histogram is created using input vector, label, col and border parameters.

The script given below will create and save the histogram in the current R working directory.

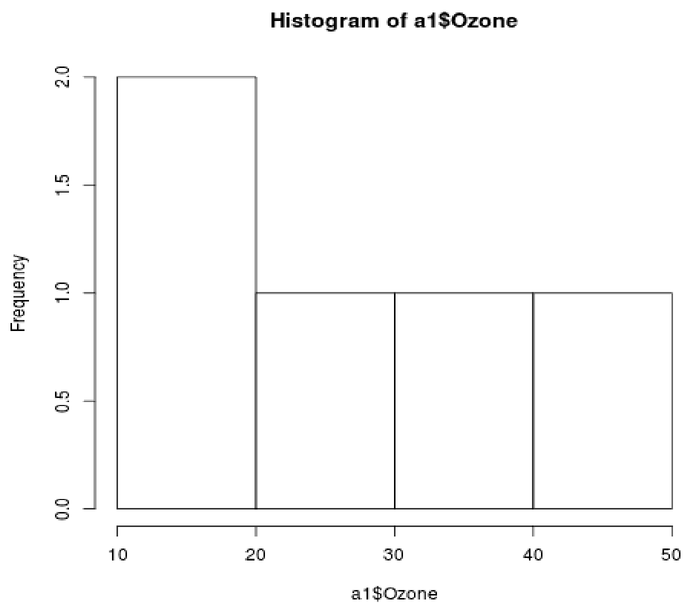
```
# Create data for the graph.
v <-  c(9,13,21,8,36,22,12,41,31,33,19)

# Give the chart file a name.
png(file ="histogram.png")

# Create the histogram.
hist(v,xlab="Weight",col="yellow",border="blue")

# Save the file.
dev.off()
```

When we execute the above code, it produces the following result −



- Histogram with dataset

```
> hist(a1$Ozone)
> png(file = "11.png")
> hist(a1$Ozone)
> dev.off()
```



5. R – Scatterplots

Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.

The simple scatterplot is created using the **plot()** function.

-Syntax

The basic syntax for creating scatterplot in R is −

plot(x, y, main, xlab, ylab, xlim, ylim, axes)

Following is the description of the parameters used −

- **x** is the data set whose values are the horizontal coordinates.
- **y** is the data set whose values are the vertical coordinates.
- **main** is the tile of the graph.
- **xlab** is the label in the horizontal axis.
- **ylab** is the label in the vertical axis.
- **xlim** is the limits of the values of x used for plotting.
- **ylim** is the limits of the values of y used for plotting.
- **axes** indicates whether both axes should be drawn on the plot.

-Example

We use the data set **"mtcars"** available in the R environment to create a basic scatterplot. Let's use the columns "wt" and "mpg" in mtcars.

```
input<-mtcars[,c('wt','mpg')]
print(head(input))
```

When we execute the above code, it produces the following result −

```
wt      mpg
Mazda RX4           2.620   21.0
Mazda RX4 Wag       2.875   21.0
Datsun 710          2.320   22.8
Hornet 4 Drive      3.215   21.4
Hornet Sportabout   3.440   18.7
Valiant             3.460   18.1
```

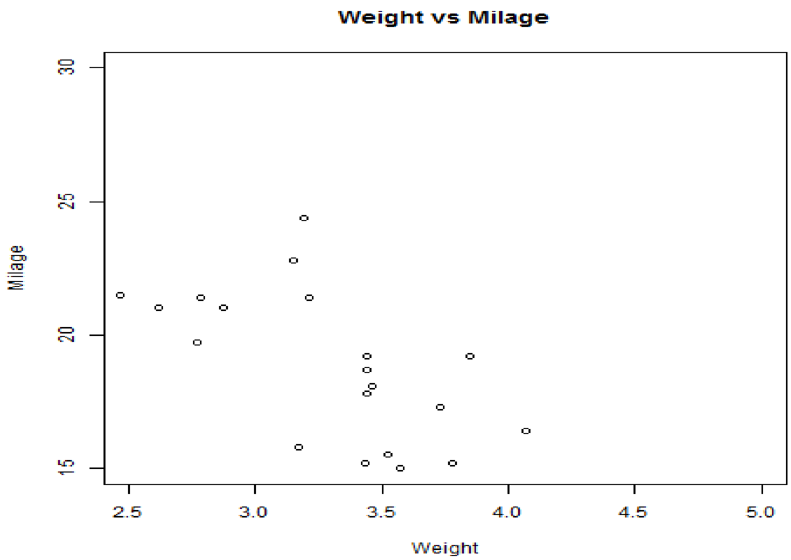
-Creating the Scatterplot

The below script will create a scatterplot graph for the relation between wt(weight) and mpg(miles per gallon).

```
# Get the input values.
input<-mtcars[,c('wt','mpg')]

# Give the chart file a name.
png(file ="scatterplot.png")

# Plot the chart for cars with weight between 2.5 to 5 and mileage between 15 and 30.
plot(x =input$wt,y=input$mpg,
xlab="Weight",
```

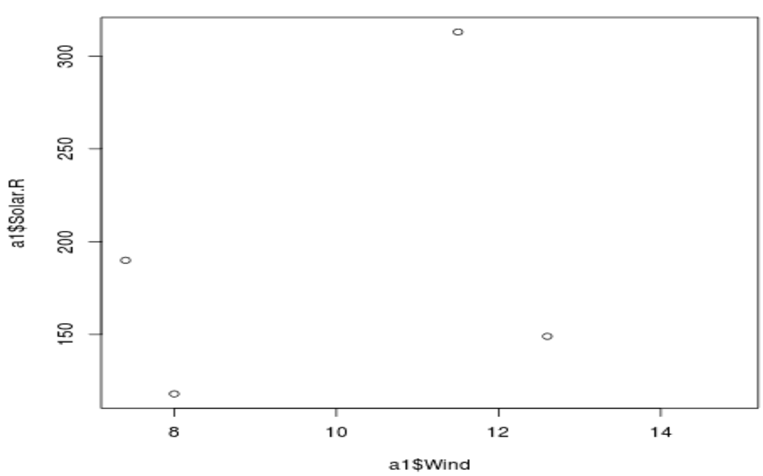
```
ylab="Milage",
xlim= c(2.5,5),
ylim= c(15,30),
main="Weight vs Milage"
)

# Save the file.
dev.off()
```

When we execute the above code, it produces the following result −



- Scatterplots with dataset

```
> png(file = "11.png")
> plot(a1$Wind,a1$Solar.R)
> dev.off()
```



6. Scatterplot Matrices

When we have more than two variables and we want to find the correlation between one variable versus the remaining ones we use scatterplot matrix. We use **pairs()** function to create matrices of scatterplots.

-Syntax

The basic syntax for creating scatterplot matrices in R is −

```
pairs(formula, data)
```

Following is the description of the parameters used −

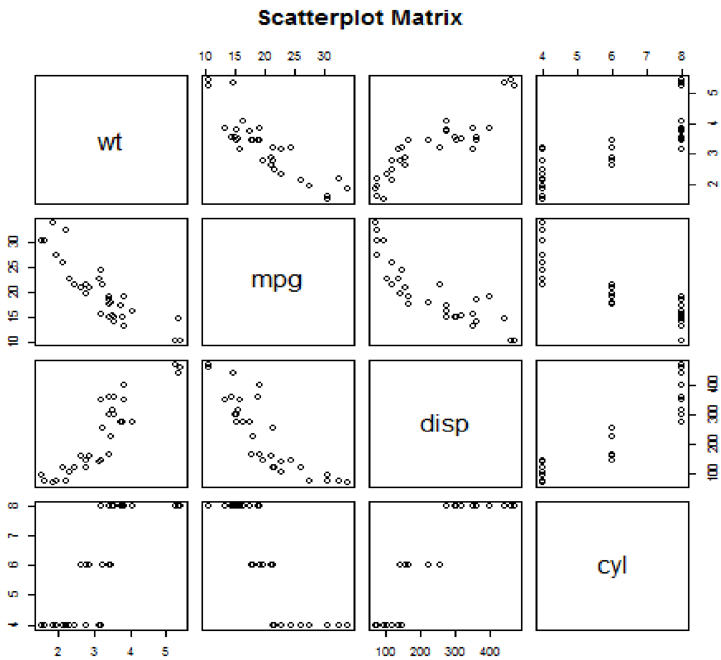
- **formula** represents the series of variables used in pairs.
- **data** represents the data set from which the variables will be taken.

-Example

Each variable is paired up with each of the remaining variable. A scatterplot is plotted for each pair.

```
# Give the chart file a name.
png(file ="scatterplot_matrices.png")

# Plot the matrices between 4 variables giving 12 plots.

# One variable with 3 others and total 4 variables.

pairs(~wt+mpg+disp+cyl,data=mtcars,
main="Scatterplot Matrix")

# Save the file.
dev.off()
```

When the above code is executed we get the following output.



- Scatterplot Matrices with dataset:

When we have more than two variables and we want to find the correlation between one variable versus the remaining ones we use scatterplot matrix. We use pairs() function to create matrices of scatterplots.

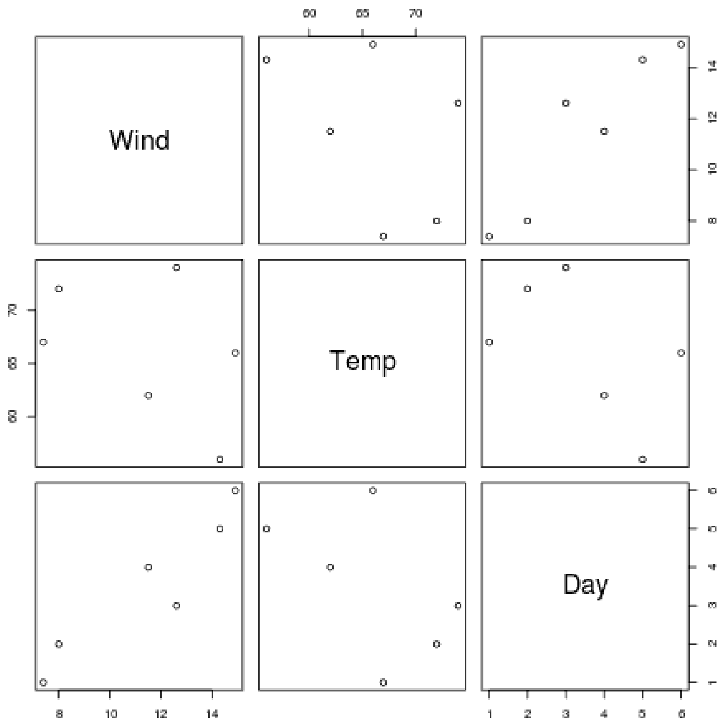
Syntax

The basic syntax for creating scatterplot matrices in R is :
pairs(formula, data)

Following is the description of the parameters used:

· formula represents the series of variables used in pairs.
· data represents the data set from which the variables will be taken.

```
> png(file = "11.png")
> pairs(~Wind+Temp+Day,data = a1)
> dev.off()
```



CONCLUSION: Thus we have learnt Visualize the data using R/Python by plotting the graphs.

ASSIGNMENT TITLE: Perform the following data visualization operations using Tableau on Adult and Iris datasets

OBJECTIVE:

1. To understand and apply the Analytical concept of Big data using Tableau..
2. To study detailed concept of Tableau.

SOFTWARE REQUIREMENTS:

1. Windows 7/8/10
2. GNU C Compiler
3. Tableau

PROBLEM STATEMENT: Perform the following data visualization operations using Tableau on Adult and Iris datasets
1) 1D (Linear) Data visualization
2) 2D (Planar) Data Visualization
3) 3D (Volumetric) Data Visualization
4) Temporal Data Visualization
5) Multidimensional Data Visualization
6) Tree/ Hierarchical Data visualization
7) Network Data visualization

THEORY:

Introduction

Data visualization or data visualization is viewed by many disciplines as a modern equivalent of visual communication. It involves the creation and study of the visual representation of data, meaning "information that has been abstracted in some schematic form, including attributes or variables for the units of information".

Data visualization refers to the techniques used to communicate data or information by encoding it as visual objects (e.g., points, lines or bars) contained in graphics. The goal is to communicate information clearly and efficiently to users. It is one of the steps in data analysis or data science

1) 1D /Linear

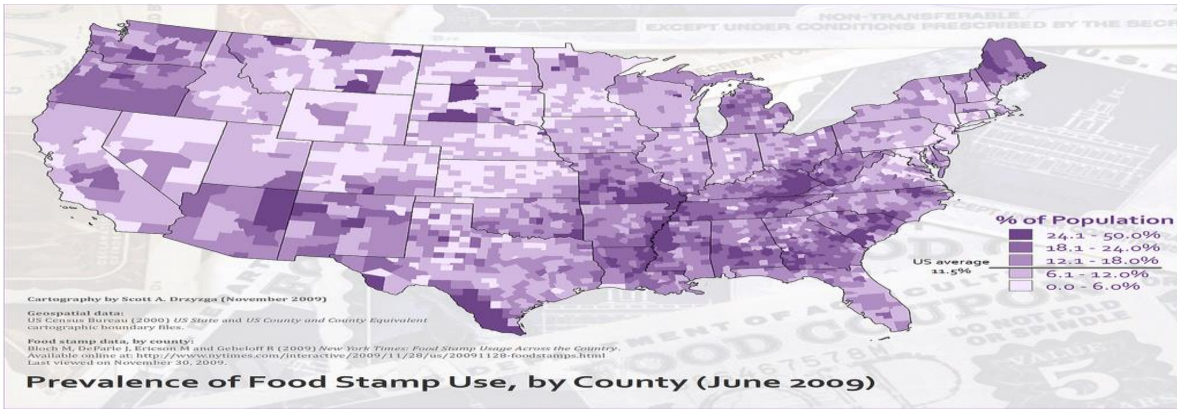
Examples:

- lists of data items, organized by a single feature (e.g., alphabetical order)
  (not commonly visualized)

2) 2D/Planner (especially geospatial)

Examples (geospatial):

- **choropleth**



3) 3D/Volumetric

Broadly, examples of scientific visualization:

- 3D computer models

In 3D computer graphics, **3D modeling** (or **three-dimensional modeling**) is the process of developing a mathematical representation of any *surface* of an object (either inanimate or living) in three dimensions via specialized software. The product is called a **3D model**. Someone who works with 3D models may be referred to as a **3D artist**. It can be displayed as a two-dimensional image through a process called *3D rendering* or used in a computer simulation of physical phenomena. The model can also be physically created using 3D printing devices.

- surface and volume rendering

Rendering is the process of generating an image from a model, by means of computer programs. The model is a description of three-dimensional objects in a strictly defined language or data structure. It would contain geometry, viewpoint, texture, lighting, and shading information. The image is a digital image or raster graphics image. The term may be by analogy with an "artist's rendering" of a scene. 'Rendering' is also used to describe the process of calculating effects in a video editing file to produce final video output.

Volume rendering is a technique used to display a 2D projection of a 3D discretely sampled data set. A typical 3D data set is a group of 2D slice images acquired by a CT or MRI scanner. Usually these are acquired in a regular pattern (e.g., one slice every millimeter) and usually have a regular number of image pixels in a regular pattern. This is an example of a regular volumetric grid, with each volume element, or voxel represented by a single value that is obtained by sampling the immediate area surrounding the voxel.

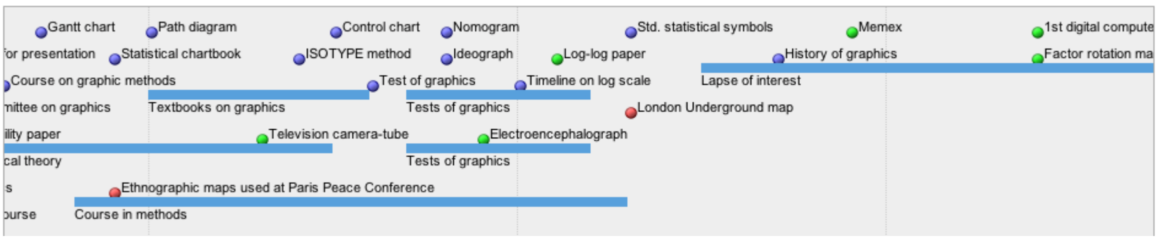
- computer simulations

Computer simulation is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system. Computer simulations have become a useful part of mathematical modeling of many natural systems in physics, and computational physics, chemistry and biology; human systems in economics, psychology, and social science; and in the process of engineering and new technology, to gain insight into the operation of those systems, or to observe their behavior.[6] The simultaneous visualization and simulation of a system is called visulation.

4) Temporal

Examples:

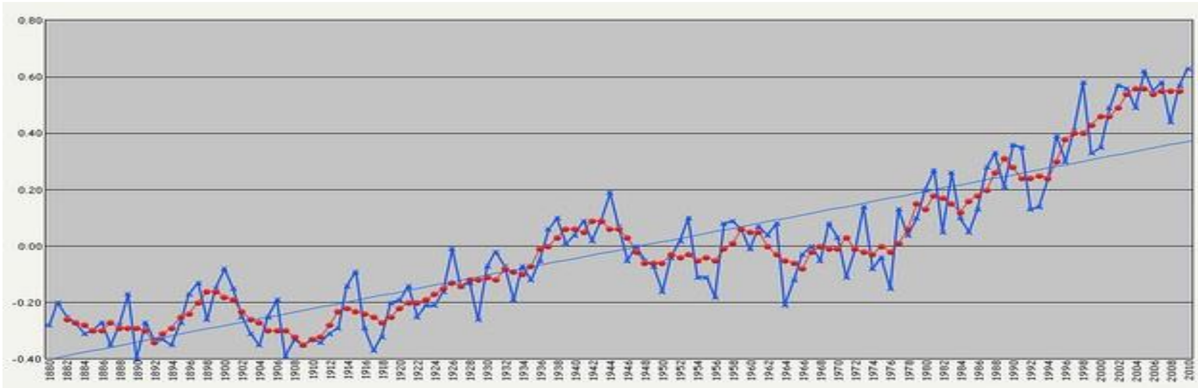
- **timeline**



Tools: SIMILE Timeline, TimeFlow, Timeline JS, Excel

Image:
Friendly, M. & Denis, D. J. (2001). Milestones in the history of thematic cartography, statistical graphics, and data visualization.Web document, http://www.datavis.ca/milestones/.Accessed: August 30, 2012.

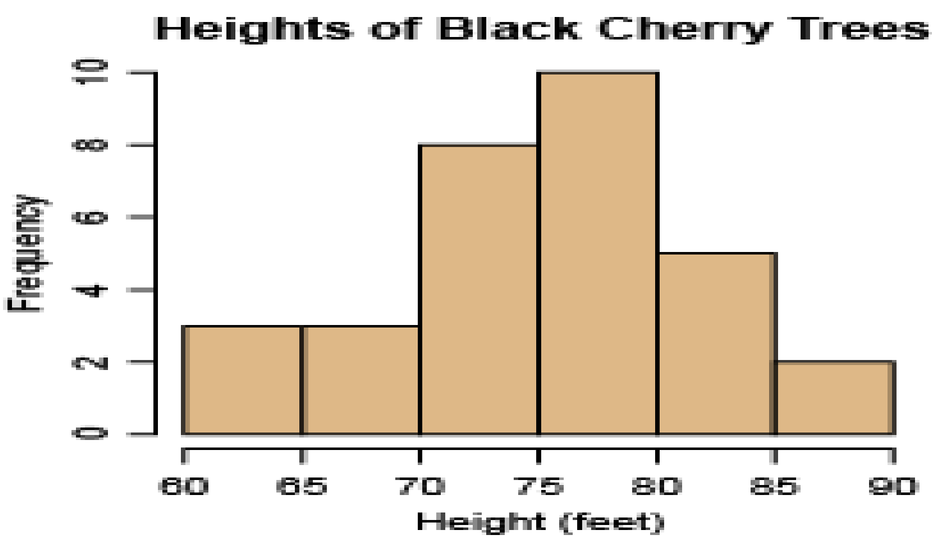
- **time series**



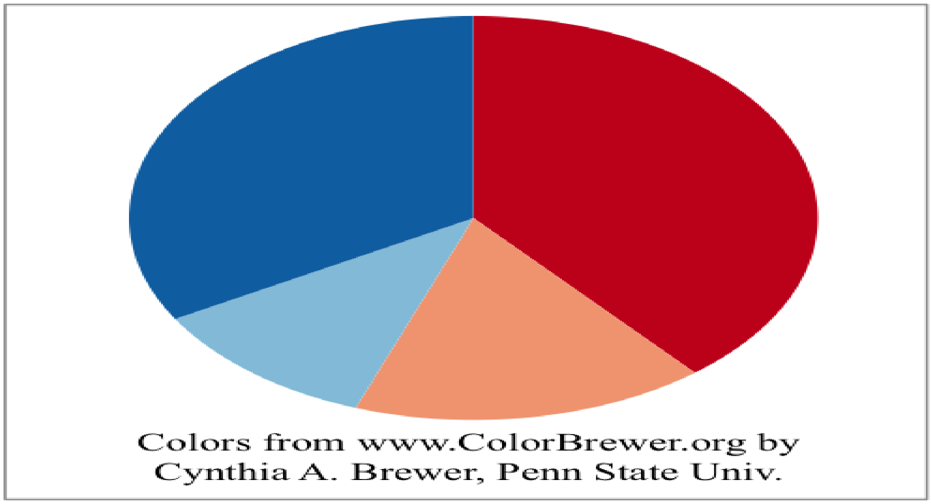
5) nD/Multidimensional

Examples (category proportions, counts):

- **histogram**



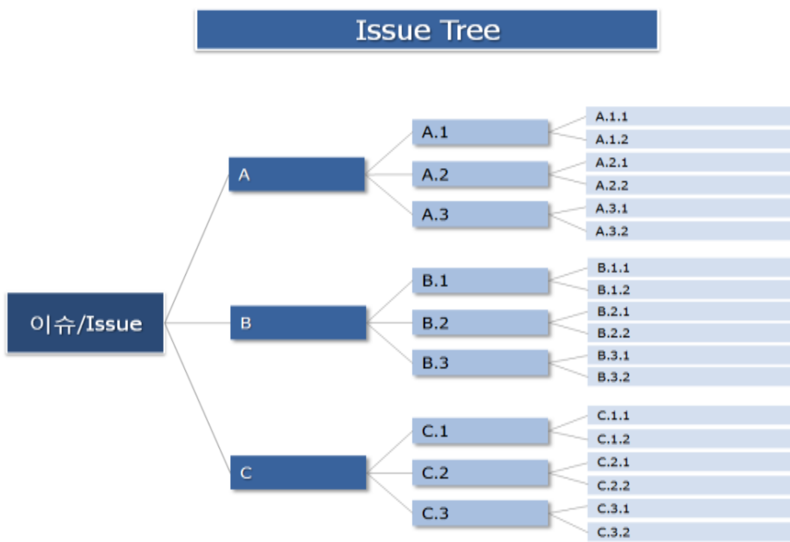
- **pie chart**



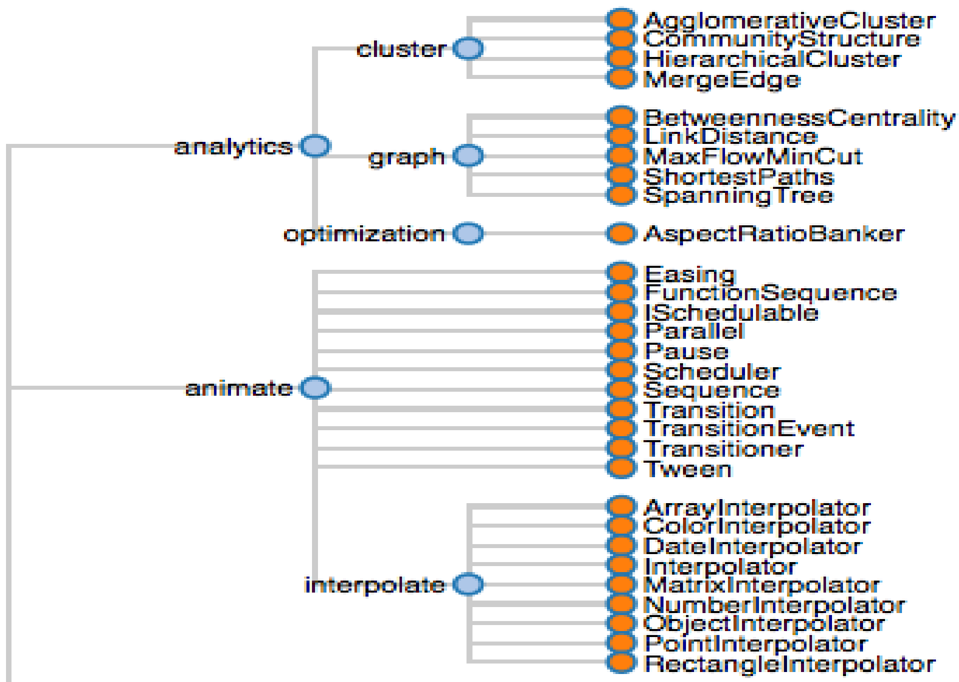
6) Tree/Hirarchical

Example

- **general tree visualization**



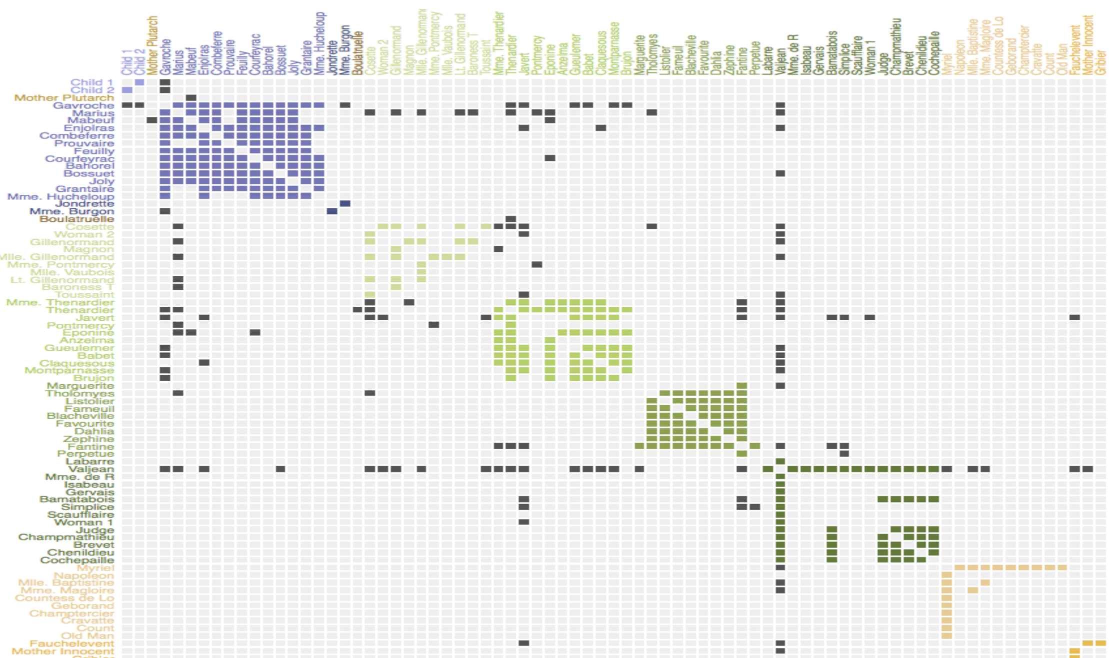
- **dendrogram**



7) Network

Example

- **matrix**



- **node-link diagram (link-based layout algorithm)**

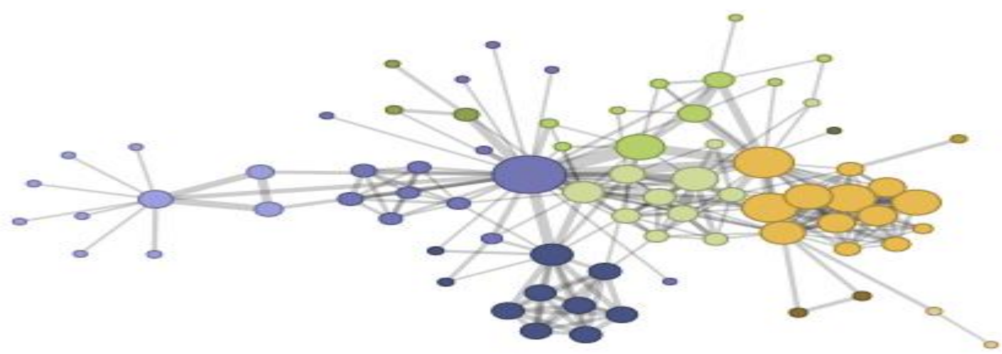


Tableau:

Tableau is a Business Intelligence tool for visually analyzing the data. Users can create and distribute an interactive and shareable dashboard, which depict the trends, variations, and density of the data in the form of graphs and charts. Tableau can connect to files, relational and Big Data sources to acquire and process data. The software allows data blending and real-time collaboration, which makes it very unique. It is used by businesses, academic researchers, and many government organizations for visual data analysis. It is also positioned as a leader Business Intelligence and Analytics Platform in Gartner Magic Quadrant.

Tableau Features:

Tableau provides solutions for all kinds of industries, departments, and data environments. Following are some unique features which enable Tableau to handle diverse scenarios.

- **Speed of Analysis** − As it does not require high level of programming expertise, any user with access to data can start using it to derive value from the data.
- **Self-Reliant** − Tableau does not need a complex software setup. The desktop version which is used by most users is easily installed and contains all the features needed to start and complete data analysis.
- **Visual Discovery** − The user explores and analyzes the data by using visual tools like colors, trend lines, charts, and graphs. There is very little script to be written as nearly everything is done by drag and drop.
- **Blend Diverse Data Sets** − Tableau allows you to blend different relational, semi structured and raw data sources in real time, without expensive up-front integration costs. The users don’t need to know the details of how data is stored.
- **Architecture Agnostic** − Tableau works in all kinds of devices where data flows. Hence, the user need not worry about specific hardware or software requirements to use Tableau.
- **Real-Time Collaboration** − Tableau can filter, sort, and discuss data on the fly and embed a live dashboard in portals like SharePoint site or Salesforce. You can save your view of data and allow colleagues to subscribe to your interactive dashboards so they see the very latest data just by refreshing their web browser.
- **Centralized Data** − Tableau server provides a centralized location to manage all of the organization’s published data sources. You can delete, change permissions, add tags, and manage schedules in one convenient location. It’s easy to schedule extract refreshes and manage them in the data server. Administrators can centrally define a schedule for extracts on the server for both incremental and full refreshes.

There are three basic steps involved in creating any Tableau data analysis report.

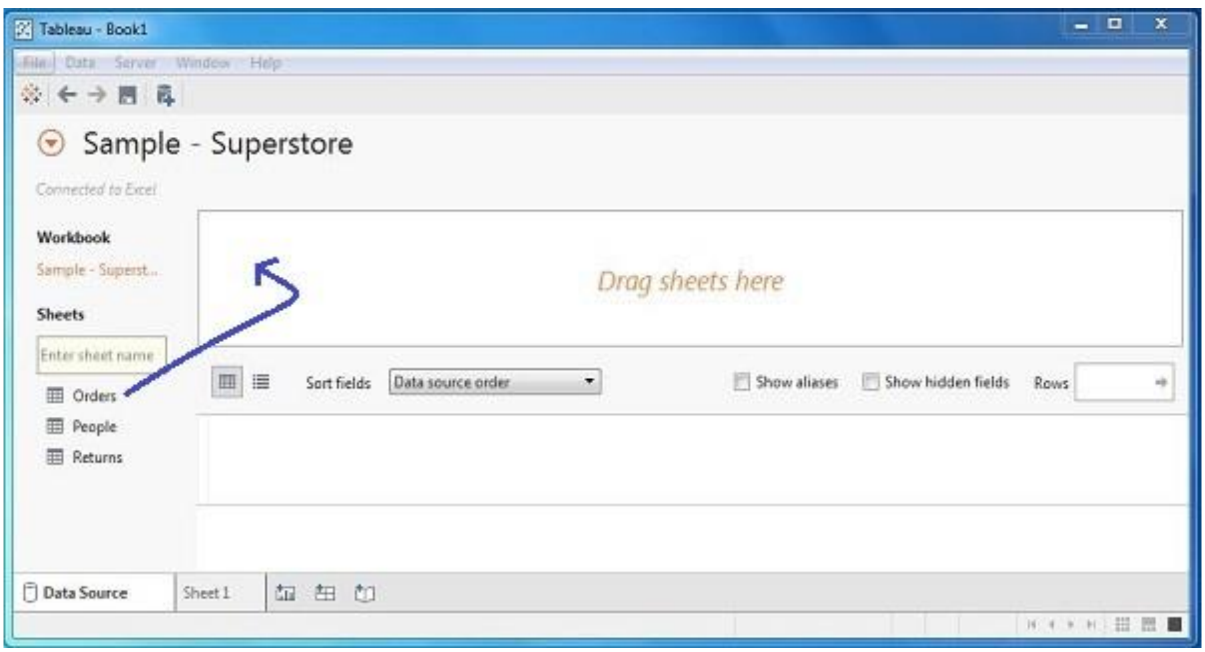
These three steps are −

- **Connect to a data source** − It involves locating the data and using an appropriate type of connection to read the data.
- **Choose dimensions and measures** − This involves selecting the required columns from the source data for analysis.
- **Apply visualization technique** − This involves applying required visualization methods, such as a specific chart or graph type to the data being analyzed.

For convenience, let's use the sample data set that comes with Tableau installation named sample – superstore.xls. Locate the installation folder of Tableau and go to **My Tableau Repository**. Under it, you will find the above file at **Datasources\9.2\en_US-US**.

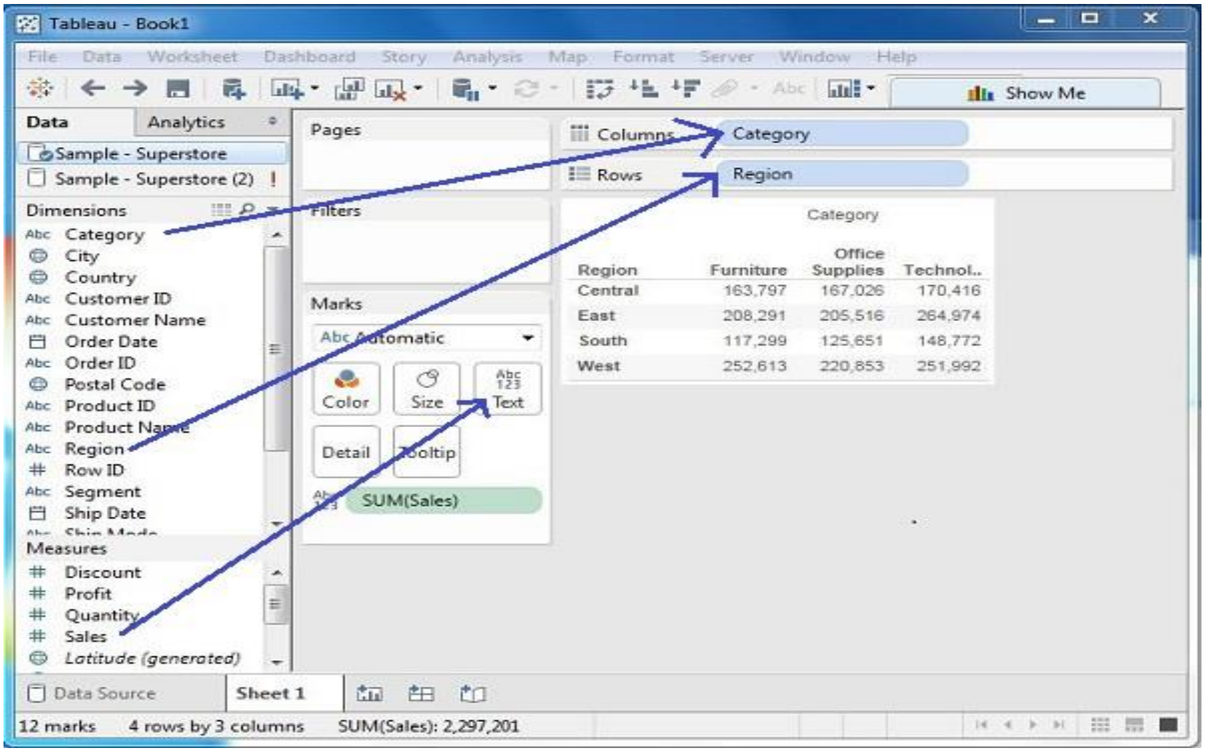
- Connect to a Data Source

On opening Tableau, you will get the start page showing various data sources. Under the header **"Connect"**, you have options to choose a file or server or saved data source. Under Files, choose excel. Then navigate to the file **"Sample – Superstore.xls"** as mentioned above. The excel file has three sheets named Orders, People and Returns. Choose **Orders**.



- Choose the Dimensions and Measures

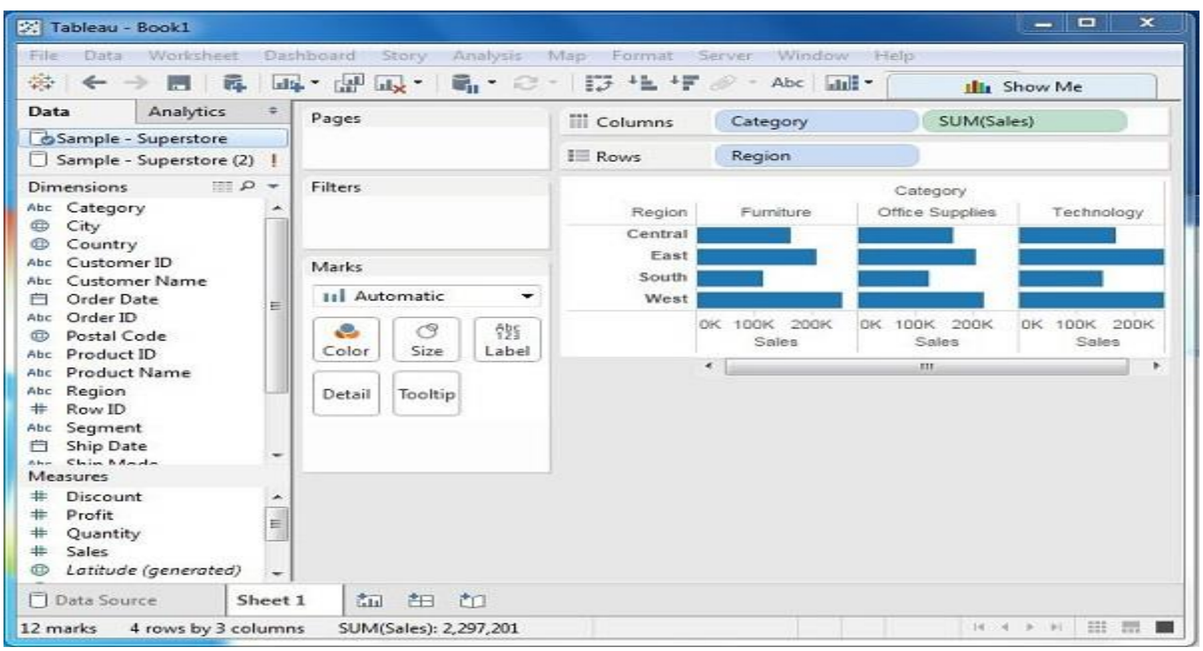
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region** as the dimensions and **Sales** as the measure. Drag and drop them as shown in the following screenshot. The result shows the total sales in each category for each region.



- Apply Visualization Technique

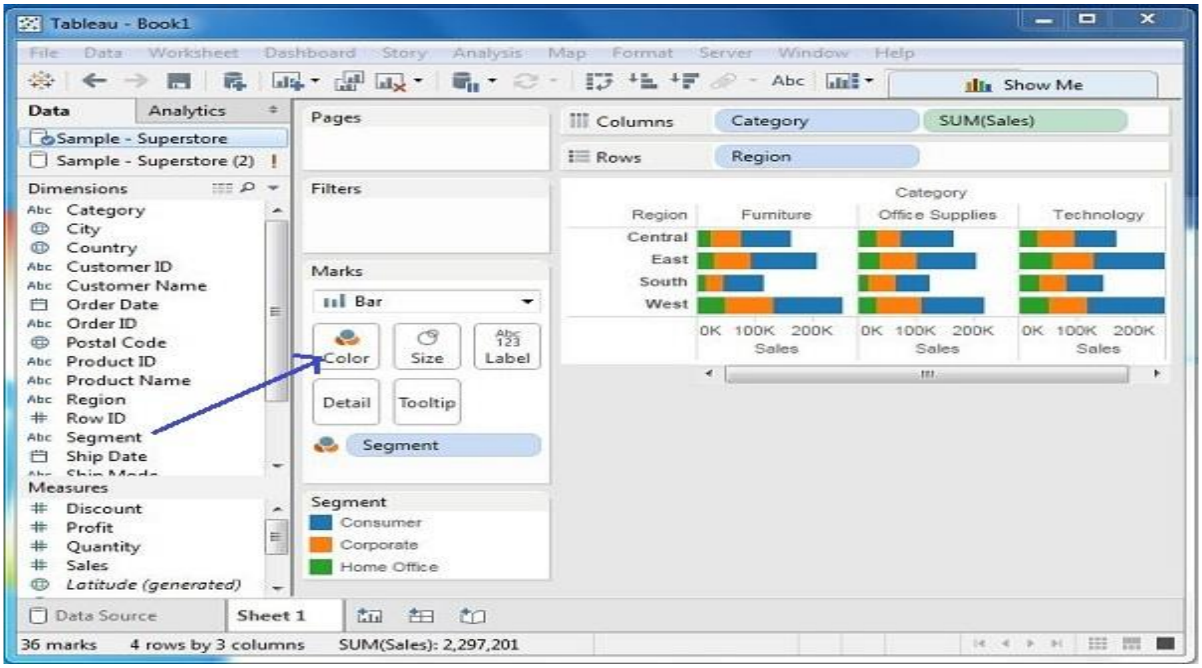
In the previous step, you can see that the data is available only as numbers. You have to read and calculate each of the values to judge the performance. However, you can see them as graphs or charts with different colors to make a quicker judgment.

We drag and drop the sum (sales) column from the Marks tab to the Columns shelf. The table showing the numeric values of sales now turns into a bar chart automatically.



ASSIGNMENT TITLE: **Case Study Assignment**

You can apply a technique of adding another dimension to the existing data. This will add more colors to the existing bar chart as shown in the following screenshot.



CONCLUSION: Thus we have learnt how to Visualize the data in different types (1 1D (Linear) Data visualization,2D (Planar) Data Visualization, 3D (Volumetric) Data Visualization, Temporal Data Visualization, Multidimensional Data Visualization, Tree/ Hierarchical Data visualization, Network Data visualization) by using Tableau Software.

Part C: Case Study Assignment

1) Social Media Analytics
2) Text Mining/ Text Analytics
3) Mobile Analytics

Note: 1. Students need to write about theory concepts on each topic and details about at least one tool such as source URL, Open source/Proprietary, installation steps, configurations, features and compatibility etc. with sample case study.

2. Single Student/Group can give presentation of 10/15 mins on the tool selected & Case study.

1. **Social Media Analytics**

   E.g.

   Tools:

Buffer, Followerwonk, ViralWoot, Google Analytics, Quintly, Cyfe, Tailwind, Keyhole, Klout, Klear, Audiense , TweetReach, IBM Watson Personality Insights, Peakfeed, WolframAlpha Facebook Report, SocialRank, WEBSTA, Talkwalker, LikeAlyzer,

Dashboard:

Facebook Insights, Instagram Insights, Twitter analytics, Pinterest analytics, LinkedIn analytics for individuals, LinkedIn analytics for businesses, Google+ Influence

2. Text Mining/ Text Analytics

Gate, Rapidminer, KH Coder, VisualTest, Datumbox, TAMS, QDA Miner lite, Carrot2, CAT, TM, GENSIM, NLTK, UIMA, Opennlp, KNIME, Apache Mahuot, LPU, Pattern

3. Mobile Analytics

Google Analytics, iOS App Analytics, Flurry Analytics, Amazon Mobile Analytics , Tune, AppsFlyer, App Annie, Adjust, Kochava, Localytics, Tenjin, Appsumer, Mixpanel, Appsee, Amplitude, Branch, Singular, Leanplum, Urban Airship, Appboy, Apptimize, GameAnalytics, Upsight , SWRVE, Taplytics, Apptentive