EXPERIMENT 1

TITLE: Hadoop Installation on Single Node

OBJECTIVE:

1. To Learn and understand 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 Windows 8/10 etc.
2 Java
3 Virtual Machine
4 Cloudera Virtual Machine

THEORY:

Introduction

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across cluster so computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Due to the advent of new technologies, devices, and communication like social networking sites, the amount of data produced by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. The same amount was created in every two days in 2011, and in every ten minutes in2013.This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected

BigData

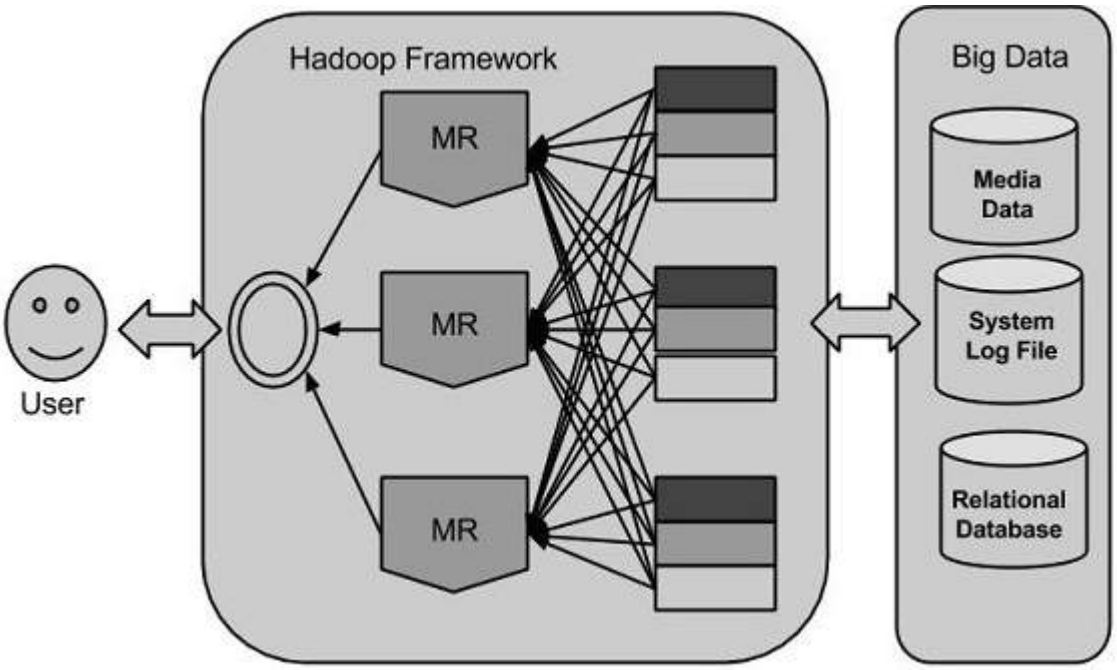
Big data means really a big data, it is a collection of large datasets that cannot be processed using traditional computing techniques. Bigdata is not merely a data, rather it has become a complete subject, which involves various tools, techniques and frameworks. Bigdata involves the data produced by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

Hadoop

Doug Cutting, Mike Cafarella and team took the solution provided by Google and start an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant.

Now Apache Hadoop is a registered trademark of the Apache Software Foundation.

Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes. Inshort, Hadoop framework 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.



Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets 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:

- **Hadoop Common:** 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.
- **HadoopYARN:** This is a framework for job scheduling and cluster resource management.
- **Hadoop Distributed File System(HDFS™):** A distributed file system that provides high-throughput access to application data.

- **Hadoop MapReduce:** This is YARN-based system for parallel processing of large datasets.

Learning Goals

In this activity, you will:

- Download and Install VirtualBox.
- Download and Install Cloudera Virtual Machine (VM) Image.
- Launch the Cloudera VM.

Hardware Requirements: (A) Quad Core Processor (VT-x or AMD-V support recommended), 64-bit;
(B) 8 GB RAM; (C) 20 GB disk free. How to find your hardware information: Open System by clicking the Start button, right-clicking Computer, and then clicking Properties. Most computers with 8 GB RAM purchased in the last 3 years will meet the minimum requirements.You will need a high speed internet connection because you will be downloading files up to 4 Gb in size.

Instructions

Please use the following instructions to download and install the Cloudera Quickstart VM with VirutalBox before proceeding to the Getting Started with the Cloudera VM Environment video. The screenshots are from a Mac but the instructions should be the same for Windows. Please see the discussion boards if you have any issues.

1. Install VirtualBox. Go to https://www.virtualbox.org/wiki/Downloads to download and install VirtualBox for your computer. The course uses Virtualbox 5.1.X, so we recommend clicking VirtualBox 5.1 builds on that page and downloading the older package for ease of following instructions and screenshots. However, it shouldn't be too different if you choose to use or upgrade to VirtualBox 5.2.X. For Windows, select the link "VirtualBox 5.1.X for Windows hosts x86/amd64" where 'X' is the latest version.
2. For Ubuntu Select "VirtualBox 5.1.X for Windows hosts x86/amd64" where 'X' is the latest version.

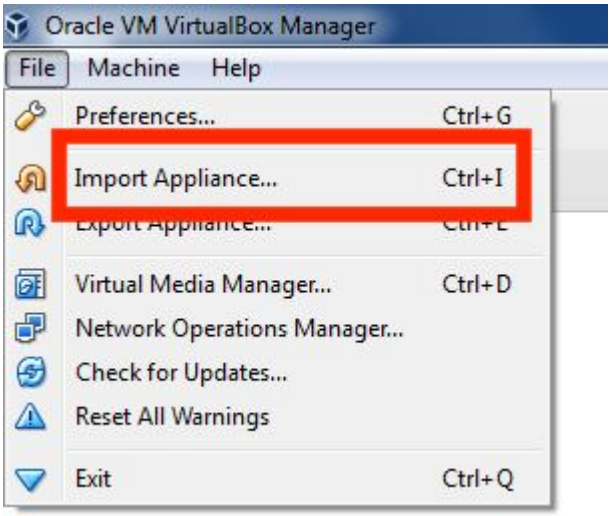
2. **Download the Cloudera VM**. Download the Cloudera VM fromhttps://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.4.2-0- virtualbox.zip. The VM is over 4GB, so will take some time to download.

3. **Unzip the Cloudera VM**:

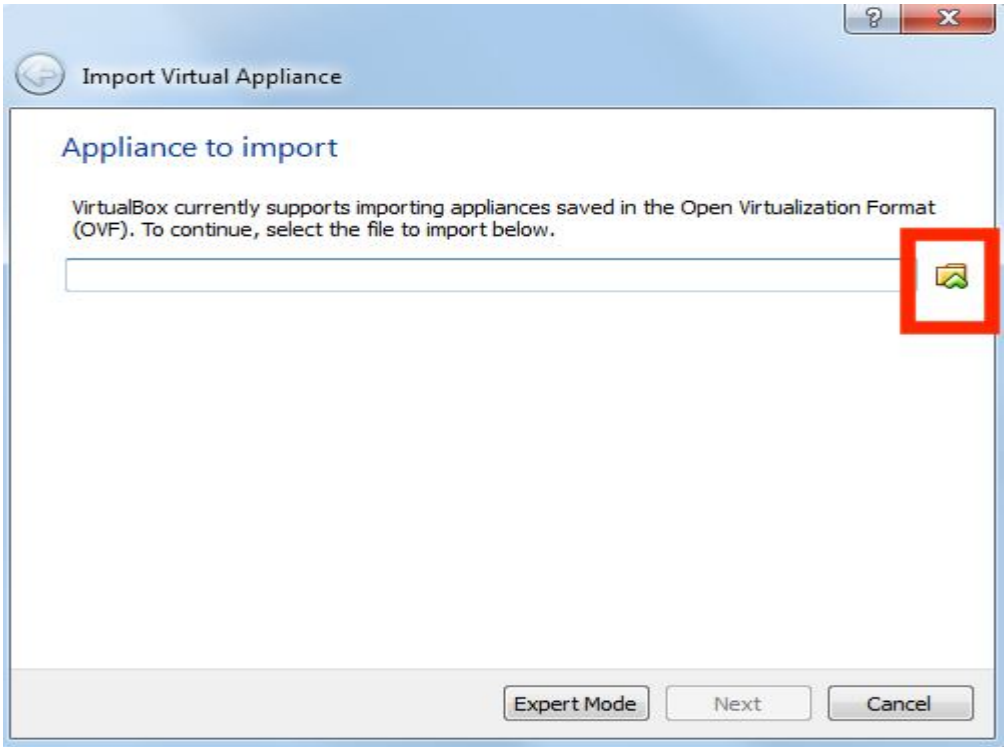
Right-click cloudera-quickstart-vm-5.4.2-0-virtualbox.zip and select “Extract All…”

4. **Start VirtualBox**.

5. **Begin importing**. Import the VM by going to File -> Import Appliance

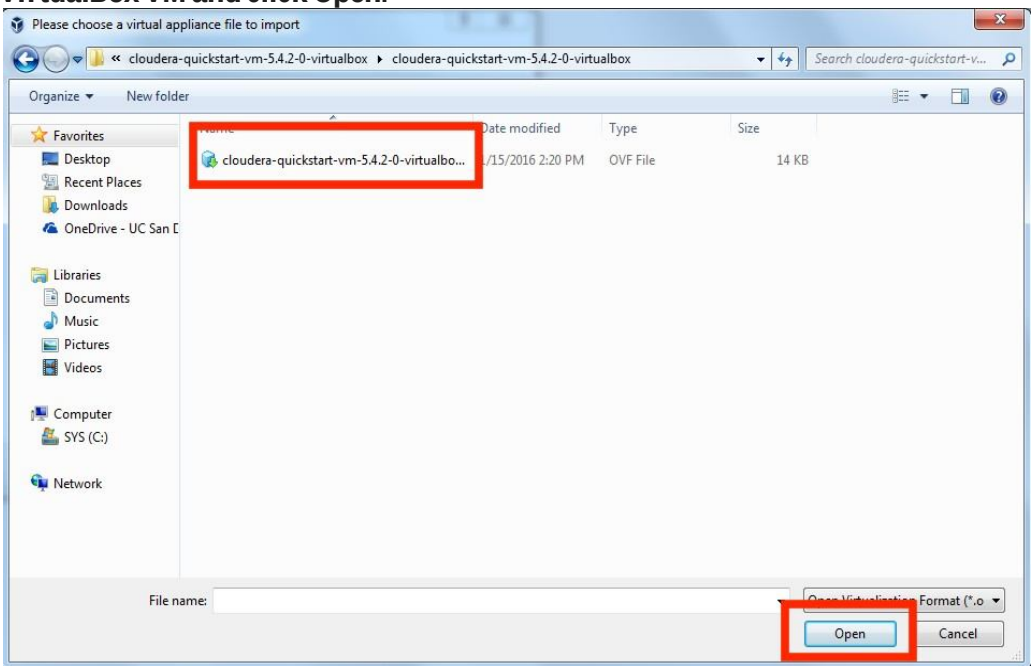


6. Click the Folder icon.

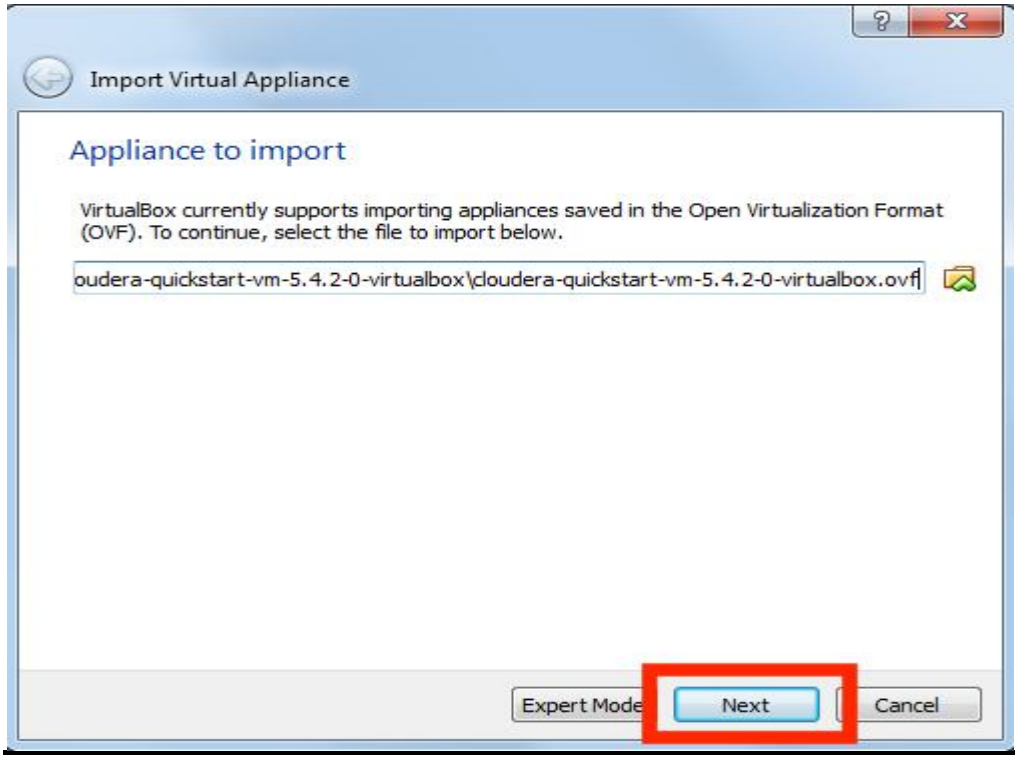


7. Select the cloudera-quickstart-vm-5.4.2-0-virtualbox.ovf from the Folder where you unzippedthe

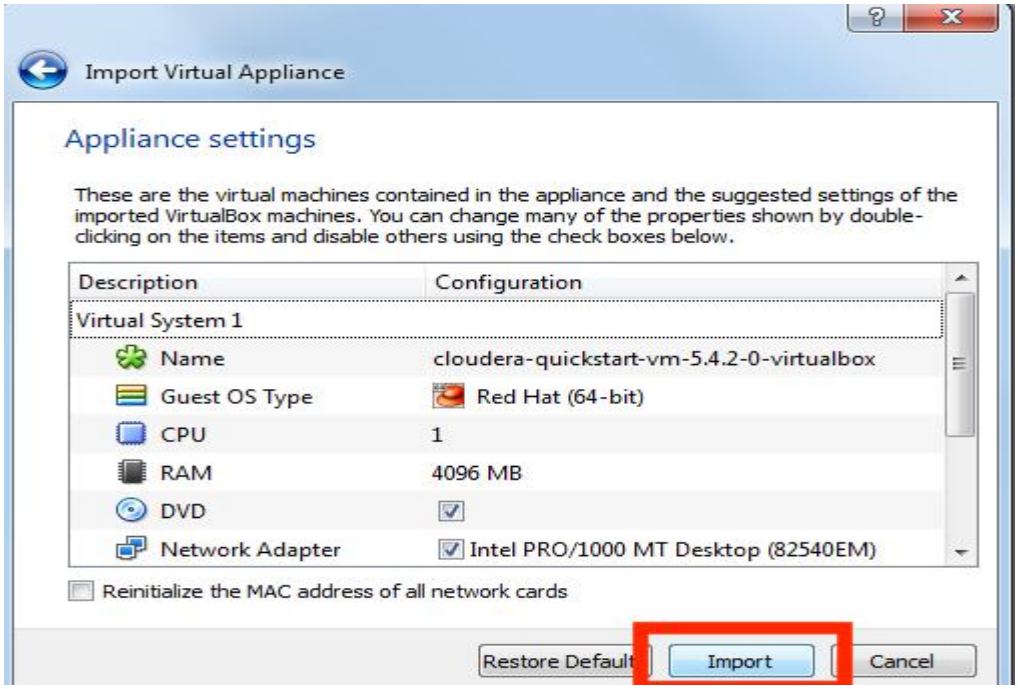
VirtualBox VM and click Open.



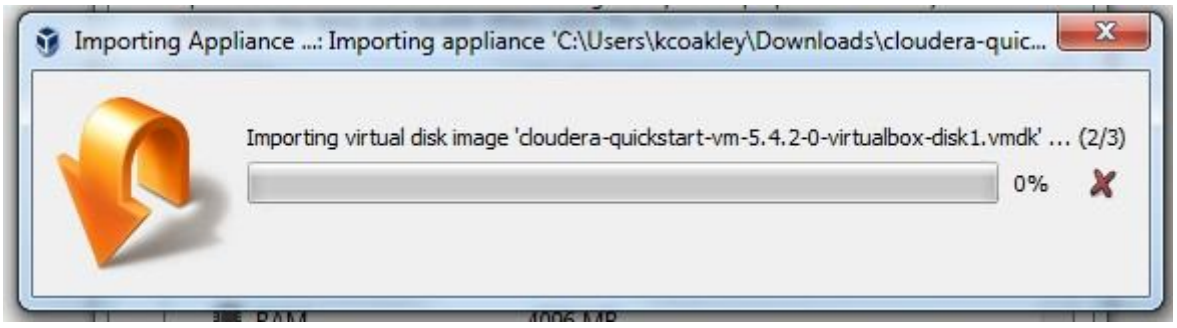
8. **Click Next** to proceed.



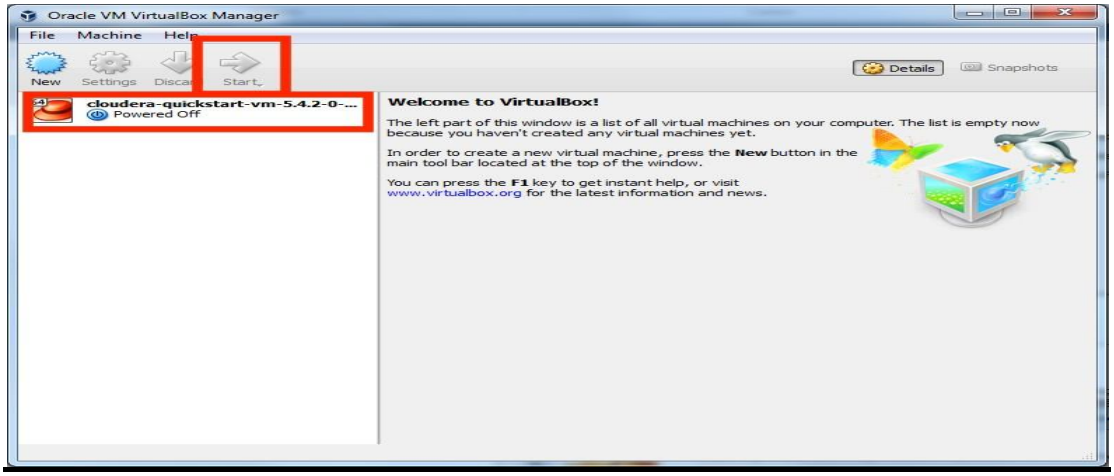
9. Click Import.



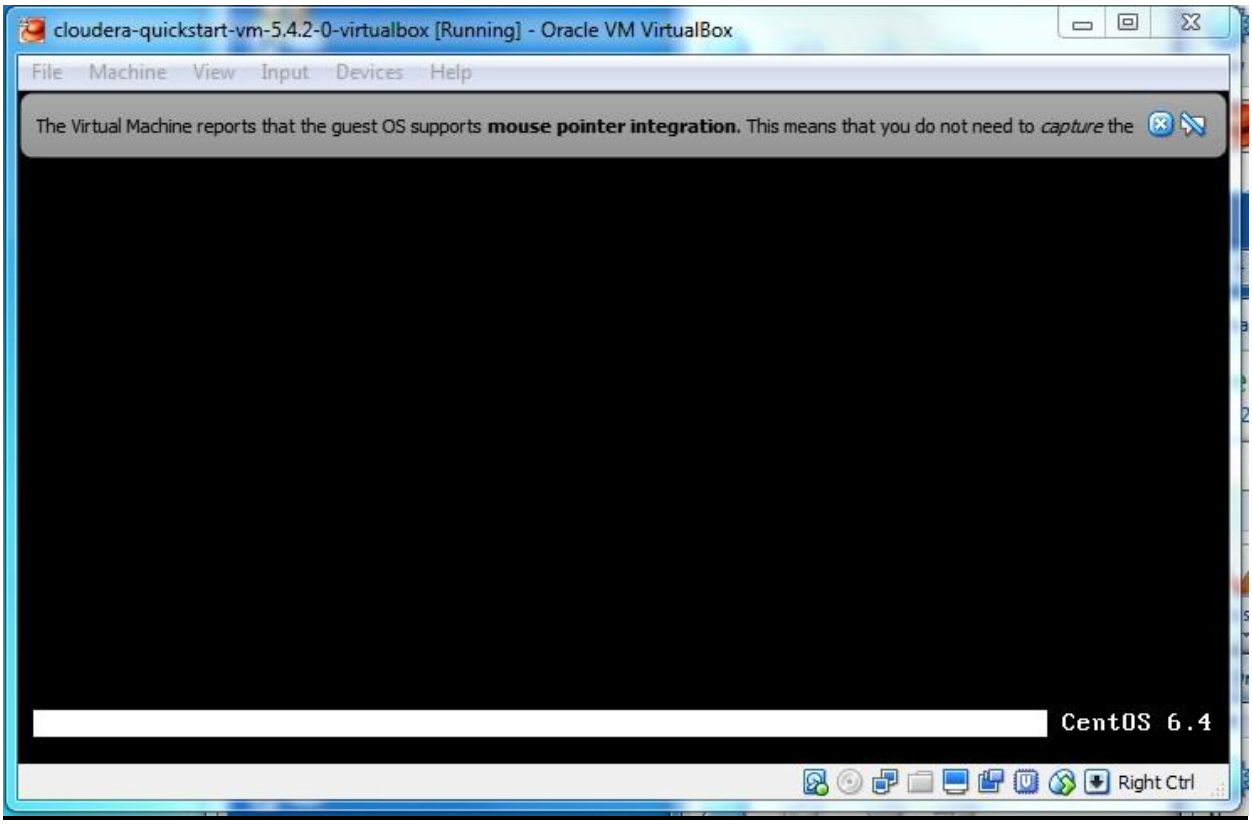
10. **The virtual machine image will be imported.** This can take several minutes.



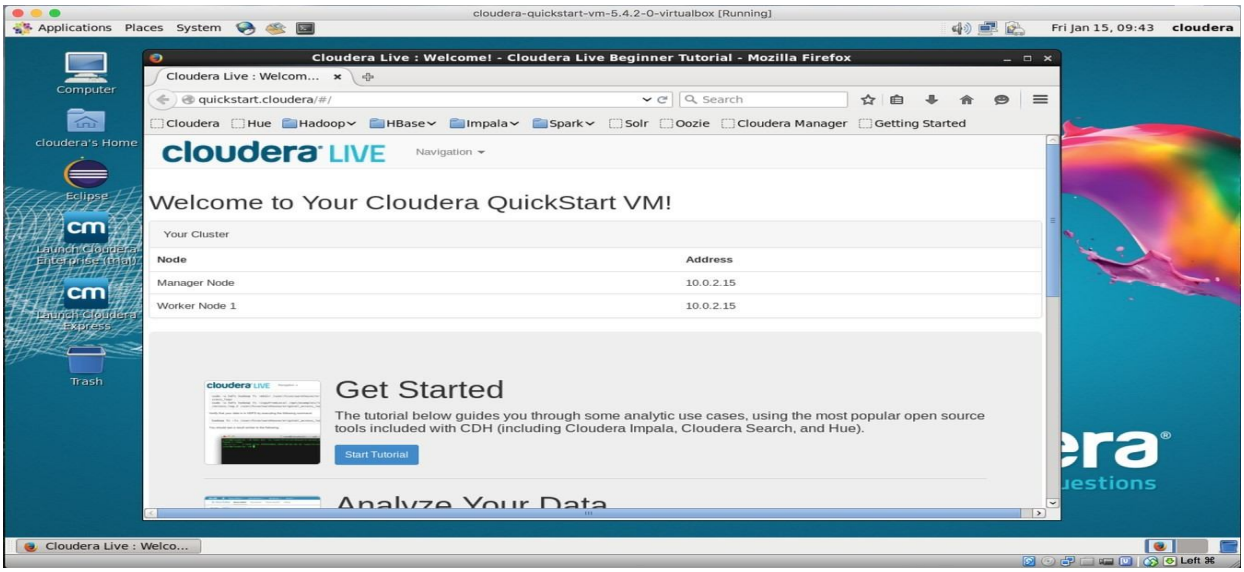
11. **Launch Cloudera VM.** When the importing is finished, the quickstart-vm-5.4.2-0 VM will appearon the left in the VirtualBox window. Select it and click the Start button to launch the VM.



12. Cloudera VM booting. It will take several minutes for the Virtual Machine to start. The bootingprocess takes a long time since many Hadoop tools are started.



12. **The Cloudera VM desktop.** Once the booting process is complete, the desktop will appear witha browser.



CONCLUSION: We studied installation of Hadoop installation and configuration.

EXPERIMENT 2

TITLE: Design a distributed application using MapReduce

OBJECTIVE:

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

SOFTWARE REQUIREMENTS:

1 Windows 8/10 etc.
2 Java
3 Virtual Machine
4 Cloudera Virtual Machine

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:

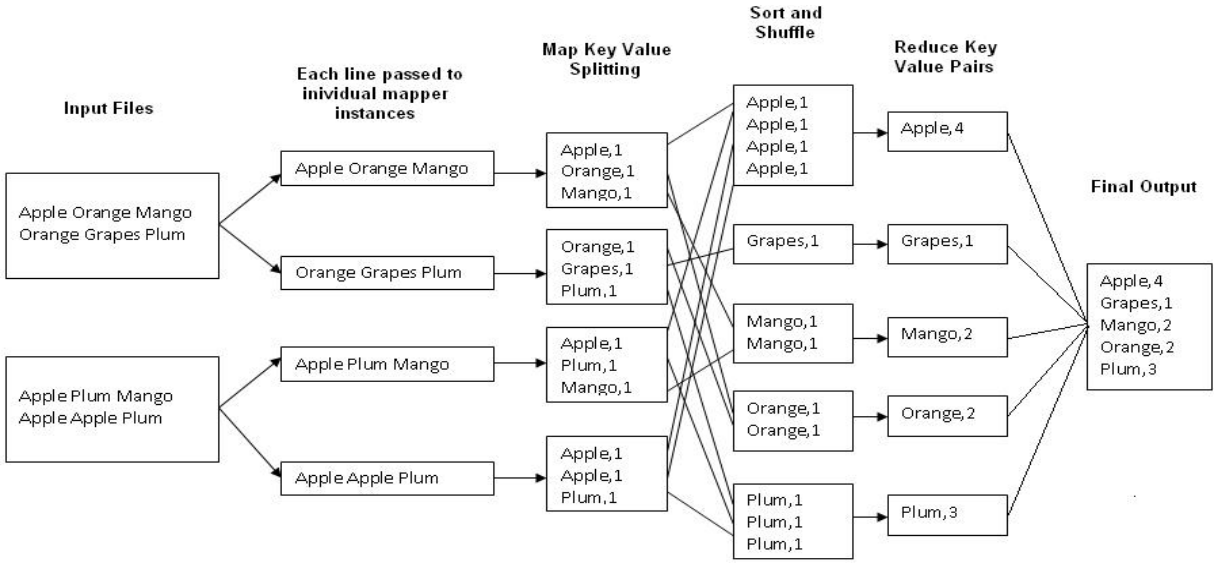


Fig.1 : An Example Program to Understand working of MapReduce Program.

•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.
• The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework.
•Input and Output types of a MapReduce job: (Input) <k1,v1> -> map -><k2, v2>-> reduce -><k3, v3> (Output).

Steps for Compilation & Execution of Program:

- **Open Eclipse IDE**
- **Create Java Project**
- **Create Java Package**
- **Create Java Class file**
- **Write source code**
- **Add some Libraries to java project**
- **Right click on Java Project > Click Build Path > Add External Archives**
  - **Select path usr/lib/hadoop/hadooop_comman.jar**
  - **Select path usr/lib/hadoop-0.20/hadoop-core-2.6.0-mr1-cdh5.4.2.jar**
- **Right click on Java Project >Click Export Project> Java>JAR file**
  - **Browse and select file name of jar file**
  - **Click next**
  - **Click finish**
- **Create input.txt file on desktop**
- **Copy from desktop to present working directory**
  - **$ hadoop fs –put source path destination path**
    for e.g $ hadoop fs –put Desktop/input.txt input.txt
- **Run a jar file**
  - **$hadoop jar JarFileName.jar Packagename.ClassName inputfile.txt OutputDirectory**
    For e.g $ hadoop jar WordCount.jar WordCount.WordCount input.txt dir
- **Check output**
  $ hadoop dfs –cat OutputDirectory/part-r-00000

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

EXPERIMENT 3

TITLE: **Write an application using HiveQL for flight information system**

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. GNU C Compiler
3. Hadoop
4. Java
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:

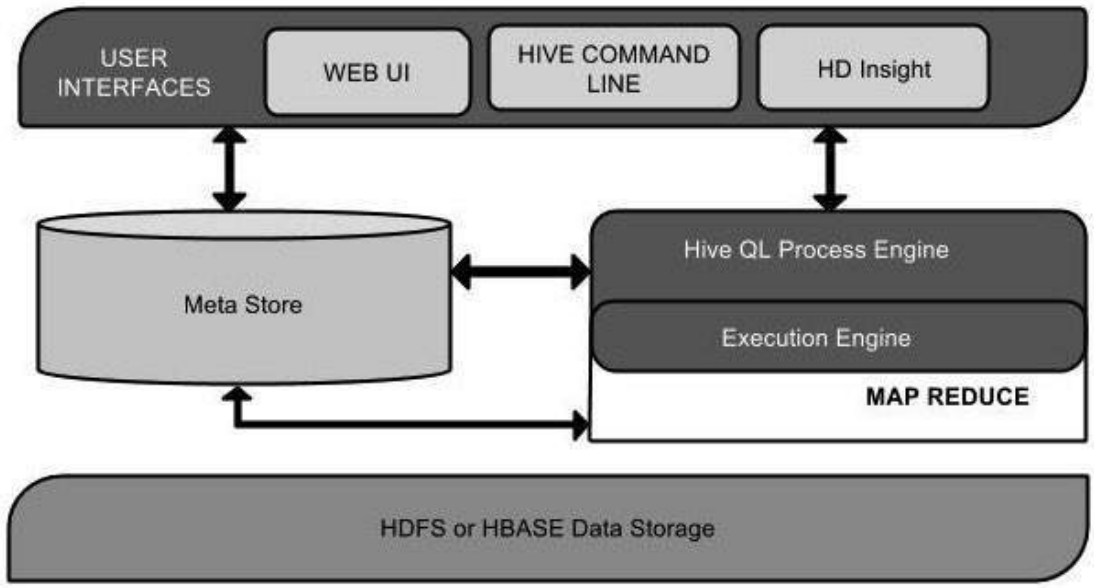


Figure 1: Hive Architecture

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	HiveQL is similar to SQL for querying on schema info on the

Engine	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 1: Hive components and their operations.

a) Creating, Dropping, and altering Database tablesUsing Hbase

#Create Table:

```
hbase(main):002:0> create 'flight','finfo','fsch'0
row(s) in 4.6960 seconds
  => Hbase::Table - flight
  #Table Created-list
  hbase(main):003:0> list
  TABLE
flight
table1
table2
3 row(s) in 0.0120 seconds
  #Insert records in created table
  hbase(main):004:0> put 'flight',1,'finfo:source','pune'
  0 row(s) in 0.2480 seconds
hbase(main):008:0> put 'flight',1,'finfo:dest','mumbai'0
row(s) in 0.0110 seconds
hbase(main):010:0> put 'flight',1,'fsch:at','10.25a.m.'0
row(s) in 0.0060 seconds
```



```
hbase(main):011:0> put 'flight',1,'fsch:dt','11.25 a.m.'0
```

```
row(s) in 0.0070 seconds
hbase(main):012:0> put 'flight',1,'fsch:delay','5min'
hbase(main):015:0> put 'flight',2,'finfo:source','pune'0
row(s) in 0.0160 seconds
hbase(main):016:0> put 'flight',2,'finfo:dest','kolkata'0
row(s) in 0.0070 seconds
hbase(main):017:0> put 'flight',2,'fsch:at','7.00a.m.'0
row(s) in 0.0080 seconds

hbase(main):018:0> put 'flight',2,'fsch:dt','7.30a.m.'0
row(s) in 0.0050 seconds
hbase(main):019:0> put 'flight',2,'fsch:delay','2 min'0
row(s) in 0.0090 seconds
hbase(main):021:0> put 'flight',3,'finfo:source','mumbai'0
row(s) in 0.0040 seconds
hbase(main):022:0> put 'flight',3,'finfo:dest','pune'0
row(s) in 0.0070 seconds
hbase(main):023:0> put 'flight',3,'fsch:at','12.30p.m.'0
row(s) in 0.0100 seconds
hbase(main):024:0> put 'flight',3,'fsch:dt','12.45p.m.'0
row(s) in 0.0040 seconds
hbase(main):025:0> put 'flight',3,'fsch:delay','1 min'0
row(s) in 0.0190 seconds


hbase(main):026:0> put 'flight',4,'finfo:source','mumbai'0
row(s) in 0.0060 seconds
hbase(main):027:0> put 'flight',4,'finfo:dest','delhi'0
row(s) in 0.0050 seconds
hbase(main):028:0> put 'flight',4,'fsch:at','2.00p.m.'0
row(s) in 0.0080 seconds
hbase(main):029:0> put 'flight',4,'fsch:dt','2.45p.m.'0
row(s) in 0.0040 seconds
hbase(main):030:0> put 'flight',4,'fsch:delay','10 min'0
row(s) in 0.0140 seconds
```

#Display Records from Table 'flight'


```
base(main):031:0> scan 'flight' ROW
```

```
                    COLUMN+CELL
 1                  column=finfo:dest, timestamp=1521312730758, value=mumbai
 1                  column=finfo:source, timestamp=1521312493881, value=pune
 1                  column=fsch:at, timestamp=1521312789417, value=10.25a.m.
 1                  column=fsch:delay, timestamp=1521312850594, value=5min

 1                  column=fsch:dt, timestamp=1521312823256, value=11.25 a.m.
 2                  column=finfo:dest, timestamp=1521313135697, value=kolkata
 2                  column=finfo:source, timestamp=1521313092772, value=pune
 2                  column=fsch:at, timestamp=1521313166540, value=7.00a.m.
 2                  column=fsch:delay, timestamp=1521313229963, value=2 min
 2                  column=fsch:dt, timestamp=1521313202767, value=7.30a.m.
 3                  column=finfo:dest, timestamp=1521313310302, value=pune
 3                  column=finfo:source, timestamp=1521313290906, value=mumbai
 3                  column=fsch:at, timestamp=1521313333432, value=12.30p.m.
 3                  column=fsch:delay, timestamp=1521313379725, value=1 min
 3                  column=fsch:dt, timestamp=1521313353804, value=12.45p.m.
 4                  column=finfo:dest, timestamp=1521313419679, value=delhi
 4                  column=finfo:source, timestamp=1521313404831, value=mumbai
 4                  column=fsch:at, timestamp=1521313440328, value=2.00p.m.
 4                  column=fsch:delay, timestamp=1521313472389, value=10 min
 4                  column=fsch:dt, timestamp=1521313455226, value=2.45p.m.
4 row(s) in 0.0300 seconds
```

#Alter Table (add one more column family)

```
hbase(main):036:0> alter 'flight',NAME=>'revenue'
Updating all regions with the new schema...
0/1  regions  updated.
1/1  regions  updated.
Done.
 0 row(s) in 3.7640 seconds

hbase(main):037:0> scan 'flight'
ROW                 COLUMN+CELL
 1                  column=finfo:dest, timestamp=1521312730758, value=mumbai
 1                  column=finfo:source, timestamp=1521312493881, value=pune
 1                  column=fsch:at, timestamp=1521312789417, value=10.25a.m.
 1                  column=fsch:delay, timestamp=1521312850594, value=5min
 1                  column=fsch:dt, timestamp=1521312823256, value=11.25 a.m.
 2                  column=finfo:dest, timestamp=1521313135697, value=kolkata
 2                  column=finfo:source, timestamp=1521313092772, value=pune
 2                  column=fsch:at, timestamp=1521313166540, value=7.00a.m.
 2                  column=fsch:delay, timestamp=1521313229963, value=2 min
 2                  column=fsch:dt, timestamp=1521313202767, value=7.30a.m.
 3                  column=finfo:dest, timestamp=1521313310302, value=pune
```

```
 3          column=finfo:source, timestamp=1521313290906, value=mumbai
 3          column=fsch:at, timestamp=1521313333432, value=12.30p.m.
 3          column=fsch:delay, timestamp=1521313379725, value=1 min
 3          column=fsch:dt, timestamp=1521313353804, value=12.45p.m.
 4          column=finfo:dest, timestamp=1521313419679, value=delhi
 4          column=finfo:source, timestamp=1521313404831, value=mumbai
 4          column=fsch:at, timestamp=1521313440328, value=2.00p.m.
 4          column=fsch:delay, timestamp=1521313472389, value=10 min

 4          column=fsch:dt, timestamp=1521313455226, value=2.45p.m.
4 row(s) in 0.0290 seconds
```

#Insert records into added column family

```
hbase(main):038:0> put 'flight',4,'revenue:rs','45000'
0 row(s) in 0.0100 seconds
```

#Check the updates

```
hbase(main):039:0> scan 'flight'
ROW            COLUMN+CELL
 1          column=finfo:dest, timestamp=1521312730758, value=mumbai
 1          column=finfo:source, timestamp=1521312493881, value=pune
 1          column=fsch:at, timestamp=1521312789417, value=10.25a.m.
 1          column=fsch:delay, timestamp=1521312850594, value=5min
 1          column=fsch:dt, timestamp=1521312823256, value=11.25 a.m.
 2          column=finfo:dest, timestamp=1521313135697, value=kolkata
 2          column=finfo:source, timestamp=1521313092772, value=pune
 2          column=fsch:at, timestamp=1521313166540, value=7.00a.m.
 2          column=fsch:delay, timestamp=1521313229963, value=2 min
 2          column=fsch:dt, timestamp=1521313202767, value=7.30a.m.
 3          column=finfo:dest, timestamp=1521313310302, value=pune
 3          column=finfo:source, timestamp=1521313290906, value=mumbai
 3          column=fsch:at, timestamp=1521313333432, value=12.30p.m.
 3          column=fsch:delay, timestamp=1521313379725, value=1 min
 3          column=fsch:dt, timestamp=1521313353804, value=12.45p.m.
 4          column=finfo:dest, timestamp=1521313419679, value=delhi
 4          column=finfo:source, timestamp=1521313404831, value=mumbai
 4          column=fsch:at, timestamp=1521313440328, value=2.00p.m.
 4          column=fsch:delay, timestamp=1521313472389, value=10 min
 4          column=fsch:dt, timestamp=1521313455226, value=2.45p.m.
 4          column=revenue:rs, timestamp=1521314406914, value=45000
4 row(s) in 0.0340 seconds
```

#Delete Column family

```
hbase(main):040:0>  alter  'flight',NAME=>'revenue',METHOD=>'delete'
Updating all regions with the new schema...
0/1  regions  updated.
1/1  regions  updated.
Done.
```

```
0 row(s) in 3.7880 seconds

#changes Reflected in Table
hbase(main):041:0> scan 'flight'
ROW                COLUMN+CELL
 1                 column=finfo:dest, timestamp=1521312730758, value=mumbai
 1                 column=finfo:source, timestamp=1521312493881, value=pune
 1                 column=fsch:at, timestamp=1521312789417, value=10.25a.m.
 1                 column=fsch:delay, timestamp=1521312850594, value=5min
 1                 column=fsch:dt, timestamp=1521312823256, value=11.25 a.m.
 2                 column=finfo:dest, timestamp=1521313135697, value=kolkata
 2                 column=finfo:source, timestamp=1521313092772, value=pune
 2                 column=fsch:at, timestamp=1521313166540, value=7.00a.m.
 2                 column=fsch:delay, timestamp=1521313229963, value=2 min
 2                 column=fsch:dt, timestamp=1521313202767, value=7.30a.m.
 3                 column=finfo:dest, timestamp=1521313310302, value=pune
 3                 column=finfo:source, timestamp=1521313290906, value=mumbai
 3                 column=fsch:at, timestamp=1521313333432, value=12.30p.m.
 3                 column=fsch:delay, timestamp=1521313379725, value=1 min
 3                 column=fsch:dt, timestamp=1521313353804, value=12.45p.m.
 4                 column=finfo:dest, timestamp=1521313419679, value=delhi
 4                 column=finfo:source, timestamp=1521313404831, value=mumbai
 4                 column=fsch:at, timestamp=1521313440328, value=2.00p.m.
 4                 column=fsch:delay, timestamp=1521313472389, value=10 min
4                  column=fsch:dt, timestamp=1521313455226, value=2.45p.m.
4 row(s) in 0.0280 seconds
```

#Drop Table

```
#Create Table for dropping
hbase(main):046:0* create 'tb1','cf'
0 row(s) in 2.3120 seconds
=> Hbase::Table - tb1

hbase(main):047:0> list
TABLE
flight
table1
table2
tb1
4 row(s) in 0.0070 seconds
=> ["flight", "table1", "table2", "tb1"]
```

#Drop Table

```
hbase(main):048:0> drop 'tb1'

ERROR: Table tb1 is enabled. Disable it first.

Here is some help for this command:
  Drop the named table. Table must first be disabled:
    hbase> drop 't1'
    hbase> drop 'ns1:t1'
```

#Disable table

```
hbase(main):049:0> disable 'tb1'
0 row(s) in 4.3480 seconds

hbase(main):050:0> drop 'tb1'0
row(s) in 2.3540 seconds

hbase(main):051:0> list
TABLE
flight
table1
table2
3 row(s) in 0.0170 seconds

=> ["flight", "table1", "table2"]
```

#Read data from table for row key 1:

```
hbase(main):052:0> get 'flight',1
COLUMN                CELL
 finfo:dest           timestamp=1521312730758, value=mumbai
finfo:source          timestamp=1521312493881, value=pune
fsch:at               timestamp=1521312789417, value=10.25a.m.
fsch:delay            timestamp=1521312850594, value=5min
fsch:dt               timestamp=1521312823256, value=11.25 a.m.
5 row(s) in 0.0450 seconds
```

Read data for particular column from HBase table:

```
hbase(main):053:0> get 'flight','1',COLUMN=>'finfo:source'
COLUMN                CELL
 finfo:source         timestamp=1521312493881, value=pune
1 row(s) in 0.0110 seconds
```

Read data for multiple columns in HBase Table:

```
hbase(main):054:0> get 'flight','1',COLUMN=>['finfo:source','finfo:dest']
COLUMN                CELL
   finfo:dest         timestamp=1521312730758, value=mumbai
   finfo:source  timestamp=1521312493881, value=pune
2 row(s) in 0.0190 seconds

hbase(main):055:0>  scan  'flight',COLUMNS=>'finfo:source'
ROW                   COLUMN+CELL
   1                  column=finfo:source, timestamp=1521312493881, value=pune
   2                  column=finfo:source, timestamp=1521313092772, value=pune
   3                  column=finfo:source, timestamp=1521313290906, value=mumbai
   4                  column=finfo:source, timestamp=1521313404831, value=mumbai
  4 row(s) in 0.0320 seconds
```

b)Creating an external Hive table to connect to the HBase forCustomer Information Table

Covers===>

c) Load table with data, insert new values and field in the table, Join tables with

d) Hive

Add these Jar files in Hive(on hive prompt)

```
add jar file:///usr/local/HBase/lib/zookeeper-3.4.6.jar;
add jar file:///usr/local/HBase/lib/guava-12.0.1.jar;
add jar file:///usr/local/HBase/lib/hbase-client-1.2.6.jar; add
jar file:///usr/local/HBase/lib/hbase-common-1.2.6.jar;add
jar file:///usr/local/HBase/lib/hbase-protocol-1.2.6.jar;add
jar file:///usr/local/HBase/lib/hbase-server-1.2.6.jar; add jar
file:///usr/local/HBase/lib/hbase-shell-1.2.6.jar; add jar
file:///usr/local/HBase/lib/hbase-thrift-1.2.6.jar;
add jar file:///usr/local/hive/lib/hive-hbase-handler-2.2.0.jar;
```

Set the values of variables in Hive

```
set hbase.zookeeprt.quorum=localhost;
set hive.metastore.client.setugi=true;
set hive.exec.stagingdir=/tmp/.hivestage;
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.auto.convert.join=false;
set hive.hbase.wal.enabled=false;
SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;
SET hive.exec.max.dynamic.partitions = 10000;
SET hive.exec.max.dynamic.partitions.pernode = 1000;
```

 # Create the external table emp using hive

```
hive>create external table empdata2 ( ename string, esal int)
  row format delimited fields terminated by "," stored as textfile location
   "/home/hduser/Desktop/empdata2";

hive>load data local inpath '/home/hduser/Desktop/empdb.txt' into table empdata2;
```

#Create External Table in hive referring to hbase table

 # create hbase table emphive first

```
hbase(main):003:0>  create 'emphive', 'cf'
0 row(s) in 4.6260 seconds
```

#create hive external table

```
CREATE external TABLE hive_table_emp(id int, name string, esal string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cf:name,cf:esal")TBLPROPERTIES
("hbase.table.name" = "emphive");
```

 # load data into hive_table_emp

(Hive doesn't allow directly inserting data into external hive table)

#for that create one hive table(managed table in hive)

Managed table and External **table in Hive**. There are two types of **tables in Hive** ,one is **Managed table** and second is external **table**. the difference is , when you drop a **table**, if it is **managed table hive** deletes both data and meta data,if it is external **table Hive** only deletes metadata.

```
hive>create table empdbnew(eno int, ename string, esal int) row format delimited fields terminatedby
',' stored as textfile;
```

#load data in managed table

```
hive>load data local inpath '/home/hduser/Desktop/empdbnew.txt' into table empdbnew;
```

#Load data in external table from managed table.

```
hive>INSERT INTO hive_table_emp select * from empdbnew;

hive> select * from hive_table_emp;
OK
  1        deepali120000
  2        mahesh          30000
  3        mangesh         25000
4          ram     39000
5          brijesh 40000
6          john    300000
Time taken: 0.52 seconds, Fetched: 6 row(s)
```

#display records where salary is greater than 40000

```
hive> select * from hive_table_emp where esal>40000;
OK
1          deepali120000
6          john    300000
Time taken: 0.546 seconds, Fetched: 2 row(s)
```

#Check hbase for updates(The records are available in associated Hbase table)

```
hbase(main):008:0> scan 'emphive'
ROW                 COLUMN+CELL
   1                column=cf:esal, timestamp=1522212425665, value=120000
   1                column=cf:name, timestamp=1522212425665, value=deepali
   2                column=cf:esal, timestamp=1522212425665, value=30000
   2                column=cf:name, timestamp=1522212425665, value=mahesh
   3                column=cf:esal, timestamp=1522212425665, value=25000
   3                column=cf:name, timestamp=1522212425665, value=mangesh
```

```
4    column=cf:esal, timestamp=1522212425665, value=39000
4    column=cf:name, timestamp=1522212425665, value=ram
5    column=cf:esal, timestamp=1522212425665, value=40000
5    column=cf:name, timestamp=1522212425665, value=brijesh
6    column=cf:esal, timestamp=1522212425665, value=300000
6    column=cf:name, timestamp=1522212425665, value=john
6 row(s) in 0.0700 seconds
```

 # Creating external table in Hive referring to Hbase
#referring to flight table created in Hbase

```
CREATE external TABLE hbase_flight_new(fno int, fsource string,fdest string,fsh_at string,fsh_dt string,fsch_delay
string,delay int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,finfo:source,finfo:dest,fsch:at,fsch:dt,fsch:delay,delay:dl")
TBLPROPERTIES ("hbase.table.name" = "flight");

hive> CREATE external TABLE hbase_flight_new(fno int, fsource string,fdest string,fsh_at string,fsh_dt
string,fsch_delay string,delay int)
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES ("hbase.columns.mapping"
=":key,finfo:source,finfo:dest,fsch:at,fsch:dt,fsch:delay,delay:dl")
    > TBLPROPERTIES ("hbase.table.name" = "flight");
OK
Time taken: 0.361 seconds
```

#table created in hive

```
hive>  show tables;
OK
abc ddl_hive
emp empdata
empdata1
empdata2
empdbnew
hbase_flight
hbase_flight1
hbase_flight_new
hbase_table_1
hive_table_emp
Time taken: 0.036 seconds, Fetched: 12 row(s)
```

 # Display records from that table

```
hive> select * from hbase_flight_new;
OK
1    pune    mumbai      10.25a.m.    11.25 a.m.    5min   10
2    pune    kolkata7.00a.m.      7.30a.m.      2 min  4
3    mumbai      pune    12.30p.m.    12.45p.m.     1 min  5
4    mumbai      delhi   2.00p.m.     2.45p.m.      10 min 16
Time taken: 0.581 seconds, Fetched: 4 row(s)
```


e) Find the average departure delay per day in 2008.

#calculate average delay


```
hive> select sum(delay) from hbase_flight_new;
```

```
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions.
Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20180328130004_47384e9a-7490-4dfb-809d-ae240507bfab
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>
Starting Job = job_1522208646737_0003, Tracking URL =
http://localhost:8088/proxy/application_1522208646737_0003/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1522208646737_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2018-03-28 13:00:20,256 Stage-1 map = 0%, reduce = 0%
2018-03-28 13:00:28,747 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.68 sec
2018-03-28 13:00:35,101 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 6.26 sec
MapReduce Total cumulative CPU time: 6 seconds 260 msec
Ended Job = job_1522208646737_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1  Cumulative CPU: 6.26 sec  HDFS Read: 9095 HDFS Write:102
SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 260 msec
OK
  35
Time taken: 31.866 seconds, Fetched: 1 row(s)
hive>
```

d) Create index on Flight information Table

#create index on hbase_flight_new

```
CREATE INDEX hbasefltnew_index
ON TABLE hbase_flight_new (delay)
AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
WITH DEFERRED REBUILD;

SHOW INDEX ON hbase_flight_new;
```

#create index on table hbase_flight_new

```
hive> CREATE INDEX hbasefltnew_index
    > ON TABLE hbase_flight_new (delay)
    > AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
    > WITH DEFERRED REBUILD;
  OK
Time taken: 0.74 seconds
```

#show index on table hbase_flight_new

```
hive> SHOW INDEX ON hbase_flight_new;
OK
hbasefltnew_index       hbase_flight_new       delay
        default_hbase_flight_new_hbasefltnew_index_____  compact
Time taken: 0.104 seconds, Fetched: 1 row(s)
```

#join two tables in Hive

#create table B for join

```
hive> create table empinfo(empno int, empgrade string) row format delimited fields terminated by','
stored as textfile;
```

#Load Data into table

```
hive> load data local inpath '/home/hduser/Desktop/empinfo.txt' into table empinfo;Loading
data to table default.empinfo
OK
Time taken: 0.552 seconds
```

#insert data into the table

```
hive> load data local inpath '/home/hduser/Desktop/empinfo.txt' into table empinfo;
```

 # Table A empdbnew

```
hive> select * from empdbnew;
OK
  1       deepali120000
  2       mahesh          30000
  3       mangesh         25000
4         ram     39000
5         brijesh 40000
6         john    300000
Time taken: 0.258 seconds, Fetched: 6 row(s)
```

 # Table B empinfo

```
hive> select * from empinfo;
OK
  1       A
  2       B
  3       B
  4       B
  5       B
  6       A
Time taken: 0.207 seconds, Fetched: 6 row(s)
```

#Join two tables(empdbnew with empinfo on empno)

```
hive> SELECT eno, ename, empno, empgrade FROM empdbnew JOIN empinfo ON eno = empno;
```


#Join==> Result

```
hive> SELECT eno, ename, empno, empgrade
    > FROM empdbnew JOIN empinfo ON eno = empno;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions.Consider
using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20180328153258_bc345f46-a1f1-4589-ac5e-4c463834731aTotal
jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1In 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>
Starting Job = job_1522208646737_0005, Tracking URL =
http://localhost:8088/proxy/application_1522208646737_0005/
Kill Command = /usr/local/hadoop/bin/hadoop job  -kill job_1522208646737_0005
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1 2018-
03-28 15:33:09,615 Stage-1 map = 0%, reduce = 0%
2018-03-28 15:33:18,231 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.17 sec
2018-03-28 15:33:24,476 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.61 secMapReduce
Total cumulative CPU time: 10 seconds 610 msec
Ended Job = job_1522208646737_0005
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1  Cumulative CPU: 10.61 sec  HDFS Read: 15336 HDFS Write:235
SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 610 msecOK
  1        deepali1      A
  2       mahesh        2             B
  3        mangesh     3             B
  4    ram      4         B
  5        brijesh 5     B
  6    john     6         A
Time taken: 26.915 seconds, Fetched: 6 row(s)
```

EXPERIMENT 4

TITLE: Perform the following operations using Python on the Facebook metrics data sets

OBJECTIVE:

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

SOFTWARE REQUIREMENTS:

1. Windows 8 or above
2. Python SDK
3. IDE (p0079charm, anaconda, cloud notebook)

PROBLEM STATEMENT: Perform the following operations using Python on the Facebook metrics data sets
a. Create data subsets
b. Merge Data
c. Sort Data
d. Transposing Data
e. Shape and reshape Data

THEORY:

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd


# In[2]:


air = pd.read_csv('airquality.csv')


# In[3]:


air.shape


# In[4]:
```

```
air.head()

# In[5]:

air.count()

# In[6]:

air.isnull().sum()

# In[7]:

air.describe()

# In[8]:

air.info()

# In[9]:

A = air.dropna()

# In[10]:

A.shape

# In[11]:

A = air.fillna(0)

# In[12]:

A.shape

# In[13]:
```

```
A.head()

# In[14]:

A = air.fillna(method='pad')

# In[15]:

A.head()

# In[16]:

A = air.fillna(method='backfill')

# In[17]:

A.head()

# In[18]:

import numpy as np

# In[19]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].mean())

# In[20]:

A.head()

# In[21]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].mean())

# In[22]:
```

```
A.head()

# In[23]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].median())

# In[24]:

A.head()

# In[25]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].mode())

# In[26]:

A.head()

# In[27]:

from sklearn.impute import SimpleImputer

# In[28]:

from sklearn.impute import SimpleImputer

# In[29]:

from sklearn.impute import SimpleImputer

# In[30]:

from sklearn.impute import SimpleImputer
```

```
# In[31]:

from sklearn.impute import SimpleImputer

# In[32]:

imp = SimpleImputer(missing_values=np.nan,strategy='mean')

# In[33]:

A = imp.fit_transform(air)

# In[34]:

A

# In[35]:

A = pd.DataFrame(A, columns=air.columns)

# In[36]:

A.head()

# In[37]:

from sklearn.model_selection import train_test_split

# In[38]:

len(A)

# In[39]:

train, test = train_test_split(A)
```

```
# In[40]:

len(train)

# In[41]:

len(test)

# In[42]:

train.head()

# In[43]:

train, test = train_test_split(A, test_size = 0.20)

# In[44]:

len(test)

# In[45]:

len(train)

# In[46]:

A.describe()

# In[47]:

from sklearn.preprocessing import StandardScaler

# In[48]:

scaler = StandardScaler()
```

```
# In[49]:

B = scaler.fit_transform(A)

# In[50]:

pd.DataFrame(B).describe()

# In[51]:

from sklearn.preprocessing import MinMaxScaler

# In[52]:

scaler = MinMaxScaler()

# In[53]:

B = scaler.fit_transform(A)

# In[54]:

pd.DataFrame(B).describe()

# In[55]:

B = pd.DataFrame(B).describe()

# In[56]:

from sklearn.preprocessing import Binarizer

# In[57]:
```

```
bin = Binarizer(threshold=0.5)

# In[58]:

B = bin.fit_transform(B)

# In[59]:

pd.DataFrame(B)

# In[60]:

data=pd.read_csv('student.csv')

# In[61]:

from sklearn.preprocessing import LabelEncoder

# In[62]:

le = LabelEncoder()

# In[63]:

B = le.fit_transform(data['name'])

# In[64]:

B

# In[65]:

B = data[:]

# In[66]:
```

```
B['name'] = le.fit_transform(B['name'])

# In[67]:

B

# In[68]:

A

# In[69]:

from sklearn.linear_model import LinearRegression

# In[80]:

X=A['Ozone'].values

# In[81]:

X=X.reshape(-1,1)

# In[82]:

Y = A['Temp']

# In[83]:

model = LinearRegression()

# In[84]:

model.fit(X,Y)

# In[85]:
```

```
model.score(X,Y)*100

#
# In[86]:

model.predict([[128]])

# In[88]:

import matplotlib.pyplot as plt

# In[89]:

plt.scatter(X,Y)

# In[ ]:
```

CONCLUSION: Thus we have learnt how to Perform the different reshape operations using Python **.**

EXPERIMENT 5

TITLE: -Perform the following operations using Python on the Air quality and Heart Diseases data sets

OBJECTIVE:

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

SOFTWARE REQUIREMENTS:

1. 1 Windows 8 or above
2. Python SDK
3. IDE (p0079charm, anaconda, cloud notebook)

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

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd


# In[2]:


air = pd.read_csv('airquality.csv')
```

```
# In[3]:

air.shape

# In[4]:

air.head()

# In[5]:

air.count()

# In[6]:

air.isnull().sum()

# In[7]:

air.describe()

# In[8]:

air.info()

# In[9]:

A = air.dropna()

# In[10]:

A.shape

# In[11]:

A = air.fillna(0)
```

```
# In[12]:

A.shape

# In[13]:

A.head()

# In[14]:

A = air.fillna(method='pad')

# In[15]:

A.head()

# In[16]:

A = air.fillna(method='backfill')

# In[17]:

A.head()

# In[18]:

import numpy as np

# In[19]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].mean())

# In[20]:

A.head()
```

```
# In[21]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].mean())

# In[22]:

A.head()

# In[23]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].median())

# In[24]:

A.head()

# In[25]:

A = air['Ozone'].replace(np.NaN,air['Ozone'].mode())

# In[26]:

A.head()

# In[27]:

from sklearn.impute import SimpleImputer

# In[28]:

from sklearn.impute import SimpleImputer

# In[29]:
```

```
from sklearn.impute import SimpleImputer

# In[30]:

from sklearn.impute import SimpleImputer

# In[31]:

from sklearn.impute import SimpleImputer

# In[32]:

imp = SimpleImputer(missing_values=np.nan,strategy='mean')

# In[33]:

A = imp.fit_transform(air)

# In[34]:

A

# In[35]:

A = pd.DataFrame(A, columns=air.columns)

# In[36]:

A.head()

# In[37]:

from sklearn.model_selection import train_test_split

# In[38]:
```

```
len(A)

# In[39]:

train, test = train_test_split(A)

# In[40]:

len(train)

# In[41]:

len(test)

# In[42]:

train.head()

# In[43]:

train, test = train_test_split(A, test_size = 0.20)

# In[44]:

len(test)

# In[45]:

len(train)

# In[46]:

A.describe()

# In[47]:
```

```
from sklearn.preprocessing import StandardScaler

# In[48]:

scaler = StandardScaler()

# In[49]:

B = scaler.fit_transform(A)

# In[50]:

pd.DataFrame(B).describe()

# In[51]:

from sklearn.preprocessing import MinMaxScaler

# In[52]:

scaler = MinMaxScaler()

# In[53]:

B = scaler.fit_transform(A)

# In[54]:

pd.DataFrame(B).describe()

# In[55]:

B = pd.DataFrame(B).describe()
```

```
# In[56]:

from sklearn.preprocessing import Binarizer

# In[57]:

bin = Binarizer(threshold=0.5)

# In[58]:

B = bin.fit_transform(B)

# In[59]:

pd.DataFrame(B)

# In[60]:

data=pd.read_csv('student.csv')

# In[61]:

from sklearn.preprocessing import LabelEncoder

# In[62]:

le = LabelEncoder()

# In[63]:

B = le.fit_transform(data['name'])

# In[64]:

B
```

```
# In[65]:

B = data[:]

# In[66]:

B['name'] = le.fit_transform(B['name'])

# In[67]:

B

# In[68]:

A

# In[69]:

from sklearn.linear_model import LinearRegression

# In[80]:

X=A['Ozone'].values

# In[81]:

X=X.reshape(-1,1)

# In[82]:

Y = A['Temp']

# In[83]:

model = LinearRegression()
```

```
# In[84]:

model.fit(X,Y)

# In[85]:

model.score(X,Y)*100

#
# In[86]:

model.predict([[128]])

# In[88]:

import matplotlib.pyplot as plt

# In[89]:

plt.scatter(X,Y)

# In[ ]:
```

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

EXPERIMENT 6

TITLE: Integrate 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 Python.
2. To study detailed concept PyHadoop .

SOFTWARE REQUIREMENTS:

1. Windows 8 and above
2. GNU C Compiler
3. Hadoop
4. Java
5. Python IDE

PROBLEM STATEMENT: Integrate Python and Hadoop and perform the following operationson forest fire dataset

1) Data analysis using the Map Reduce in Pyhadoop
2) Data mining in Hive

THEORY:

EXPERIMENT 7

TITLE: Visualize the data using Python by plotting the graphs for assignment no. 4 and 5

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. GNU C Compiler
3. Hadoop
4. Java
5. R Studio

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

THEORY:

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

EXPERIMENT 8

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. Ubuntu 14.04 / 14.10
2. GNU C Compiler
3. Hadoop
4. Java
5. 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

1D/Linear

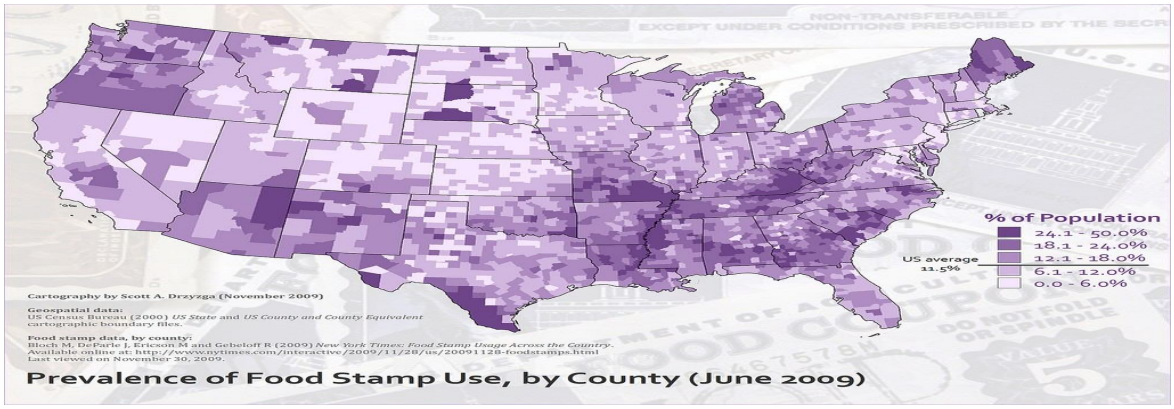
Examples:

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

2D/Planar (especially geospatial)

Examples (geospatial):

- **choropleth**



3D/Volumetric

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.

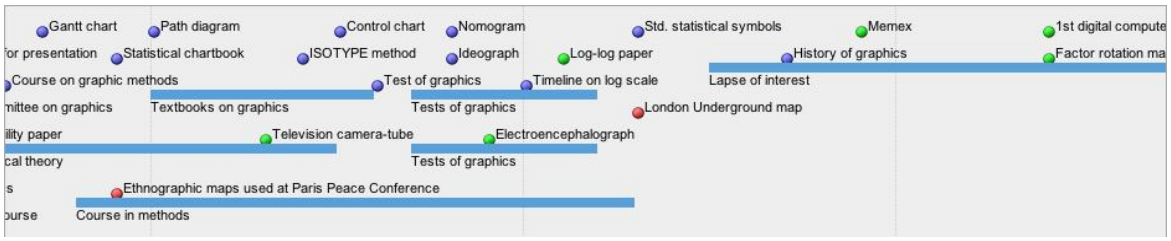
- computer simulations

  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.

Temporal

Examples:

- **timeline**

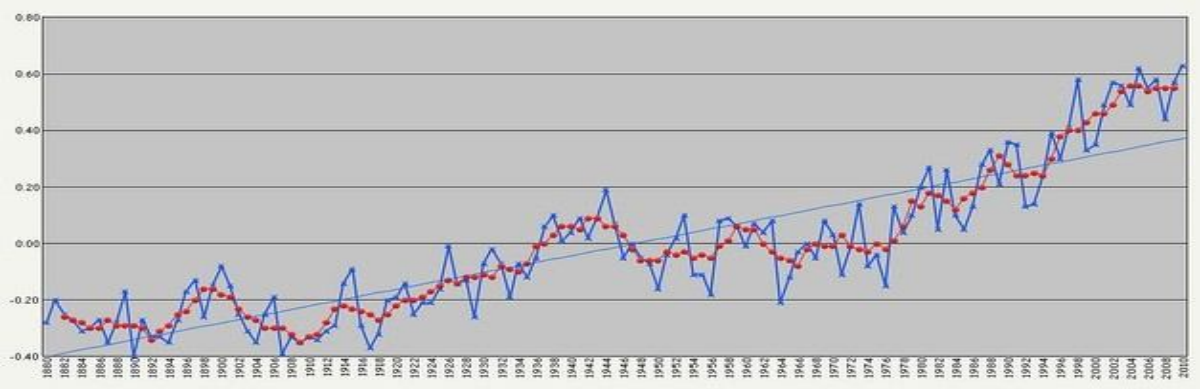


Tools: SIMILE Timeline, TimeFlow, Timeline JS, Excel

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.

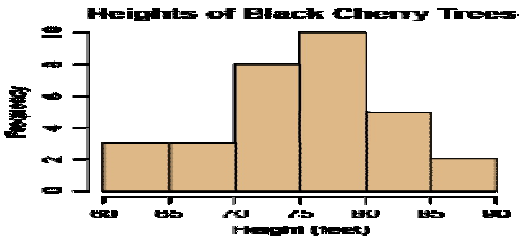
- **time series**



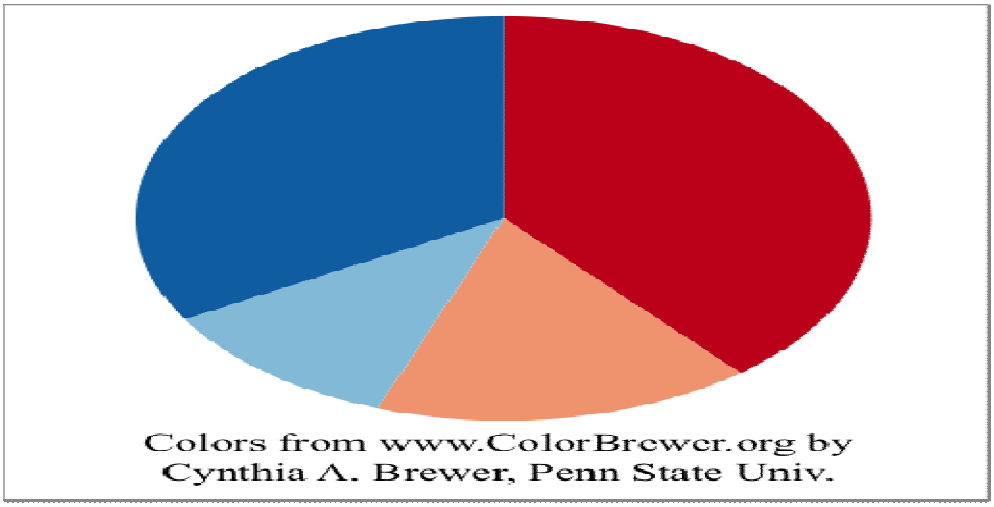
nD/Multidimensional

Examples (category proportions, counts):

- **histogram**



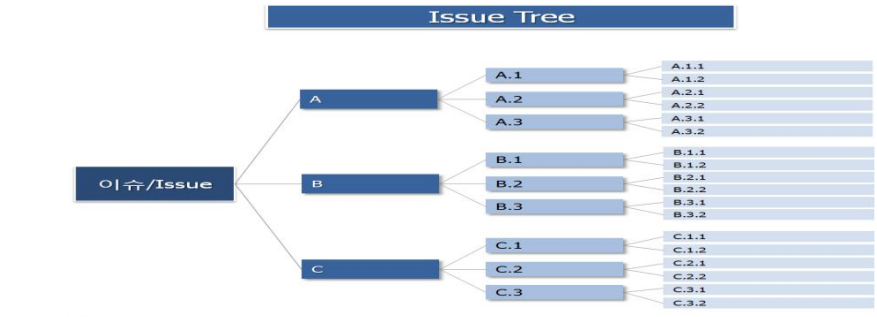
- **pie chart**



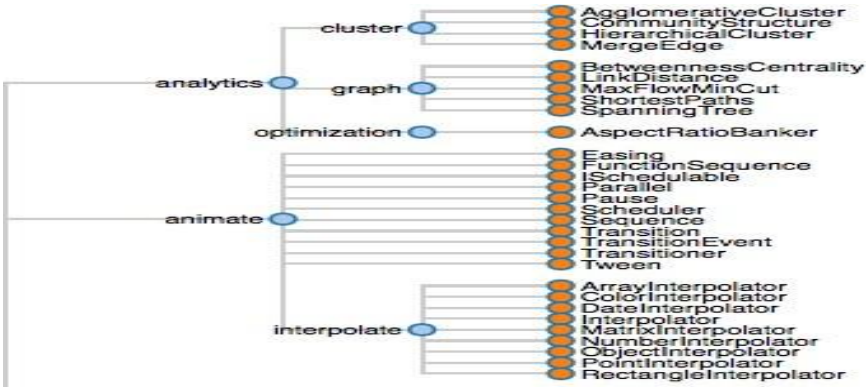
Tree/Hierarchical

Examples:

- **general tree visualization**



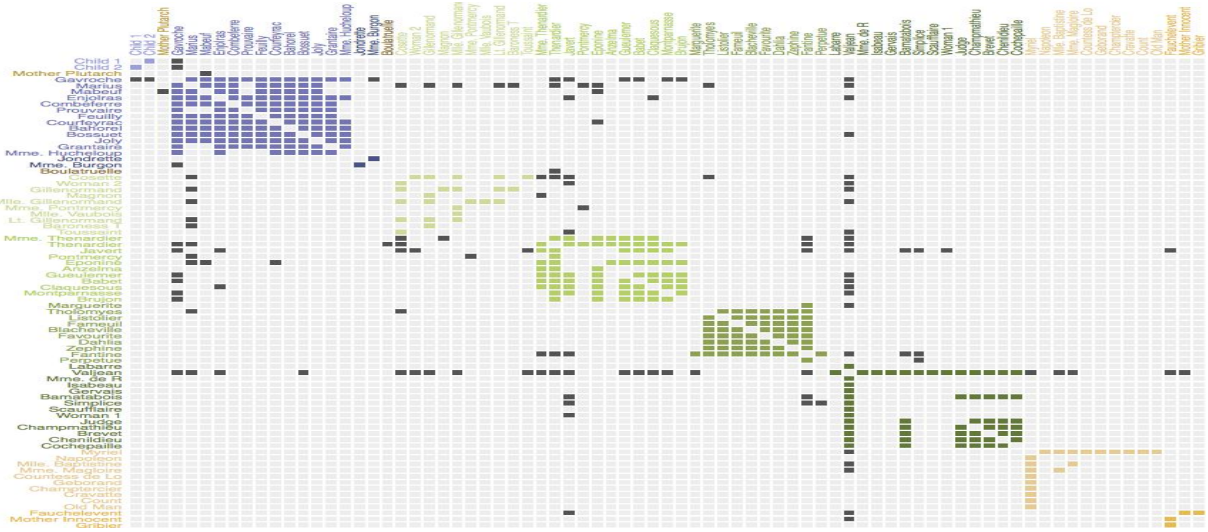
- **dendrogram**



Network

Examples:

- **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.

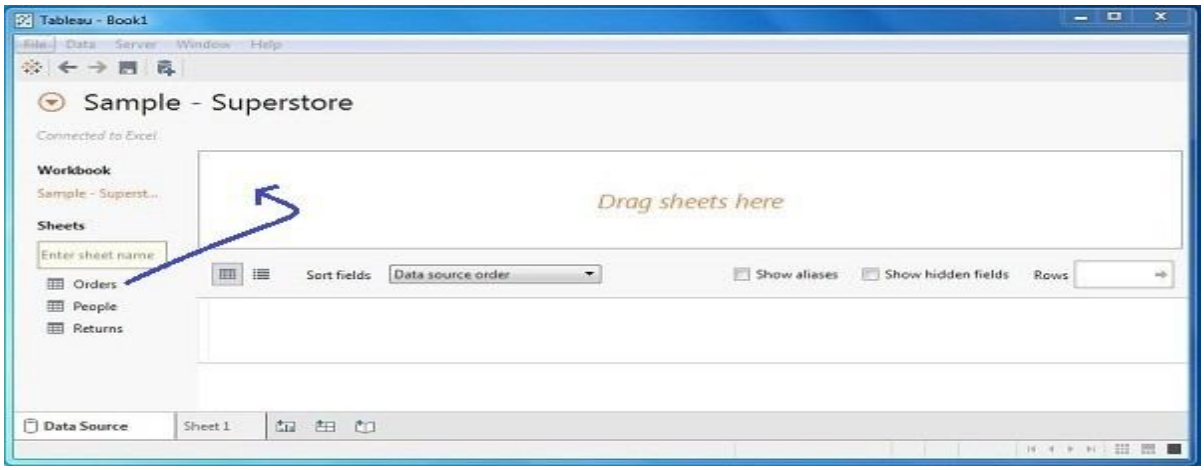
These three steps are −

- **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**.

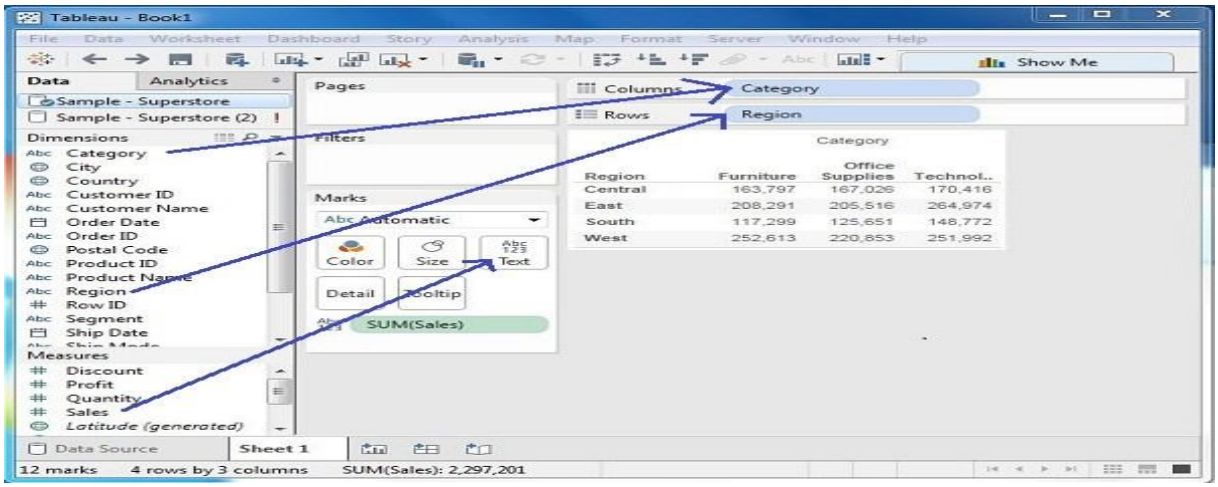
Connect to a Data Source

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

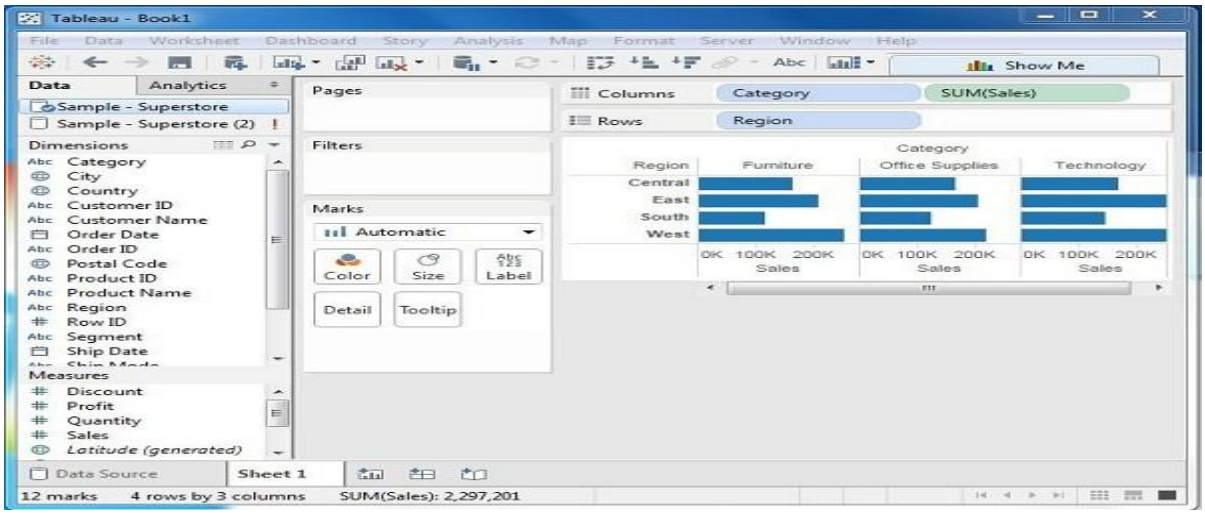
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region** 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.



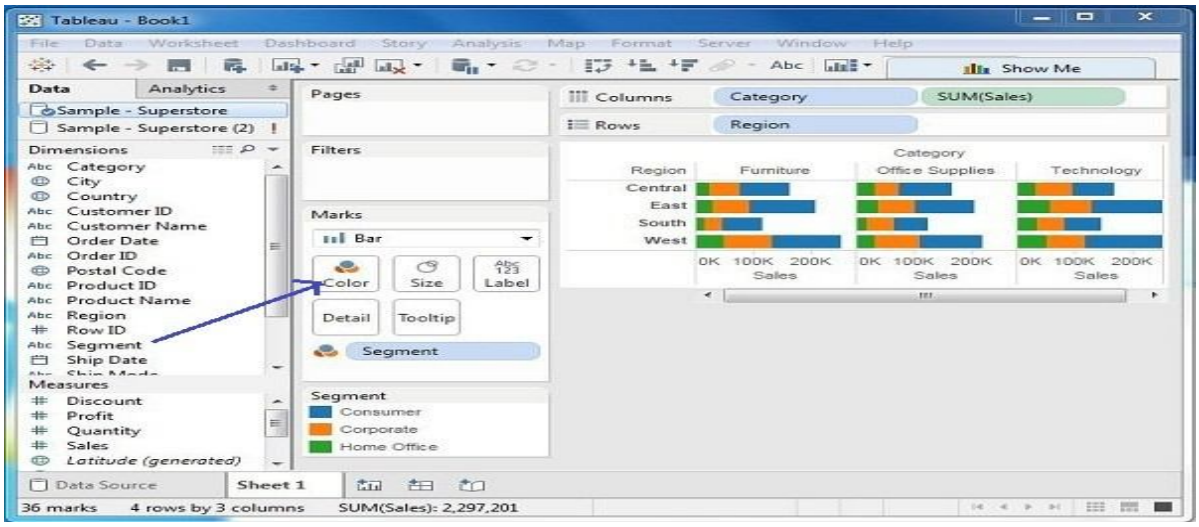
Apply Visualization Technique

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.



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.