

Department Vision and Mission

Vision

The department endeavors to be recognized globally as a center of academic excellence & research in Information Technology.

Mission

To inculcate research culture among students by imparting Information Technology related fundamental knowledge, recent technology trends and ethics to get recognized as globally acceptable and socially responsible professionals.

Course Outcomes and CO-PO/PSO Mapping

Class: T.E

Semester: II

Subject Code: 314457

Subject Name: Data Science And Big Data Analytics Lab

Course Outcome: (Co's)	
C19317.1	Students will be able to design and execute distributed application using MapReduce framework by installing Hadoop and perform various operations such as create, drop, alter load data, insert values and create index on database tables using HiveQL.
C19317.2	Students will be able to demonstrate operations such as subset creation, merge, sort, transpose, shape and reshape data and perform Data cleaning, Data integration, Data transformation, Error correction and Data model building on various datasets using Python.
C19317.3	Students will be able to showcase data mining in Hive and analyze data using the Map Reduce in PyHadoop by Integrating Python and Hadoop.
C19317.4	Students will be able to examine the effect of data visualization using Python libraries such as Matplotlib, seaborn and Tableau tool by plotting the graphs on various datasets.
C19317.5	Students will be able to design review scrapper for any ecommerce website to fetch real time comments and other parameters and develop mini-project in a group using different predictive models techniques to solve any real life problem using Python.

- Mapping of COs with Program Outcomes and Program Specific Outcomes

Co-Po articulation matrix															
CO	PO 1	PO2	PO3	PO4	PO 5	PO 6	PO 7	PO8	PO9	PO1 0	PO1 1	PO12	PSO1	PSO 2	PSO3
C19317.1	3	3	3	3	3	-	-	-	1	1	-	3	3	1	-
C19317.2	3	3	3	3	3	-	-	-	1	1	-	3	3	1	-
C19317.3	3	3	3	3	3	-	-	-	1	1	-	3	3	1	-
C19317.4	3	3	3	3	3	-	2	-	1	2	-	3	3	2	-
C19317.5	3	3	3	3	3	3	3	2	3	3	1	3	3	3	3
AVG	3	3	3	3	3	3	2.5	2	1.4	1.6	1	3	3	1.6	2

PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2021-22

LAB MANUAL

DEPARTMENT: INFORMATION TECHNOLOGY

CLASS: T.E.

SEMESTER: VI

Subject Name: DS & BDA Lab

INDEX OF LAB EXPERIMENTS

LAB EXPT. NO.	PROBLEM STATEMENT	LAST DATE OF COMPLETION
	Part A : Assignments based on the Hadoop	
1.	Study of Hadoop Installation. a. Single Node b. Multiple Node	3RD WEEK OF JAN
2.	Design a distributed application using MapReduce(Using Java) 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.	1ST WEEK OF FEB
3.	Write an application using HiveQL for flight information system which will include a. Creating, Dropping, and altering Database tables. b. Creating an external Hive table. c. Load table with data, insert new values and field in the table, Join tables with Hive d. Create index on Flight Information Table e. Find the average departure delay per day in 2008.	3RD WEEK OF FEB
	Part B : Assignments based on Data Analytics using Python	

5.	<p>Perform the following operations using Python on the Facebook metrics data sets</p> <ul style="list-style-type: none"> a. Create data subsets b. Merge Data c. Sort Data d. Transposing Data e. Shape and reshape Data 	4TH WEEK OF FEB
6.	<p>Perform the following operations using Python on the Air quality and Heart Diseases data sets</p> <ul style="list-style-type: none"> a. Data cleaning b. Data integration c. Data transformation d. Error correcting e. Data model building 	2ND WEEK OF MAR
7.	<p>Integrate Python and Hadoop and perform the following operations on forest fire dataset</p> <ul style="list-style-type: none"> a. Data analysis using the Map Reduce in PyHadoop b. Data mining in Hive 	3RD WEEK OF MAR
8.	<p>Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 2 and 3 (Group B)</p>	4TH WEEK OF MAR
9.	<p>Perform the following data visualization operations using Tableau on Adult and Iris datasets.</p> <ul style="list-style-type: none"> a. 1D (Linear) Data visualization b. 2D (Planar) Data Visualization c. 3D (Volumetric) Data Visualization d. Temporal Data Visualization e. Multidimensional Data Visualization f. Tree/ Hierarchical Data visualization g. Network Data visualization <p>Perform the data visualization operations using Tableau to get answers to various business questions on Retail dataset.</p>	1ST WEEK OF APR

	<ul style="list-style-type: none"> a. Find and Plot top 10 products based on total sale b. Find and Plot product contribution to total sale c. Find and Plot the month wise sales in year 2010 in descending order d. Find and Plot most loyal customers based on purchase order e. Find and Plot yearly sales comparison f. Find and Plot country wise total sales price and show on Geospatial graph g. Find and Plot country wise popular product h. Find and Plot bottom 10 products based on total sale i. Find and Plot top 5 purchase order j. Find and Plot most popular products based on sales k. Find and Plot half yearly sales for the year 2011 l. Find and Plot country wise total sales quantity and show on Geospatial graph 	
	Group C: Model Implementation	
10.	Create a review scrapper for any ecommerce website to fetch real time comments, reviews, ratings, comment tags, customer name using Python.	3RD WEEK OF APR
11.	<p>Develop a mini project in a group using different predictive models techniques to solve any real life problem. (Refer link dataset- https://www.kaggle.com/tanmoyie/us-graduate-schools-admission-parameters)</p> <p>Design and implement any Data Science application using R/Python. Obtain data, scrub data (Data cleaning). Explore data, Prepare and validate data model and interpret data (Data visualization). Visualize data using any visualization tool like Matplotlib, ggplot, Tableau etc. Prepare Project Report.</p>	3RD WEEK OF APR

Ms.Deepali D.Londhe
Subject Coordinator

Head of Department

Part-A

Assignments based on the Hadoop

Assignment: 1

AIM: Study of Hadoop Installation.

PROBLEM STATEMENT / DEFINITION:

1. Study of Hadoop Installation on Single Node.
2. Study of Hadoop Installation on Multiple Nodes.

OBJECTIVE:

- I. To understand Installation & Configuration of Hadoop on single Node.
- II. To understand Installation & Configuration of Hadoop on multiple nodes.

THEORY:

a) THE Single Node:

Introduction

Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common and should be automatically handled by the framework. The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part called MapReduce. Hadoop splits files into large blocks and distributes them across nodes in a cluster. To process data, Hadoop transfers packaged code for nodes to process in parallel based on the data that needs to be processed. This approach takes advantage of data locality— nodes manipulating the data they have access to— to allow the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking. The base Apache Hadoop framework is composed of the following modules:

- ☐ **Hadoop Common** – contains libraries and utilities needed by other Hadoop modules;
- ☐ **Hadoop Distributed File System (HDFS)** – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;

- ❑ **Hadoop YARN** – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications; and
- ❑ **Hadoop MapReduce** – an implementation of the MapReduce programming model for large scale data processing.

A.1) Apache Hadoop Installation

Steps:

Open Terminal

Update the source list

```
aaditya@laptop:~$ sudo apt-get update
```

```
# The OpenJDK project is the default version of Java  
Install JDK
```

```
aaditya@laptop:~$ sudo apt-get install default-jdk`
```

```
aaditya@laptop:~$ java -version  
java version "1.7.0_95"
```

Adding a dedicated Hadoop user

```
aaditya@laptop:~$ sudo addgroup hadoop  
Adding group `hadoop' (GID 1002) ...  
Done.
```

```
aaditya@laptop:~$ sudo adduser --ingroup hadoop hduser  
(Press Enter)
```

```
Is the information correct? [Y/n] Y
```

Enter new password for new hduser

```
aaditya@laptop:~$ sudo adduser hduser sudo  
Adding user `hduser' to group `sudo' ...  
Adding user hduser to group sudo  
Done.
```

Logout from current user and log in to hduser

Installing SSH


```
aaditya@laptop:~$su hduser
```

```
hduser@laptop:~$ sudo apt-get install ssh
```

Create and Setup SSH Certificates

```
hduser@laptop:~$cd /home/hduser/
```

```
hduser@laptop:~$ ssh-keygen -t rsa -P ""
```

Generating public/private rsa key pair.

Enter file in which to save the key (/home/hduser/.ssh/id_rsa): (**press enter)

Created directory '/home/hduser/.ssh'.

Your identification has been saved in /home/hduser/.ssh/id_rsa.

Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.

The key fingerprint is:

50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 hduser@laptop

The key's randomart image is:

+--[RSA 2048]-----+

```
| .oo.o |  
| ..o=. o |  
| .+ . o . |  
| o = E |  
| S + |  
| . + |  
| O + |  
| O o |  
| o.. |
```

+-----+

```
hduser@laptop:$ cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys
```

```
hduser@laptop:$ ssh localhost
```

```
hduser@laptop:$ exit
```

```
hduser@laptop:$ cd /home/hduser
```

Download Hadoop

hduser@laptop:~\$

wget <http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz>

Or from <http://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz>

hduser@laptop:~\$ **tar xvzf hadoop-2.8.3.tar.gz**

hduser@laptop:~\$ **cd hadoop-2.8.3**

hduser@laptop:~\$ **sudo mkdir /usr/local/hadoop**

hduser@laptop:~/hadoop-2.7.2\$ **sudo mv * /usr/local/hadoop**

hduser@laptop:~/hadoop-2.7.2\$ **sudo chown -R hduser:hadoop /usr/local/hadoop**

hduser@laptop:~/hadoop-2.7.2\$ **sudo apt-get install vim**

Check Java

hduser@laptop:~\$ **readlink -f /usr/bin/javac**

/usr/lib/jvm/java-8-openjdk-amd64/bin/javac *** for 64 bit

/usr/lib/jvm/java-8-openjdk-i386/bin/javac ***for 32 bit

Set-up the Configuration Files

hduser@laptop:~\$ **vim ~/.bashrc**

#HADOOP VARIABLES START **Append this at end of file*******

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export HADOOP_INSTALL=/usr/local/hadoop

export PATH=\$PATH:\$HADOOP_INSTALL/bin

export PATH=\$PATH:\$HADOOP_INSTALL/sbin

export HADOOP_MAPRED_HOME=\$HADOOP_INSTALL

export HADOOP_COMMON_HOME=\$HADOOP_INSTALL

export HADOOP_HDFS_HOME=\$HADOOP_INSTALL

export YARN_HOME=\$HADOOP_INSTALL

export HADOOP_COMMON_LIB_NATIVE_DIR=\$HADOOP_INSTALL/lib/native

export HADOOP_OPTS="-Djava.library.path=\$HADOOP_INSTALL/lib"

#HADOOP VARIABLES END

Save the file and Exit (**esc:wq!** to save the file in vim)

hduser@laptop:~\$ **source ~/.bashrc**

```
hduser@laptop:~$ vim /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

(Change existing JAVA_HOME to /usr/lib/jvm/java-7-openjdk-amd64 if machine is 64 Bit or
If machine is 32 bit change it to /usr/lib/jvm/java-7-openjdk-i386)

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64  
Save and Exit
```

```
hduser@laptop:~$ vim /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

Open the file and enter the following in between <configuration></configuration> tag and save :

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

Create Temporary Directory to store App-data

```
hduser@laptop:~$ sudo mkdir -p /app/hadoop/tmp
```

```
hduser@laptop:~$ sudo chown hduser:hadoop /app/hadoop/tmp
```

```
hduser@laptop:~$ vim /usr/local/hadoop/etc/hadoop/core-site.xml
```

Open the file and enter the following in between the <configuration></configuration> tag:

```
<property>  
<name>hadoop.tmp.dir</name>  
<value>/app/hadoop/tmp</value>  
<description>A base for other temporary directories.</description>  
</property>
```

```
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</value></property>
```

```
hduser@laptop:~$ cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
/usr/local/hadoop/etc/hadoop/mapred-site.xml
```

```
hduser@laptop:~$ vim /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

Add these properties in <configuration></configuration> tag:

```
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
</property>

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

Save and Exit

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
```

```
hduser@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
```

```
hduser@laptop:~$ sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

```
hduser@laptop:~$ vim /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

Open the file and add the following properties between the <configuration></configuration> tag:

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

```
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
```

```
hduser@laptop:~$ hadoop namenode -format
```

Starting Hadoop

```
hduser@laptop$: /usr/local/hadoop/sbin/start-all.sh
```

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

```
hduser@laptop$: jps
```

```
9026 NodeManager
```

```
7348 NameNode
```

```
9766 SecondaryNameNode
```

```
8887 ResourceManager
```

```
7507 DataNode
```

Hadoop UI visible in browser at **localhost:50070**

MR Job progress and history UI visible at **localhost:8088**

To Stop Hadoop

```
hduser@laptop$: /usr/local/hadoop/sbin/stop-all.sh
```

```
*****Done*****
```

Conclusion: In this way single node Hadoop was installed & configured on Ubuntu for BigData analytics.

Reference:<http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/SingleCluster.html>

Downloading and Installing the Cloudera VM Instructions (Windows & Ubuntu)

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 VirtualBox 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.

For Ubuntu Select [VirtualBox 5.2.44](#) (released July 14 2020) *Virtual box for Linux Hosts: Ubuntu 32bit|64bit*

Open Terminal -

```
$ sudo apt-get update
```

```
$ sudo apt-get install VirtualBox 5.2
```

```
$ sudo apt-get -f install
```

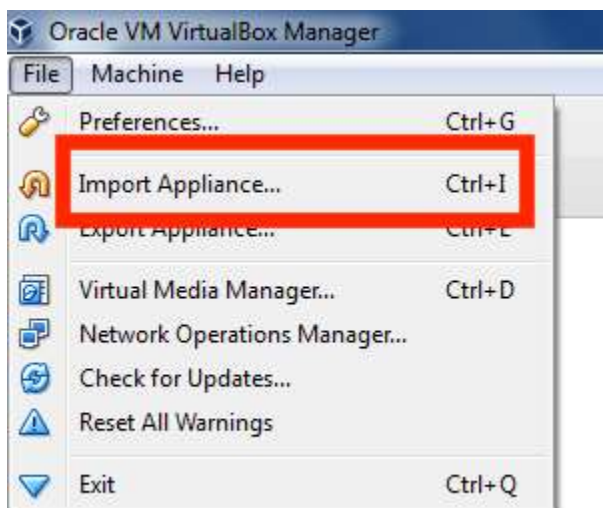
2. **Download the Cloudera VM.** Download the Cloudera VM from https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.4.2-0-virtualbox.zip. The VM is over 4GB, so will take some time to download.

3. **Unzip the Cloudera VM:**

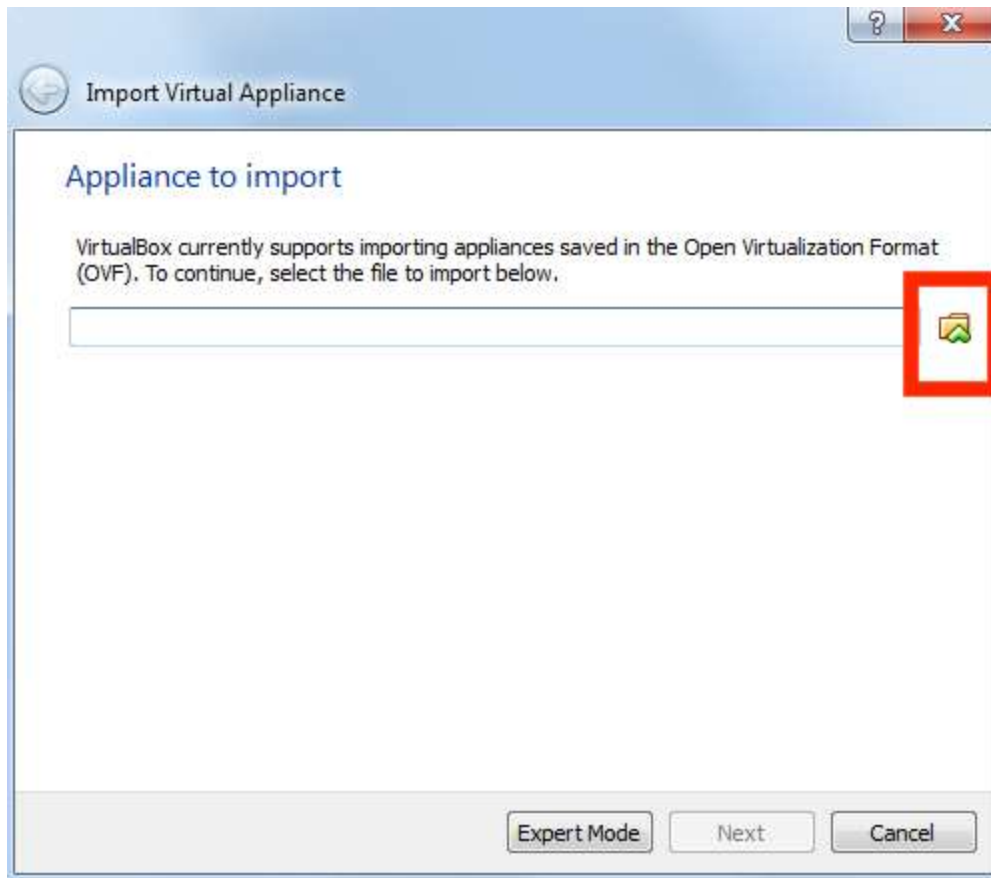
Right-click cloudera-quickstart-vm-5.4.2-0-virtualbox.zip and select “Extract All...”

4. **Start VirtualBox.**

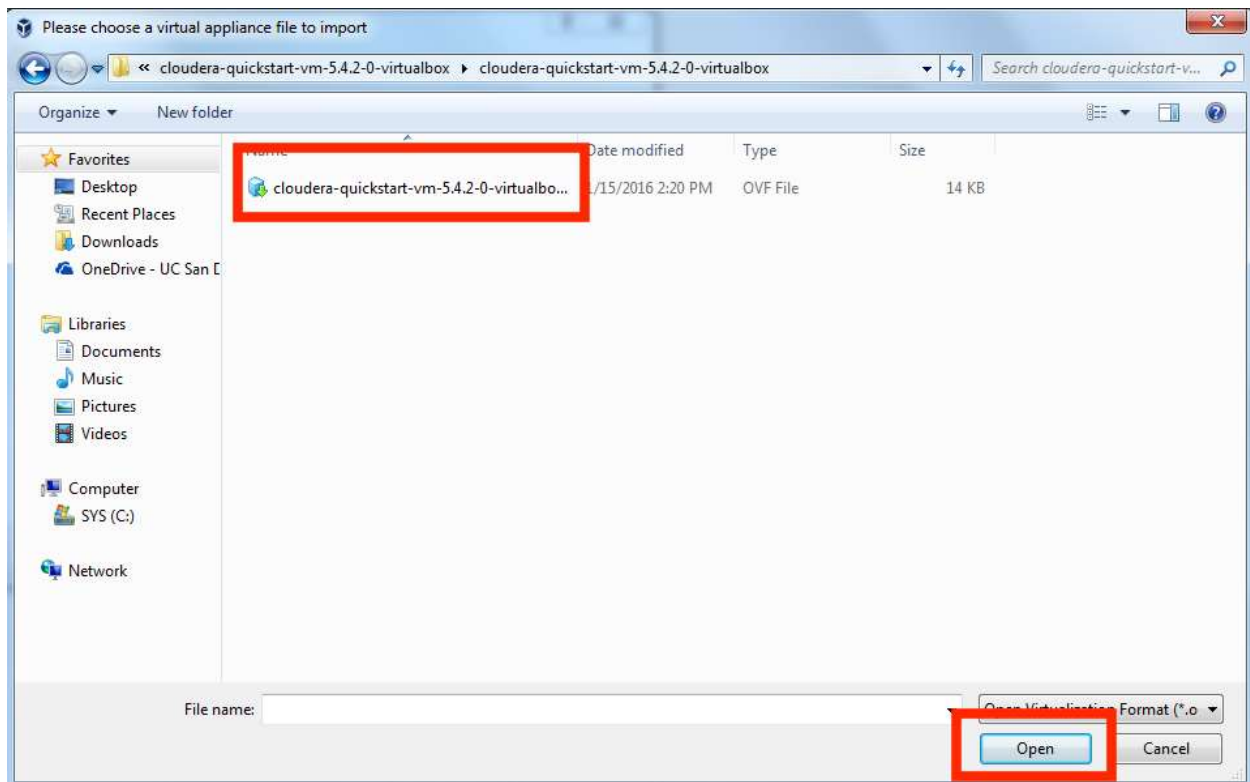
5. **Begin importing.** Import the VM by going to File -> Import Appliance



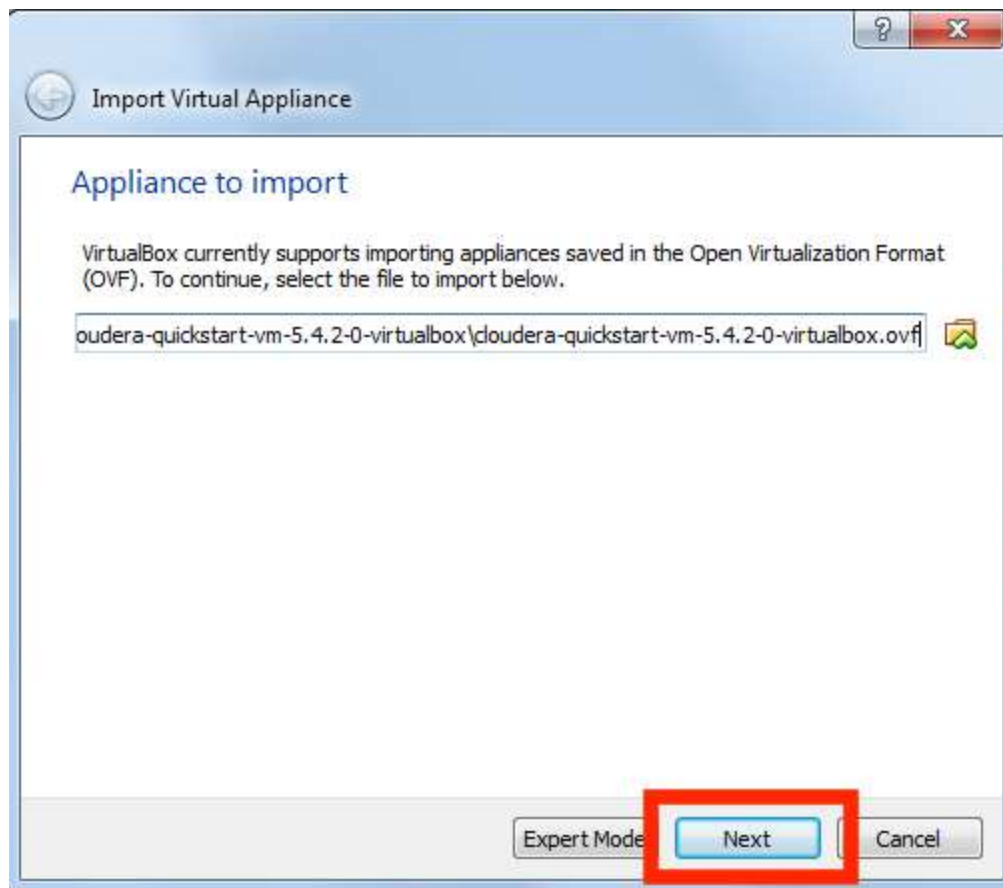
6. **Click the Folder icon.**



7. Select the **cloudera-quickstart-vm-5.4.2-0-virtualbox.ovf** from the Folder where you unzipped the VirtualBox VM and click Open.



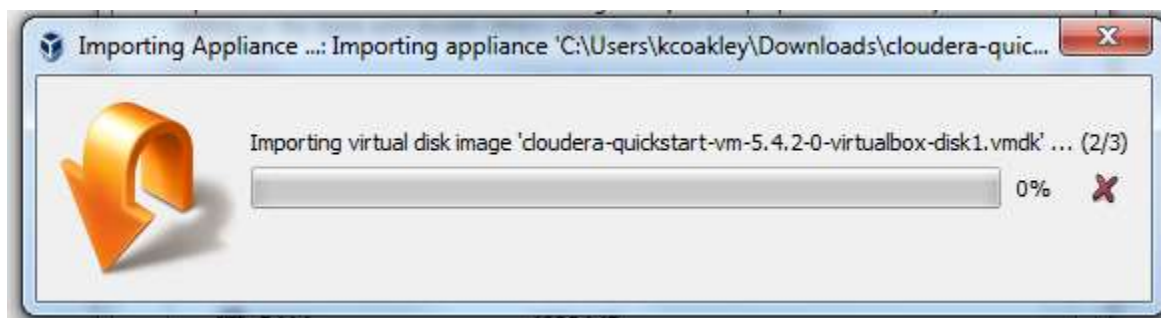
8. Click Next to proceed.



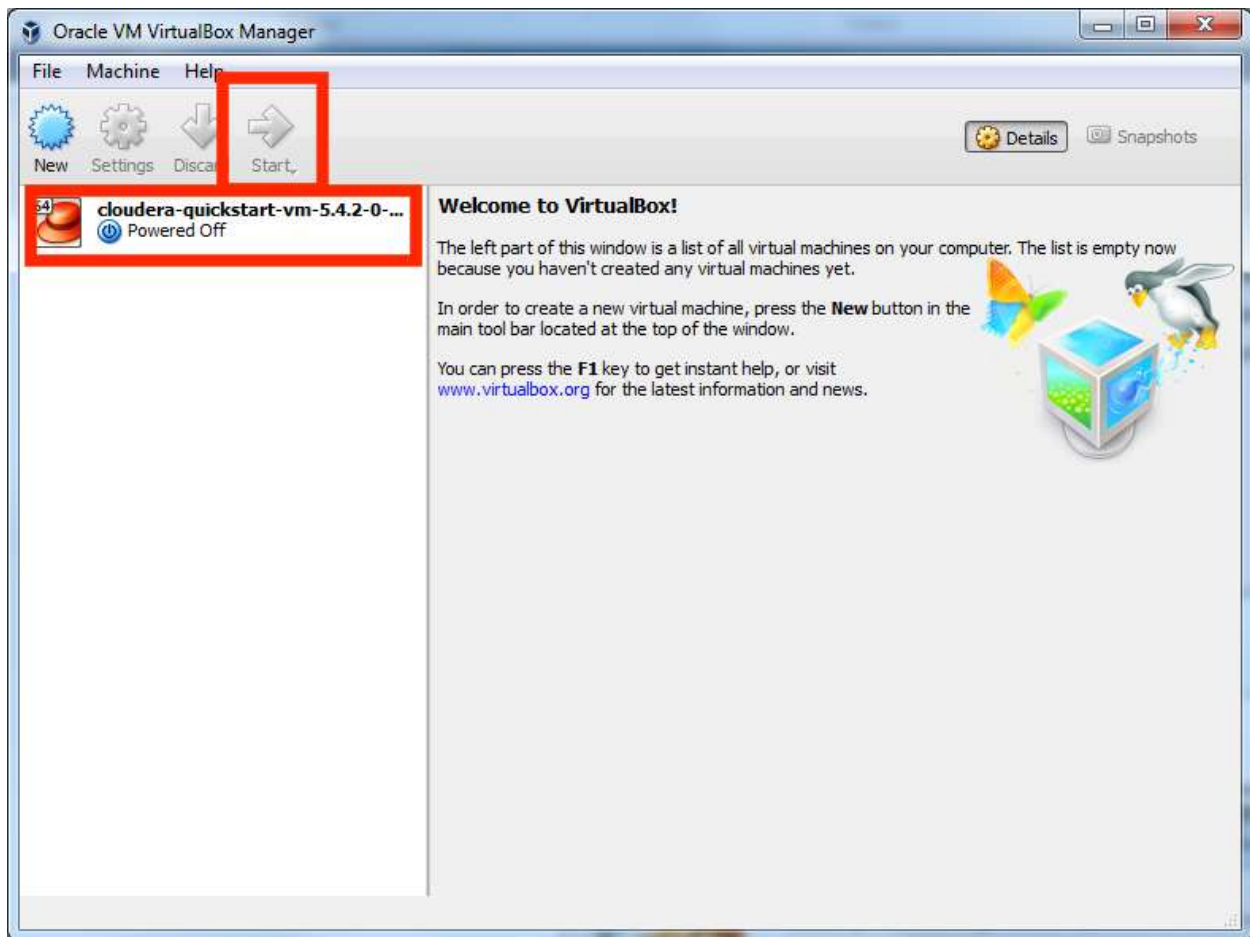
9. Click Import.



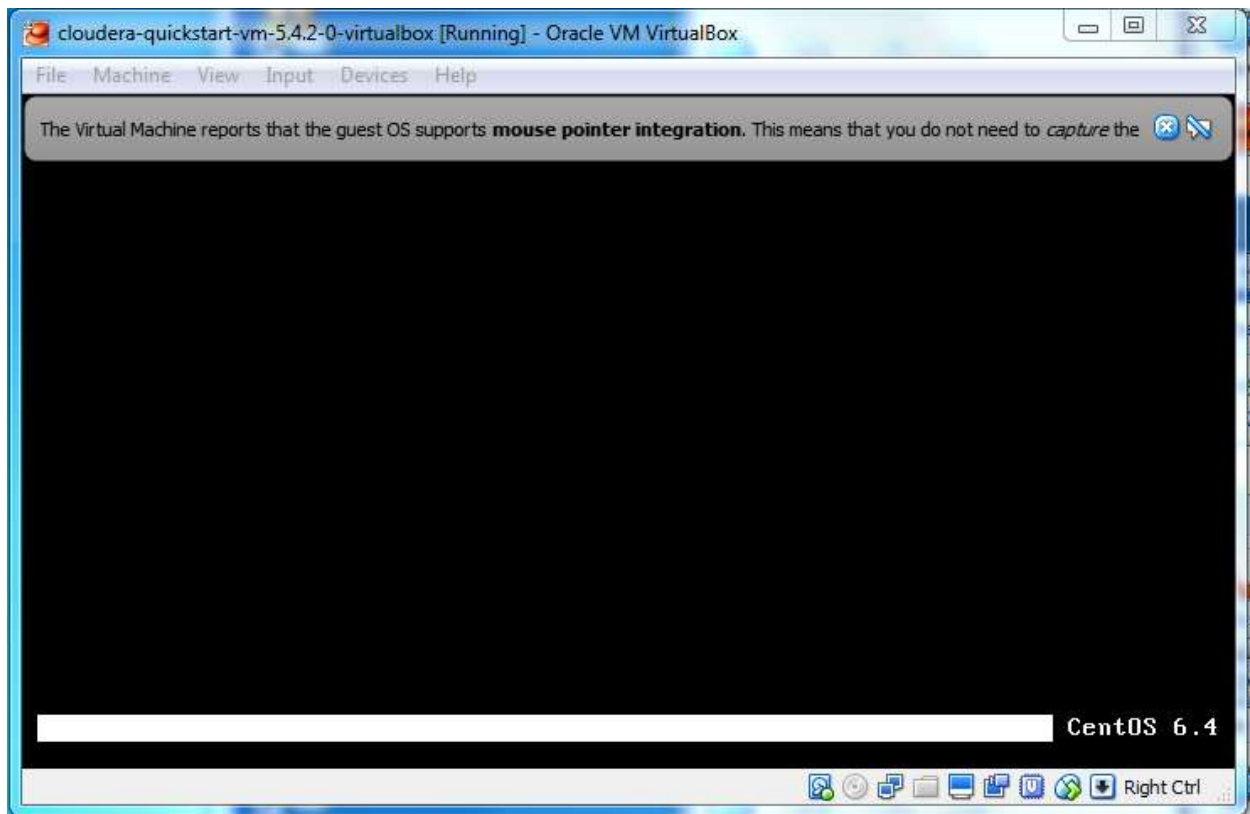
10. The virtual machine image will be imported. This can take several minutes.



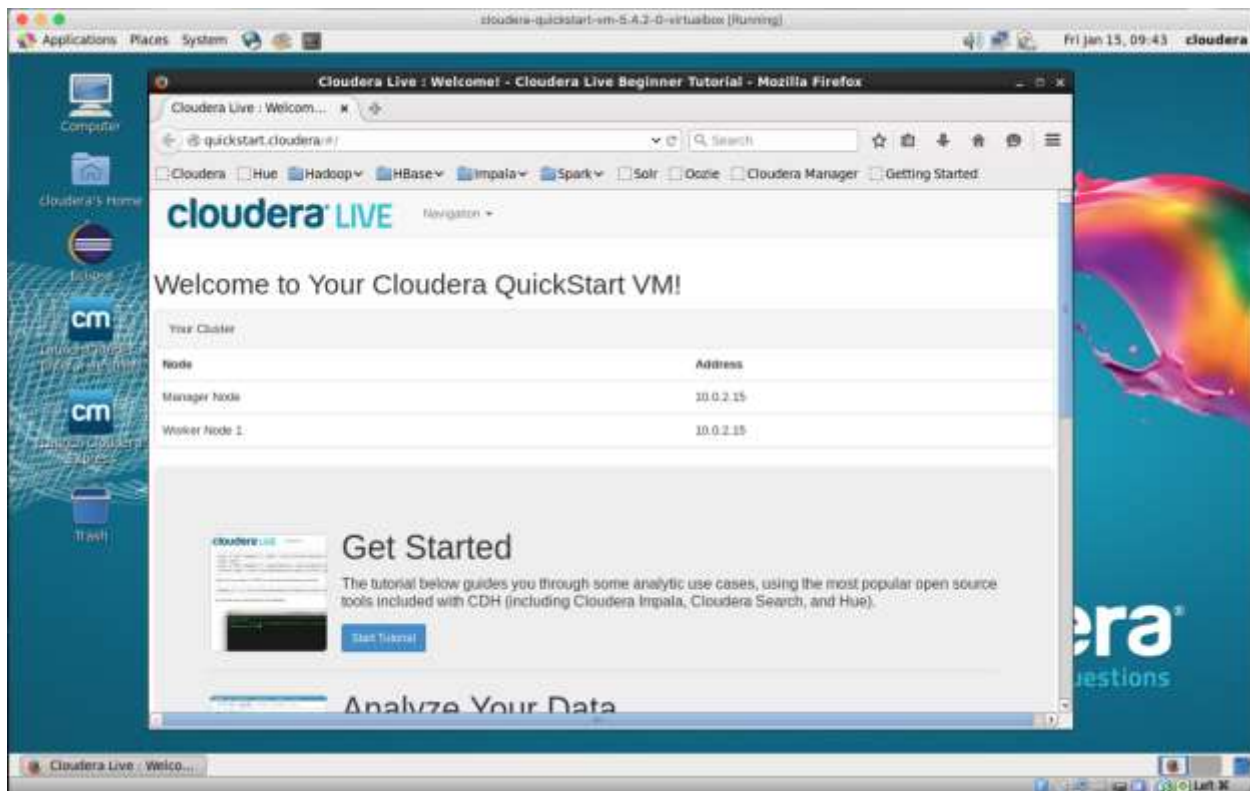
11. **Launch Cloudera VM.** When the importing is finished, the quickstart-vm-5.4.2-0 VM will appear on the left in the VirtualBox window. Select it and click the Start button to launch the VM.



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



13. **The Cloudera VM desktop.** Once the booting process is complete, the desktop will appear with a browser.



b) MultipleNodes:

Install a Multi Node Hadoop Cluster on Ubuntu 16.04

This article is about multi-node installation of Hadoop cluster. You would need minimum of 2 ubuntu machines or virtual images to complete a multi-node installation. If you want to just try out a single node cluster, follow this article on [Installing Hadoop on Ubuntu 16.04](#).

I used Hadoop Stable version 2.6.0 for this article. I did this setup on a 3 node cluster. For simplicity, i will designate one node as **master**, and 2 nodes as **slaves (slave-1, and slave-2)**. Make sure all slave nodes are reachable from master node. To avoid any unreachable hosts error, make sure you add the slave hostnames and ip addresses in **/etc/hosts** file. Similarly, slave nodes should be able to resolve **master** hostname.

Installing Java on Master and Slaves

```
$sudo add-apt-repository ppa:webupd8team/java  
$sudo apt-get update  
$sudo apt-get install oracle-java7-installer  
# Update Java runtime  
$sudo update-java-alternatives -s java-7-oracle
```

Disable IPv6

As of now Hadoop does not support IPv6, and is tested to work only on IPv4 networks. If you are using IPv6, you need to switch Hadoop host machines to use IPv4. [The Hadoop Wiki](#) link provides a one liner command to disable the IPv6. If you are not using IPv6, skip this step:

```
sudo sed -i 's/net.ipv6.conf\ == 1/net.ipv6.conf\ == 0/'  
/etc/sysctl.d/bindv6only.conf && sudo invoke-rc.d procps restart
```

Setting up a Hadoop User

Hadoop talks to other nodes in the cluster using no-password ssh. By having Hadoop run under a specific user context, it will be easy to distribute the ssh keys around in the Hadoop cluster. Let's create a user **hadoopuser** on **master** as well as **slave** nodes.

```
# Create hadoopgroup  
$sudo addgroup hadoopgroup  
# Create hadoopuser user  
$sudo adduser --ingroup hadoopgroup hadoopuser
```

Our next step will be to generate a ssh key for password-less login between master and slave nodes. Run the following commands only on **master** node. Run the last two commands for each slave node. Password less ssh should be working before you can proceed with further steps.

```
# Login as hadoopuser
$ su - hadoopuser

#Generate a ssh key for the user
$ ssh-keygen -t rsa -P ""

#Authorize the key to enable password less ssh
$ cat /home/hadoopuser/.ssh/id_rsa.pub >> /home/hadoopuser/.ssh/authorized_keys
$ chmod 600 authorized_keys

#Copy this key to slave-1 to enable password less ssh
$ ssh-copy-id -i ~/.ssh/id_rsa.pub slave-1

#Make sure you can do a password less ssh using following command.
$ ssh slave-1
```

Download and Install Hadoop binaries on Master and Slave nodes

Pick the best mirror site to download the binaries from [Apache Hadoop](http://www.apache.org/hadoop/), and download the stable/hadoop-2.6.0.tar.gz for your installation. **Do this step on master and every slave node.** You can download the file once and then distribute to each slave node using scp command.

```
$ cd /home/hadoopuser
$ wget http://www.webhostingjams.com/mirror/apache/hadoop/core/stable/hadoop-2.2.0.tar.gz
$ tar xvf hadoop-2.2.0.tar.gz
$ mv hadoop-2.2.0 hadoop
```

Setup Hadoop Environment on Master and Slave Nodes

Copy and paste following lines into your .bashrc file under /home/hadoopuser. **Do this step on master and every slave node.**

```
# Set HADOOP_HOME
```



```
export HADOOP_HOME=/home/hduser/hadoop
# Set JAVA_HOME
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
# Add Hadoop bin and sbin directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

Update hadoop-env.sh on Master and Slave Nodes

Update JAVA_HOME in /home/hadoopuser/hadoop/etc/hadoop/hadoop_env.sh to following. **Do this step on master and every slave node.**

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
```

Common Terminologies

Before we start getting into configuration details, let's discuss some of the basic terminologies used in Hadoop.

- **Hadoop Distributed File System:** A distributed file system that provides high-throughput access to application data. A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data. If you compare HDFS to a traditional storage structures (e.g. FAT, NTFS), then NameNode is analogous to a Directory Node structure, and DataNode is analogous to actual file storage blocks.
- **Hadoop YARN:** A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce:** A YARN-based system for parallel processing of large data sets.

Update Configuration Files

Add/update core-site.xml on **Master and Slave nodes** with following options. Master and slave nodes should all be using the same value for this property **fs.defaultFS**, and should be pointing to master node only.

```
/home/hadoopuser/hadoop/etc/hadoop/core-site.xml (Other Options)
```

```
<property>
<name>hadoop.tmp.dir</name>
```

```

<value>/home/hadoopuser/tmp</value>

<description>Temporary Directory.</description>

</property>

<property>

<name>fs.defaultFS</name>

<value>hdfs://master:54310</value>

<description>Use HDFS as file storage engine</description>

</property>

```

Add/update mapred-site.xml on **Master node only** with following options.

```

/home/hadoopuser/hadoop/etc/hadoop/mapred-site.xml (Other Options)

```

```

<property>

<name>mapreduce.jobtracker.address</name>

<value>master:54311</value>

<description>The host and port that the MapReduce job tracker runs
at. If “local”, then jobs are run in-process as a single map
and reduce task.

</description>

</property>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

<description>The framework for running mapreduce jobs</description>

```

```
</property>
```

Add/update hdfs-site.xml on **Master and Slave Nodes**. We will be adding following three entries to the file.

- **dfs.replication**– Here I am using a replication factor of 2. That means for every file stored in HDFS, there will be one redundant replication of that file on some other node in the cluster.
- **dfs.namenode.name.dir** – This directory is used by Namenode to store its metadata file. Here i manually created this directory /hadoop-data/hadoopuser/hdfs/namenode on master and slave node, and use the directory location for this configuration.
- **dfs.datanode.data.dir** – This directory is used by Datanode to store hdfs data blocks. Here i manually created this directory /hadoop-data/hadoopuser/hdfs/datanode on master and slave node, and use the directory location for this configuration.

/home/hadoopuser/hadoop/etc/hadoop/hdfs-site.xml (Other Options)

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>2</value>
```

```
<description>Default block replication.
```

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

```
</description>
```

```
</property>
```

```
<property>
```

```
<name>dfs.namenode.name.dir</name>
```

```
<value>/hadoop-data/hadoopuser/hdfs/namenode</value>
```

`<description>`Determines where on the local filesystem the DFS name node should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.

`</description>`

`</property>`

`<property>`

`<name>dfs.datanode.data.dir</name>`

`<value>/hadoop-data/hadoopuser/hdfs/datanode</value>`

`<description>`Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored.

`</description>`

`</property>`

Add yarn-site.xml on **Master and Slave Nodes**. This file is required for a Node to work as a Yarn Node. Master and slave nodes should all be using the same value for the following properties, and should be pointing to master node only.

```
/home/hadoopuser/hadoop/etc/hadoop/yarn-site.xml
```

`<property>`

`<name>yarn.nodemanager.aux-services</name>`

`<value>mapreduce_shuffle</value>`

`</property>`

`<property>`

`<name>yarn.resourcemanager.scheduler.address</name>`

`<value>master:8030</value>`

```
</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>master:8032</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>master:8088</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>master:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>master:8033</value>

</property>
```

Add/update **slaves** file on Master node only. Add just name, or ip addresses of master and all slave node. If file has an entry for localhost, you can remove that. This file is just helper file that are used by hadoop scripts to start appropriate services on master and slave nodes.

```
/home/hadoopuser/hadoop/etc/hadoop/slave
```

```
master
```

```
slave-1
```

```
slave-2
```

Format the Namenode

Before starting the cluster, we need to format the Namenode. Use the following command only on **master node**:

```
$ hdfs namenode -format
```

Start the Distributed Format System

Run the following on **master node** command to start the DFS.

```
$/home/hadoopuser/hadoop/sbin/start-dfs.sh
```

You should observe the output to ascertain that it tries to start datanode on slave nodes one by one. To validate the success, run following command on master nodes, and slave node.

```
$ su - hadoopuser
```

```
$ jps
```

The output of this command should list *NameNode*, *SecondaryNameNode*, *DataNode* on **master** node, and *DataNode* on all slave nodes. If you don't see the expected output, review the log files listed in Troubleshooting section.

Start the Yarn MapReduce Job tracker

Run the following command to start the Yarn mapreduce framework.

```
$/home/hadoopuser/hadoop/sbin/start-yarn.sh
```

To validate the success, run **jps** command again on master nodes, and slave node. The output of this command should list *NodeManager*, *ResourceManager* on **master** node, and *NodeManager*, on all **slave** nodes. If you don't see the expected output, review the log files listed in Troubleshooting section.

Review Yarn Web console

If all the services started successfully on all nodes, then you should see all of your nodes listed under Yarn nodes. You can hit the following url on your browser and verify that:

```
http://master:8088/cluster/nodes
```

Lets's execute a MapReduce example now

You should be all set to run a MapReduce example now. Run the following command

```
$hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar pi 30100
```

Once the job is submitted you can validate that its running on the cluster by accessing following url.

```
http://master:8088/cluster/apps
```

Troubleshooting

Hadoop uses \$HADOOP_HOME/logs directory. In case you get into any issues with your installation, that should be the first point to look at. In case, you need help with anything else, do leave me a comment.

REFERENCE BOOK:

<http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html>

CONCLUSION:

Thus we have successfully installed and tested single and multimode cluster.

Assignment: 2

AIM: Design a distributed application using MapReduce.

PROBLEM STATEMENT /DEFINITION

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.

OBJECTIVE:

- To understand the concept of Map Reduce.
- To understand the details of Hadoop File system
- To understand the technique for log file processing
- Analyze the performance of hadoop file system
- To understand use of distributed processing

THEORY:

What is MapReduce?

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

- Generally MapReduce paradigm is based on sending the computer to where the data resides!

- 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.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

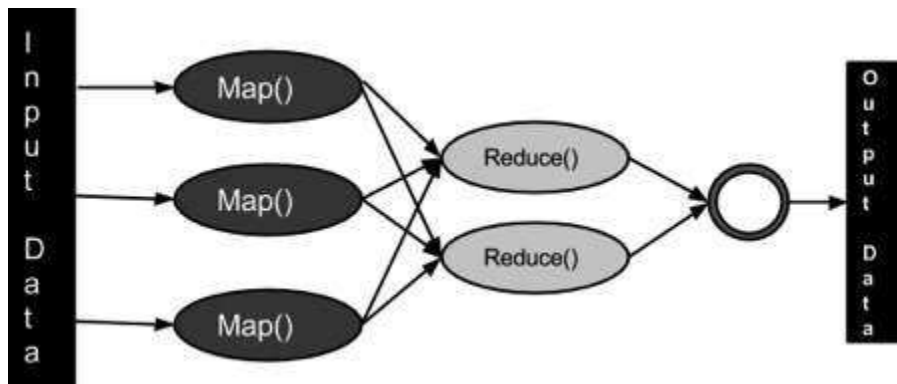


Fig.1. Map Reduce job processing

Inputs and Outputs (Java Perspective)

The MapReduce framework operates on $\langle \text{key}, \text{value} \rangle$ pairs, that is, the framework views the input to the job as a set of $\langle \text{key}, \text{value} \rangle$ pairs and produces a set of $\langle \text{key}, \text{value} \rangle$ 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) $\langle k1, v1 \rangle \rightarrow \text{map} \rightarrow \langle k2, v2 \rangle \rightarrow \text{reduce} \rightarrow \langle k3, v3 \rangle$ (Output).

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Terminology

- **PayLoad** - Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** - Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** - Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** - Node where data is presented in advance before any processing takes place.
- **MasterNode** - Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** - Node where Map and Reduce program runs.
- **JobTracker** - Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** - Tracks the task and reports status to JobTracker.
- **Job** - A program is an execution of a Mapper and Reducer across a dataset.
- **Task** - An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** - A particular instance of an attempt to execute a task on a SlaveNode.

Example Scenario

Given below is the data regarding the electrical consumption of an organization. It contains the monthly electrical consumption and the annual average for various years.

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29

1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

If the above data is given as input, we have to write applications to process it and produce results such as finding the year of maximum usage, year of minimum usage, and so on. This is a walkover for the programmers with finite number of records. They will simply write the logic to produce the required output, and pass the data to the application written.

But, think of the data representing the electrical consumption of all the largescale industries of a particular state, since its formation.

When we write applications to process such bulk data,

- They will take a lot of time to execute.
- There will be a heavy network traffic when we move data from source to network server and so on.

To solve these problems, we have the MapReduce framework.

Input Data

The above data is saved as **sample.txt** and given as input. The input file looks as shown below.

```
197923232432425262626252625
198026272828283031313130303029
1981313232323334353636343434
198439383939394142434039383840
198538393939394141410040393945
```

Example Program

Given below is the program to the sample data using MapReduce framework.

```
package hadoop;
```

```

import java.util.*;

import java.io.IOException;
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ProcessUnits
{
    //Mapper class

    public static class E_EMapper extends MapReduceBase implements
        Mapper<LongWritable, /*Input key Type */
            Text, /*Input value Type*/
            Text, /*Output key Type*/
            IntWritable> /*Output value Type*/
    {

        //Map function

        public void map(LongWritable key, Text value,

```

```

OutputCollector<Text,IntWritable> output,

Reporter reporter)throwsIOException

{

String line =value.toString();

String lasttoken =null;

StringTokenizer s =newStringTokenizer(line,"t");

String year =s.nextToken();


while(s.hasMoreTokens())

{

lasttoken=s.nextToken();

}


int avgprice =Integer.parseInt(lasttoken);

output.collect(newText(year),newIntWritable(avgprice));

}

}


//Reducer class

publicstaticclass E_EReduce extendsMapReduceBaseimplements

Reducer<Text,IntWritable,Text,IntWritable>

{

```

```

//Reduce function

public void reduce(Text key, Iterator<IntWritable> values,
    OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException
{
    int maxavg=30;
    int val=Integer.MIN_VALUE;

    while(values.hasNext())
    {
        if((val=values.next().get())>maxavg)
        {
            output.collect(key, new IntWritable(val));
        }
    }

}

}

//Main function

public static void main(String args[]) throws Exception
{
    JobConf conf = new JobConf(ProcessUnits.class);

```

```

conf.setJobName("max_electricityunits");

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

conf.setMapperClass(E_EMapper.class);

conf.setCombinerClass(E_EReduce.class);

conf.setReducerClass(E_EReduce.class);

conf.setInputFormat(TextInputFormat.class);

conf.setOutputFormat(TextOutputFormat.class);


FileInputFormat.setInputPaths(conf,newPath(args[0]));

FileOutputFormat.setOutputPath(conf,newPath(args[1]));


JobClient.runJob(conf);

}

}

```

Save the above program as **ProcessUnits.java**. The compilation and execution of the program is explained below.

Compilation and Execution of Process Units Program

Let us assume we are in the home directory of a Hadoop user (e.g. /home/hadoop).

Follow the steps given below to compile and execute the above program.

Step 1

The following command is to create a directory to store the compiled java classes.

```
$ mkdir units
```

Step 2

Download **Hadoop-core-1.2.1.jar**, which is used to compile and execute the MapReduce program. Visit the following link <http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1> to download the jar. Let us assume the downloaded folder is **/home/hadoop/**.

Step 3

The following commands are used for compiling the **ProcessUnits.java** program and creating a jar for the program.

```
$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java
$ jar -cvf units.jar -C units/.
```

Step 4

The following command is used to create an input directory in HDFS.

```
$HADOOP_HOME/bin/hadoop fs -mkdir input_dir
```

Step 5

The following command is used to copy the input file named **sample.txt** in the input directory of HDFS.

```
$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir
```

Step 6

The following command is used to verify the files in the input directory.

```
$HADOOP_HOME/bin/hadoop fs -ls input_dir/
```

Step 7

The following command is used to run the Eleunit_max application by taking the input files from the input directory.

```
$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir output_dir
```

Wait for a while until the file is executed. After execution, as shown below, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc.

INFO mapreduce.**Job:Job** job_1414748220717_0002

completed successfully

14/10/3106:02:52

INFO mapreduce.**Job:Counters:49**

FileSystemCounters

FILE:**Number** of bytes read=**61**

FILE:**Number** of bytes written=**279400**

FILE:**Number** of read operations=**0**

FILE:**Number** of large read operations=**0**

FILE:**Number** of write operations=**0**

HDFS:**Number** of bytes read=**546**

HDFS:**Number** of bytes written=**40**

HDFS:**Number** of read operations=**9**

HDFS:**Number** of large read operations=**0**

HDFS:**Number** of write operations=**2JobCounters**

Launched map tasks=**2**

Launched reduce tasks=**1**

Data-local map tasks=**2**

Total time spent **by** all maps **in** occupied slots (ms)=**146137**

Total time spent **by** all reduces **in** occupied slots (ms)=**441**

Total time spent **by** all map tasks (ms)=**14613**

Total time spent by all reduce tasks (ms)=44120

Total vcore-seconds taken by all map tasks=146137

Total vcore-seconds taken by all reduce tasks=44120

Total megabyte-seconds taken by all map tasks=149644288

Total megabyte-seconds taken by all reduce tasks=45178880

Map-ReduceFramework

Map input records=5

Map output records=5

Map output bytes=45

Map output materialized bytes=67

Input split bytes=208

Combine input records=5

Combine output records=5

Reduce input groups=5

Reduce shuffle bytes=6

Reduce input records=5

Reduce output records=5

SpilledRecords=10

ShuffledMaps=2

FailedShuffles=0

MergedMap outputs=2

GC time elapsed (ms)=948

CPU time spent (ms)=5160

Physical memory (bytes) snapshot=47749120
Virtual memory (bytes) snapshot=2899349504
Total committed heap usage (bytes)=277684224

FileOutputFormatCounters
BytesWritten=40

Step 8

The following command is used to verify the resultant files in the output folder.

```
$HADOOP_HOME/bin/hadoop fs -ls output_dir/
```

Step 9

The following command is used to see the output in **Part-00000** file. This file is generated by HDFS.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000
```

Below is the output generated by the MapReduce program.

```
198134  
198440  
198545
```

Step 10

The following command is used to copy the output folder from HDFS to the local file system for analyzing.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs get output_dir /home/hadoop
```

2. MapReduce (Log File)

Use Hadoop to Analyze Java Logs (Tomcat catalina.out)

One of the Java applications I develop deploys in Tomcat and is load-balanced across a couple dozen servers. Each server can produce gigabytes of log output daily due to the high volume. This post demonstrates simple use of [hadoop](#) to quickly extract useful and relevant information from catalina.out files using Map Reduce. I followed [Hadoop: The Definitive Guide](#) for setup and example code.

Get some seed data

Now that Hadoop is all setup, I need some seed data to operate on. For this I just reached out and grabbed a log file from one of my production servers.

```
[watrous@myhost ~]$ mkdir input  
[watrous@myhost ~]$ scp watrous@mywebhost.com:/var/lib/tomcat/logs/catalina.out ./input/
```

Creating Map Reduce Classes

The most simple operation in Hadoop requires a Mapper class, a Reducer class and a third class that identifies the Mapper and Reducer including the datatypes that connect them. The examples below required two jars from the release downloaded above:

- `hadoop-2.2.0.tar.gz\hadoop-2.2.0.tar\hadoop-2.2.0\share\hadoop\common\hadoop-common-2.2.0.jar`
- `hadoop-2.2.0.tar.gz\hadoop-2.2.0.tar\hadoop-2.2.0\share\hadoop\mapreduce\hadoop-mapreduce-client-core-2.2.0.jar`

I also use [regular expressions in Java](#) to analyze each line in the log. Regular expressions can be more resilient to variations and allow for grouping, which gives easy access to specific data elements. As always, I used [Kodos to develop the regular expression](#).

In the example below, I don't actually use the log value, but instead I just count up how many occurrences there are by key.

Mapper class

```

import java.io.IOException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TomcatLogErrorMapper extends Mapper<LongWritable, Text, Text, Text>{

    String pattern = "([0-9]{4}-[0-9]{2}-[0-9]{2})\\s([0-9]{2}:[0-9]{2}:[0-9]{2}).[0-9]{3})\\s*([a-zA-Z]+)\\s*([a-zA-Z.]+)\\s*-[\\s*](.+)S";
    // Create a Pattern object
    Pattern r = Pattern.compile(pattern);

    @Override
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();

        Matcher m = r.matcher(line);
        if(m.find()){
            // only consider ERRORS for this example
            if(m.group(2).contains("ERROR")){
                // example log line
                // 2013-11-08 04:06:56,586 DEBUG component.helpers.GenericSOAPConnector - Attempting to connect to: https://remotehost.com/app/rfc/entry/msg_status
                //      System.out.println("Found value: " + m.group(0)); //complete line
                //      System.out.println("Found value: " + m.group(1)); // date
                //      System.out.println("Found value: " + m.group(2)); // log level
                //      System.out.println("Found value: " + m.group(3)); // class
                //      System.out.println("Found value: " + m.group(4)); // message
                context.write(new Text(m.group(1)), new Text(m.group(2)+m.group(3)+m.group(4)));
            }
        }
    }
}

```

Reducer class

```

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class TomcatLogErrorReducer extends Reducer<Text, Text, Text, IntWritable>{

    @Override
    public void reduce(Text key, Iterable<Text> values, Context context)
        throws IOException, InterruptedException {
        int countValue = 0;
        for(Text value : values){

```

```

        countValue++;
    }
    context.write(key, new IntWritable(countValue));
}
}

```

Job class with main

```

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TomcatLogError {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: TomcatLogError <input path> <output path>");
            System.exit(-1);
        }

        Job job = new Job();
        job.setJarByClass(TomcatLogError.class);
        job.setJobName("Tomcat Log Error");
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setMapperClass(TomcatLogErrorMapper.class);
        job.setReducerClass(TomcatLogErrorReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

Running Hadoop

In netbeans I made sure that the Main Class was TomcatLogError in the compiled jar. I then ran Clean and Build to get a jar which I transferred up to the server where I installed Hadoop.

```

[watrous@myhost ~]$ hadoop jar HadoopExample.jar input/catalina.out ~/output
13/11/1119:20:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
13/11/1119:20:52 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
...
13/11/1118:36:57 INFO mapreduce.Job: Job job_local1725513594_0001 completed successfully
13/11/1118:36:57 INFO mapreduce.Job: Counters: 27
    File System Counters
        FILE: Number of bytes read=430339145

```

```
FILE: Number of bytes written=1057396
FILE: Number of readoperations=0
FILE: Number of large readoperations=0
FILE: Number of writeoperations=0
Map-Reduce Framework
Map input records=1101516
Map output records=105
Map output bytes=20648
Map output materialized bytes=20968
Input splitbytes=396
Combine input records=0
Combine output records=0
Reduce input groups=23
Reduce shuffle bytes=0
Reduce input records=105
Reduce output records=23
Spilled Records=210
Shuffled Maps =0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=234
CPU time spent (ms)=0
Physical memory (bytes)snapshot=0
Virtual memory (bytes)snapshot=0
Total committed heap usage (bytes)=1827143680
File Input Format Counters
  Bytes Read=114455257
File Output Format Counters
  Bytes Written=844
```

The output folder now contains a file named **part-r-00000** with the results of the processing.

```
[watrous@c0003913 ~]$ more output/part-r-00000
2013-11-08 04:04:51,8942
2013-11-08 05:04:52,7112
2013-11-08 05:33:23,073 3
2013-11-08 06:04:53,6892
2013-11-08 07:04:54,3663
2013-11-08 08:04:55,096 2
2013-11-08 13:34:28,9362
2013-11-08 17:32:31,6293
2013-11-08 18:51:17,3571
2013-11-08 18:51:17,4231
2013-11-08 18:51:17,4911
2013-11-08 18:51:17,4991
2013-11-08 18:51:17,5001
```

```
2013-11-08 18:51:17,5021
2013-11-08 18:51:17,5031
2013-11-08 18:51:17,5041
2013-11-08 18:51:17,5061
2013-11-08 18:51:17,6516
2013-11-08 18:51:17,65223
2013-11-08 18:51:17,65325
2013-11-08 18:51:17,65419
2013-11-08 19:01:13,7712
2013-11-08 21:32:34,5222
```

Based on this analysis, there were a number of errors produced around the hour **18:51:17**. It is then easy to change the Mapper class to emit based on a different key, such as Class or Message to identify more precisely what the error is, now that I know when the errors happened.

References:

<https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

CONCLUSION:

Understand the uses of distributed data processing using Map reduce.

Assignment: 3

AIM: Write an application using HiveQL for flight information system

PROBLEM STATEMENT /DEFINITION

Write an application using HiveQL for flight information system which will include

- a. Creating, Dropping, and altering Database tables.
- b. Creating an external Hive table.
- c. Load table with data, insert new values and field in the table, Join tables with Hive
- d. Create index on Flight Information Table
- e. Find the average departure delay per day in 2008.

OBJECTIVE

- To understand various NOSQL database
- To understand the integration of NOSQL database with Hadoop.
- To analyze the performance of distributed processing with NOSQL.

THEORY:

- Hive and HBase Architecture
- Explanation of Hive Architecture
- HBase Architecture
- Explanation of HBase Architecture
- List and details of DDL and DML Commands in HBase and Hive

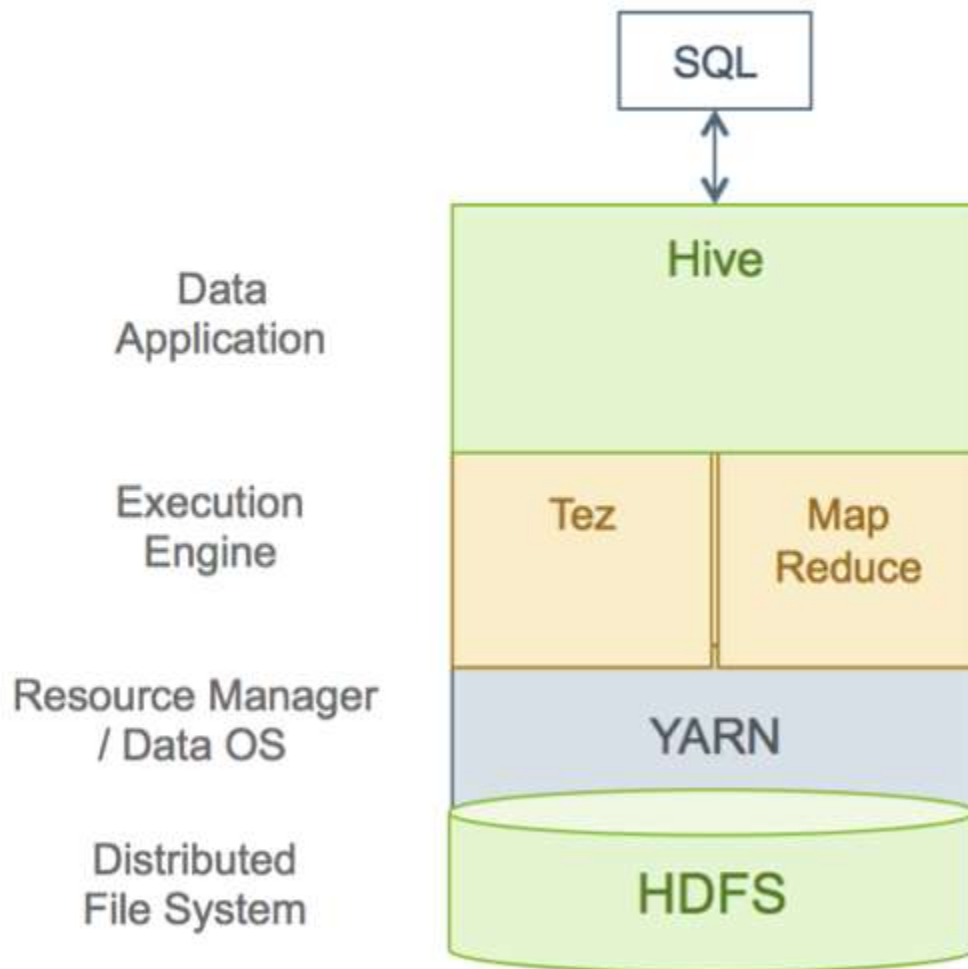
SQL queries are submitted to Hive and they are executed as follows:

1. Hive compiles the query.
2. An execution engine, such as Tez or MapReduce, executes the compiled query.
3. The resource manager, YARN, allocates resources for applications across the cluster.

4. The data that the query acts upon resides in HDFS (Hadoop Distributed File System). Supported data formats are ORC, AVRO, Parquet, and text.
5. Query results are then returned over a JDBC/ODBC connection.

A simplified view of this process is shown in the following figure.

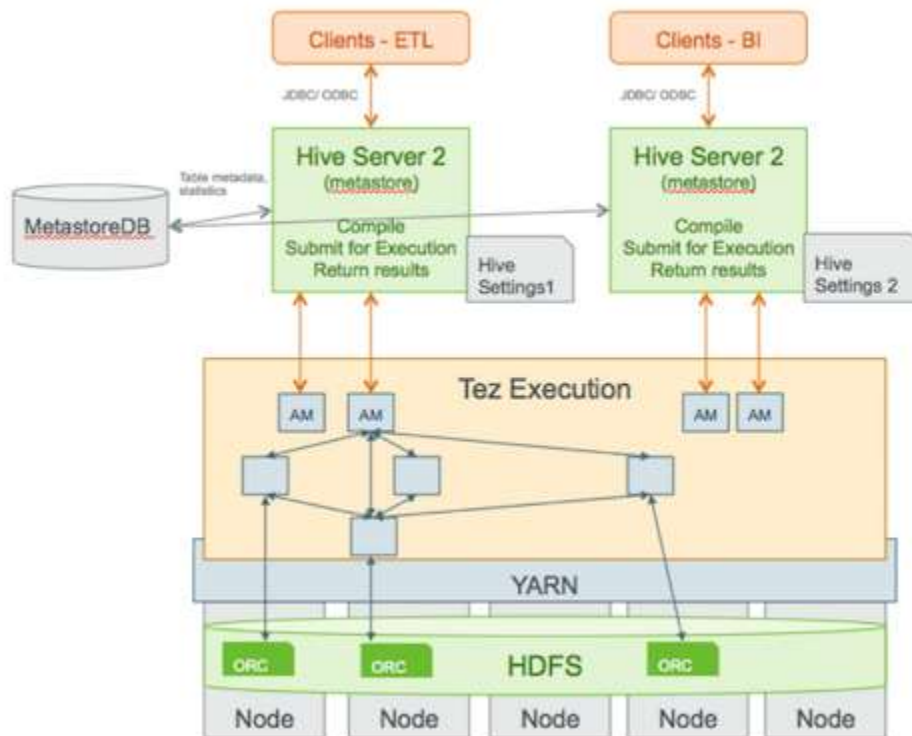
Figure 1.1. SQL Query Execution Process



Detailed Query Execution Architecture

The following diagram shows a detailed view of the HDP query execution architecture:

Figure 1.2. SQL Query Execution Process



The following sections explain major parts of the query execution architecture.

Hive Clients

You can connect to Hive using a JDBC/ODBC driver with a BI tool, such as Microstrategy, Tableau, BusinessObjects, and others, or from another type of application that can access Hive over a JDBC/ODBC connection. In addition, you can also use a command-line tool, such as Beeline, that uses JDBC to connect to Hive. The Hive command-line interface (CLI) can also be used, but it has been

deprecated in the current release and Hortonworks does not recommend that you use it for security reasons.

SQL in Hive

Hive supports a large number of standard SQL dialects. In a future release, when SQL:2011 is adopted, Hive will support ANSI-standard SQL.

HiveServer2

Clients communicate with HiveServer2 over a JDBC/ODBC connection, which can handle multiple user sessions, each with a different thread. HiveServer2 can also handle long-running sessions with asynchronous threads. An embedded metastore, which is different from the MetastoreDB, also runs in HiveServer2. This metastore performs the following tasks:

- Get statistics and schema from the MetastoreDB
- Compile queries
- Generate query execution plans
- Submit query execution plans
- Return query results to the client

Multiple HiveServer2 Instances for Different Workloads

Multiple HiveServer2 instances can be used for:

- Load-balancing and high availability using ZooKeeper
- Running multiple applications with different settings

Because HiveServer2 uses its own settings file, using one for ETL operations and another for interactive queries is a common practice. All HiveServer2 instances can share the same MetastoreDB. Consequently, setting up multiple HiveServer2 instances that have embedded metastores is a simple operation.

Tez Execution

After query compilation, HiveServer2 generates a Tez graph that is submitted to YARN. A Tez Application Master (AM) monitors the query while it is running.

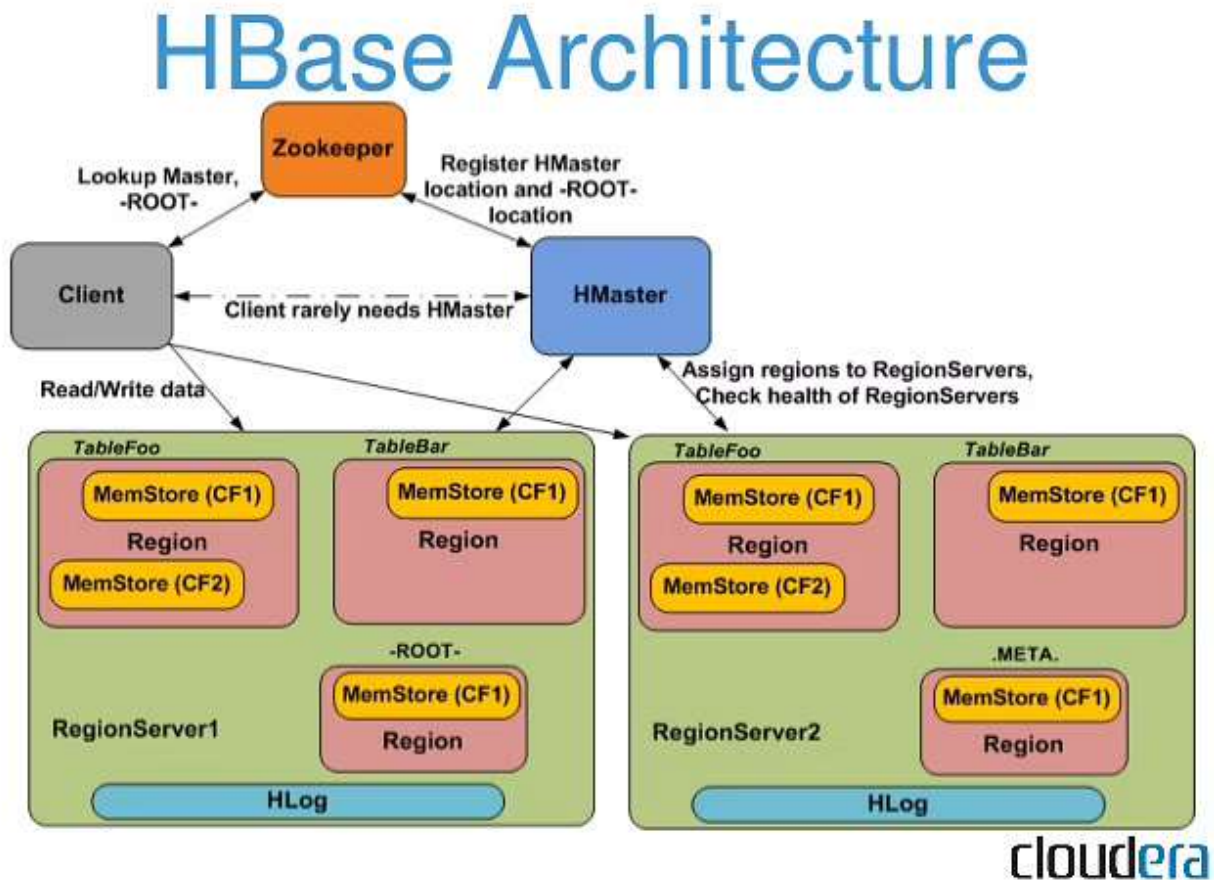
Security

HiveServer2 performs standard SQL security checks when a query is submitted, including connection authentication. After the connection authentication check, the server runs authorization checks to make sure that the user who submits the query has permission to access the databases, tables, columns, views, and other resources required by the query. Hortonworks recommends that you use SQLStdAuth or Ranger

to implement security. Storage-based access controls, which is suitable for ETL workloads only, is also available.

File Formats

Hive supports many file formats. You can write your own SerDes (Serializers, Deserializers) interface to support new file formats.



Components of Apache HBase Architecture

HBase architecture has 3 important components- HMaster, Region Server and ZooKeeper.

i. HMaster

HBase HMaster is a lightweight process that assigns regions to region servers in the Hadoop cluster for load balancing. Responsibilities of HMaster –

- Manages and Monitors the Hadoop Cluster
- Performs Administration (Interface for creating, updating and deleting tables.)
- Controlling the failover
- DDL operations are handled by the HMaster
- Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.
- Region Server

These are the worker nodes which handle read, write, update, and delete requests from clients. Region Server process, runs on every node in the hadoop cluster. Region Server runs on HDFS DataNode and consists of the following components –

- Block Cache – This is the read cache. Most frequently read data is stored in the read cache and whenever the block cache is full, recently used data is evicted.
- MemStore- This is the write cache and stores new data that is not yet written to the disk. Every column family in a region has a MemStore.
- Write Ahead Log (WAL) is a file that stores new data that is not persisted to permanent storage.
- HFile is the actual storage file that stores the rows as sorted key values on a disk.
- i. Zookeeper

HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization. Whenever a client wants to communicate with regions, they have to approach Zookeeper first. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquorum will trigger error messages and start repairing failed nodes.

Build an Awesome Job Winning Project Portfolio with Solved [End-to-End Big Data Projects](#)

ZooKeeper service keeps track of all the region servers that are there in an HBase cluster- tracking information about how many region servers are there and which region servers are holding which DataNode. HMaster contacts ZooKeeper to get the details of region servers. Various services that Zookeeper provides include –

- Establishing client communication with region servers.
- Tracking server failure and network partitions.

- Maintain Configuration Information
- Provides ephemeral nodes, which represent different region servers.

Implementation

a) Creating, Dropping, and altering Database tables Using 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'

hbase(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=>'info:source'
```

COLUMN CELL

info: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=>['info:source','info:dest']
```

COLUMN CELL

info:dest timestamp=1521312730758, value=mumbai

info:source timestamp=1521312493881, value=pune

2 row(s) in 0.0190 seconds

```
hbase(main):055:0> scan 'flight',COLUMNS=>'info:source'
```

ROW COLUMN+CELL

1 column=info:source, timestamp=1521312493881, value=pune

2 column=info:source, timestamp=1521313092772, value=pune

3 column=info:source, timestamp=1521313290906, value=mumbai

4 column=info:source, timestamp=1521313404831, value=mumbai

4 row(s) in 0.0320 seconds

b) Creating an external Hive table to connect to the HBase for

Customer Information Table

Covers====>

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

Hive

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 terminated

by ',' 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 kolkata 7.00a.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-4c463834731a

Total jobs = 1

Launching Job 1 out of 1

Number of reduce tasks not specified. Estimated from input data size: 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_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 sec

MapReduce 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 msec

OK

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)

REFERENCES:

<https://cwiki.apache.org/confluence/display/Hive/Tutorial>

<https://cwiki.apache.org/confluence/display/Hive/GettingStarted>

https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.5.0/bk_hive-performance-tuning/content/ch_hive_architectural_overview.html

<https://hbase.apache.org/book.html>

CONCLUSION:

Understand to use various aggregate functions using map reduce.

PartB

Assignments based on Data Analytics using Python

Assignment: 4

AIM: Perform the different operations using Python on the Facebook metrics data set.

PROBLEM STATEMENT /DEFINITION

Perform the following operations using R/Python on the Amazon book review and facebook metrics data sets

- a. Create data subsets
- b. Merge Data
- c. Sort Data
- d. Transposing Data
- e. Melting Data to long format

Casting data to wide format

OBJECTIVE:

To learn Python programming

To learn different data preprocessing techniques.

THEORY:

a. Create data subsets

1. Create a subset of a Python dataframe using the loc() function

[Python loc\(\) function](#) enables us to form a subset of a data frame according to a specific row or column or a combination of both.

The loc() function works on the basis of labels i.e. we need to provide it with the label of the row/column to choose and create the customized subset.

Syntax: `pandas.dataframe.loc[]`

2. Using Python iloc() function to create a subset of a dataframe

[Python iloc\(\) function](#) enables us to create subset choosing specific values from rows and columns based on indexes. That is, unlike loc() function which works on labels, iloc() function works on

index values. We can choose and create a subset of a Python dataframe from the data providing the index numbers of the rows and columns.

Syntax: `pandas.dataframe.iloc[]`

3. Indexing operator to create a subset of a dataframe

In a simple manner, we can make use of an indexing operator i.e. square brackets to create a subset of the data.

Syntax: `dataframe[['col1','col2','colN']]`

b. Merge Data

Pandas has full-featured, high performance in-memory join operations idiomatically very similar to relational databases like SQL.

Pandas provides a single function, merge, as the entry point for all standard database join operations between DataFrame objects –

```
pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None,
left_index=False, right_index=False, sort=True)
```

Here, we have used the following parameters –

- **left** – A DataFrame object.
- **right** – Another DataFrame object.
- **on** – Columns (names) to join on. Must be found in both the left and right DataFrame objects.
- **left_on** – Columns from the left DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.

- **right_on** – Columns from the right DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.
- **left_index** – If True, use the index (row labels) from the left DataFrame as its join key(s). In case of a DataFrame with a MultiIndex (hierarchical), the number of levels must match the number of join keys from the right DataFrame.
- **right_index** – Same usage as left_index for the right DataFrame.
- **how** – One of 'left', 'right', 'outer', 'inner'. Defaults to inner. Each method has been described below.
- **sort** – Sort the result DataFrame by the join keys in lexicographical order. Defaults to True, setting to False will improve the performance substantially in many cases.

c. Sort Data

`pandas.DataFrame.sort_values`

`DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last', ignore_index=False, key=None)[source]`

Sort by the values along either axis.

d. Transposing Data

`pandas.DataFrame.transpose`

`DataFrame.transpose(*args, copy=False)[source]`

Transpose index and columns.

Reflect the DataFrame over its main diagonal by writing rows as columns and vice-versa. The property **T** is an accessor to the method `transpose()`.

Parameters

**args* tuple, optional

Accepted for compatibility with NumPy.

copybool, default False

Whether to copy the data after transposing, even for DataFrames with a single dtype. Note that a copy is always required for mixed dtype DataFrames, or for DataFrames with any extension types.

Returns

DataFrame

The transposed DataFrame.

e. Melting Data to long format

pandas.melt

`pandas.melt(frame, id_vars=None, value_vars=None, var_name=None, value_name='value', col_level=None, ignore_index=True)`

Unpivot a DataFrame from wide to long format, optionally leaving identifiers set.

This function is useful to massage a DataFrame into a format where one or more columns are identifier variables (`id_vars`), while all other columns, considered measured variables (`value_vars`), are “unpivoted” to the row axis, leaving just two non-identifier columns, ‘variable’ and ‘value’.

Parameters

id_varstuple, list, or ndarray, optional

Column(s) to use as identifier variables.

value_varstuple, list, or ndarray, optional

Column(s) to unpivot. If not specified, uses all columns that are not set as `id_vars`.

var_namescalar

Name to use for the ‘variable’ column. If `None` it uses `frame.columns.name` or ‘variable’.

value_name*scalar, default 'value'*

Name to use for the 'value' column.

col_level*int or str, optional*

If columns are a MultiIndex then use this level to melt.

ignore_index*bool, default True*

If True, original index is ignored. If False, the original index is retained. Index labels will be repeated as necessary.

Returns

DataFrame

Unpivoted DataFrame.

pandas.DataFrame.pivot

DataFrame.pivot(*index=None, columns=None, values=None*)

Return reshaped DataFrame organized by given index / column values.

Reshape data (produce a “pivot” table) based on column values. Uses unique values from specified index / columns to form axes of the resulting DataFrame. This function does not support data aggregation, multiple values will result in a MultiIndex in the columns. See the [User Guide](#) for more on reshaping.

Parameters

index*str or object or a list of str, optional*

Column to use to make new frame's index. If None, uses existing index.

columns*str or object or a list of str*

Column to use to make new frame's columns.

values*str, object or a list of the previous, optional*

Column(s) to use for populating new frame's values. If not specified, all remaining columns will be used and the result will have hierarchically indexed columns.

Returns

DataFrame

Returns reshaped DataFrame.

Raises

ValueError:

When there are any index, columns combinations with multiple values.
DataFrame.pivot_table when you need to aggregate.

Implementation

```
import pandas as pd

from google.colab import files

uploaded = files.upload()

import io

df = pd.read_csv(io.BytesIO(uploaded['dataset_Facebook.csv']), sep = ';')

#print(df)

df.head(5)

df['Type']

video_sub=df['Type']

print(df.columns)

Bydf_subset = df[['like', 'share']]
```

```

print(Bydf_subset)

Hydf_subset1 = df[df['like']>100]

print(Hydf_subset1)

df.loc[1:7,['like','share']]

df.sort_values(by = "like",ascending= False)

df.sort_values("like")

df.sort_values(by = "like",ascending= False, kind="mergesort")

Byresult = df.transpose()

print(Byresult)

selective_df = pd.DataFrame(df,columns
=['like','share','Category','Type'])

selective_df.head(5)

pivot_table = pd.pivot_table(selective_df,index= ['Category','like'])

print(pivot_table)

```

REFERENCES:

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.pivot.html>

CONCLUSION:

Understand different data preprocessing techniques

Assignment: 5

AIM : To Perform Data Preprocessing and model building.

PROBLEM STATEMENT /DEFINITION

Perform the following operations using 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

OBJECTIVE

- To understand the concept of Data Preprocessing
- To understand the methods of Data preprocessing.

THEORY:

Data Preprocessing:

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following

- Accuracy: To check whether the data entered is correct or not.
- Completeness: To check whether the data is available or not recorded.
- Consistency: To check whether the same data is kept in all the places that do or do not match.
- Timeliness: The data should be updated correctly.
- Believability: The data should be trustable.
- Interpretability: The understandability of the data.

Major Tasks in Data Preprocessing:

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



Let's discuss each of these points in detail:

Before Data Preprocessing, The necessary libraries and dataset have to be imported.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
url='/content/drive/MyDrive/heart.csv'
df=pd.read_csv(url)
```

1.Data cleaning:

Data Cleaning means the process of identifying the incorrect, incomplete, inaccurate, irrelevant or missing part of the data and then modifying, replacing or deleting them according to the necessity. Data cleaning is considered a foundational element of basic data science.

Different Ways of Cleaning Data

Now let's take a closer look in the different ways of cleaning data.

Changing the Datatype of the columns:

The variables types are in heart dataset are:

- Binary: sex, fbs, exang, target
- Categorical: cp, restecg, slope, ca, thal
- Continuous: age, trestbps, chol, thalach, oldpeak

But, df.info() gives the following output.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   age         303 non-null   int64
 1   sex         303 non-null   int64
 2   cp          303 non-null   int64
 3   trestbps    303 non-null   int64
 4   chol        303 non-null   int64
 5   fbs         303 non-null   int64
 6   restecg     303 non-null   int64
 7   thalach     303 non-null   int64
 8   exang       303 non-null   int64
 9   oldpeak     303 non-null   float64
10   slope       303 non-null   int64
11   ca          303 non-null   int64
12   thal        303 non-null   int64
13   target      303 non-null   int64
```

```
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Hence, We will change the Datatypes of columns to appropriate types.

```
df['sex'] = df['sex'].astype('object')
df['cp'] = df['cp'].astype('object')
df['fbs'] = df['fbs'].astype('object')
df['restecg'] = df['restecg'].astype('object')
df['exang'] = df['exang'].astype('object')
df['slope'] = df['slope'].astype('object')
df['ca'] = df['ca'].astype('object')
df['thal'] = df['thal'].astype('object')
df.dtypes
```

Inconsistent column :

If your DataFrame contains columns that are irrelevant or you are never going to use them then you can drop them to give more focus on the columns you will work on. Let's see an example of how to deal with such a data set.

From the Air Quality dataset,

Dropping of less valued columns: stn_code, agency, sampling_date, location_monitoring_agency do not add much value to the dataset in terms of information. Therefore, we can drop those columns.

Changing the types to uniform format: When you see the dataset, you may notice that the 'type' column has values such as 'Industrial Area' and 'Industrial Areas' — both actually mean the same, so let's remove such type of stuff and make it uniform.

Creating a year column To view the trend over a period of time, we need year values for each row and also when you see in most of the values in date column only has 'year' value. So, let's create a new column holding year values.

1.stn_code, agency, sampling_date, location_monitoring_agency do not add much value to the dataset in terms of information. Therefore, we can drop those columns.

2.Dropping rows where no date is available.


```
df=df.drop(['stn_code',
'agency','sampling_date','location_monitoring_station'], axis = 1)
#dropping columns that aren't required

df=df.dropna(subset=['date']) # dropping rows where no date is available
```

Missing data:

It is rare to have a real world dataset without having any missing values. When you start to work with real world data, you will find that most of the dataset contains missing values. Handling missing values is very important because if you leave the missing values as it is, it may affect your analysis and machine learning models. So, you need to be sure that your dataset contains missing values or not. If you find missing values in your dataset you must handle it. If you find any missing values in the dataset you can perform any of these three task on it:

1. Leave as it is
2. Filling the missing values
3. Drop them

For filling the missing values we can perform different methods. For example, Figure 4 shows that airquality dataset has missing values.

The column such as SO₂, NO₂, rspm, spm, pm_{2.5} are the ones which contribute much to our analysis. So, we need to remove null from those columns to avoid inaccuracy in the prediction. We use the Imputer from sklearn.preprocessing to fill the missing values in every column with the mean.

```
# defining columns of importance, which shall be used reguarly
COLS = ['so2', 'no2', 'rspm', 'spm', 'pm2_5']
from sklearn.impute import SimpleImputer
# invoking SimpleImputer to fill missing values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
df[COLS] = imputer.fit_transform(df[COLS])
```

Outliers:

*“In statistics, an **outlier** is a data point that differs significantly from other observations.”*

That means an outlier indicates a data point that is significantly different from the other data points in the data set. Outliers can be created due to the errors in the experiments or the variability in the measurements. Let's look an example to clear the concept.

So, now the question is how can we detect the outliers in the dataset.

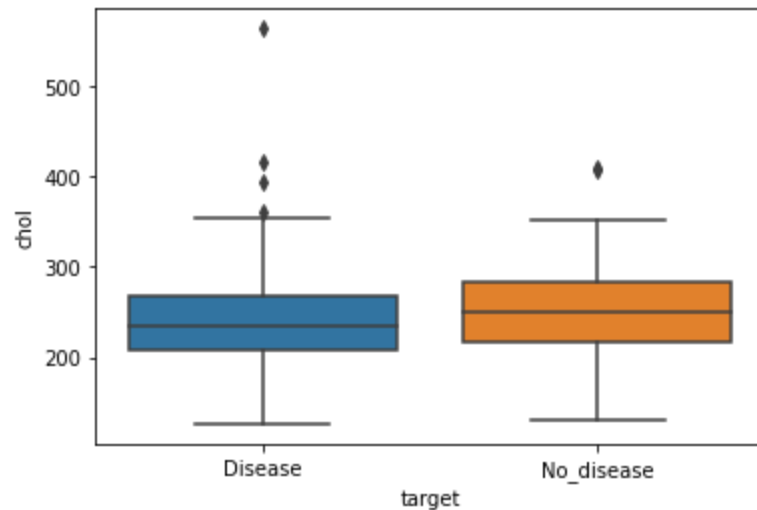
For detecting the outliers we can use :

1. Box Plot
2. Scatter plot
3. Z-score etc.

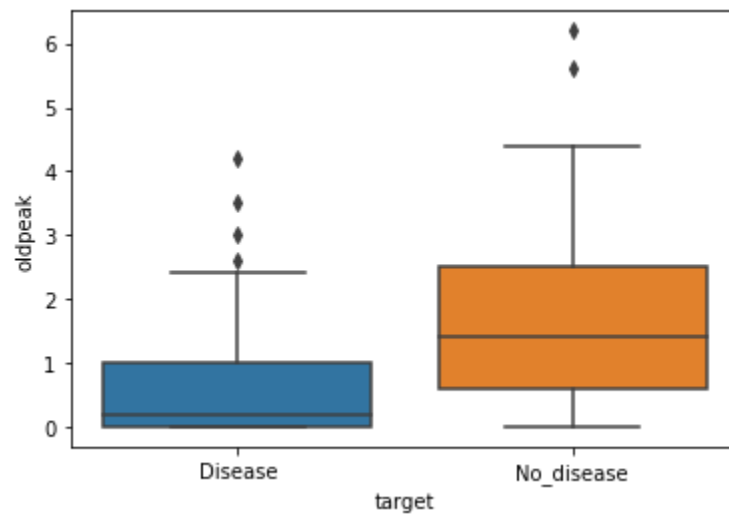
Before we plot the outliers, let's change the labeling for better visualization and interpretation for heart dataset.

```
df['target'] = df.target.replace({1: "Disease", 0: "No_disease"})
df['sex'] = df.sex.replace({1: "Male", 0: "Female"})
df['cp'] = df.cp.replace({0: "typical_angina",
                        1: "atypical_angina",
                        2: "non-anginal pain",
                        3: "asymtomatic"})
df['exang'] = df.exang.replace({1: "Yes", 0: "No"})
df['fbs'] = df.fbs.replace({1: "True", 0: "False"})
df['slope'] = df.slope.replace({0: "upsloping", 1: "flat", 2: "downsloping"})
df['thal'] = df.thal.replace({1: "fixed_defect", 2: "reversible_defect", 3: "normal"})

import matplotlib.pyplot as plt
import seaborn as sb
bxplt = sb.boxplot(df["target"], df["chol"])
plt.show()
```



```
sb.boxplot(x='target', y='oldpeak', data=df)
<matplotlib.axes._subplots.AxesSubplot at 0x7f95fceb7e10>
```



```
define continuous variable & plot
continous_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
def outliers(df_out, drop = False):
    for each_feature in df_out.columns:
        feature_data = df_out[each_feature]
        Q1 = np.percentile(feature_data, 25.) # 25th percentile of the
data of the given feature
        Q3 = np.percentile(feature_data, 75.) # 75th percentile of the
data of the given feature
        IQR = Q3-Q1 #Interquartile Range
        outlier_step = IQR * 1.5 #That's we were talking about above
```

```

        outliers = feature_data[~((feature_data >= Q1 - outlier_step) &
(feature_data <= Q3 + outlier_step))].index.tolist()
        if not drop:
            print('For the feature {}, No of Outliers is
{}'.format(each_feature, len(outliers)))
        if drop:
            df.drop(outliers, inplace = True, errors = 'ignore')
            print('Outliers from {} feature removed'.format(each_feature))

outliers(df[continous_features])

```

```

For the feature age, No of Outliers is 0
For the feature trestbps, No of Outliers is 9
For the feature chol, No of Outliers is 5
For the feature thalach, No of Outliers is 1
For the feature oldpeak, No of Outliers is 5

```

Drop Outliers

```

outliers(df[continous_features],drop=True)
Outliers from age feature removed
Outliers from trestbps feature removed
Outliers from chol feature removed
Outliers from thalach feature removed
Outliers from oldpeak feature removed

```

Duplicate rows:

Datasets may contain duplicate entries. It is one of the easiest tasks to delete duplicate rows. To delete the duplicate rows you can use —*dataset_name.drop_duplicates()*.

We have used another approach on the heart dataset.

Checking for the duplicate rows

```

duplicated=df.duplicated().sum()

if duplicated:

    print("Duplicated rows :{}".format(duplicated))

else:

    print("No duplicates")

```

```
Duplicated rows :1
```

Displaying duplicate rows

```
duplicates=df[df.duplicated(keep=False)]
```

```
duplicates.head()
```

Remove the duplicate rows using the above mentioned method.

2.Data integration

So far, we've made sure to remove the impurities in data and make it clean. Now, the next step is to combine data from different sources to get a unified structure with more meaningful and valuable information. This is mostly used if the data is segregated into different sources. To make it simple, let's assume we have data in CSV format in different places, all talking about the same scenario. Say we have some data about an employee in a database. We can't expect all the data about the employee to reside in the same table. It's possible that the employee's personal data will be located in one table, the employee's project history will be in a second table, the employee's time-in and time-out details will be in another table, and so on. So, if we want to do some analysis about the employee, we need to get all the employee data in one common place. This process of bringing data together in one place is called data integration. To do data integration, we can merge multiple pandas DataFrames using the merge function.

In this exercise, we'll merge the details of students from two datasets, namely student.csv and marks.csv. The student dataset contains columns such as Age, Gender, Grade, and Employed. The marks.csv dataset contains columns such as Mark and City. The Student_id column is common between the two datasets. Follow these steps to complete this exercise:

```
dataset1 ="https://github.com/TrainingByPackt/Data-Science-with-
```

```
Python/blob/master/Chapter01/Data/student.csv"
```

```
dataset2 = "https://github.com/TrainingByPackt/Data-Science-with-
```

```
Python/blob/master/Chapter01/Data/mark.csv"
```

```
df1 = pd.read_csv(dataset1, header = 0)
df2 = pd.read_csv(dataset2, header = 0)
```

To print the first five rows of the first DataFrame, add the following code:

```
df1.head()
```

The preceding code generates the following output:

	Student_id	Mark	City
0	1	95	Chennai
1	2	70	Delhi
2	3	98	Mumbai
3	4	75	Pune
4	5	89	Kochi

Figure : The first five rows of the first DataFrame

To print the first five rows of the second DataFrame, add the following code:

```
df2.head()
```

The preceding code generates the following output:

	Student_id	Age	Gender	Grade	Employed
0	1	19	Male	1st Class	yes
1	2	20	Female	2nd Class	no
2	3	18	Male	1st Class	no
3	4	21	Female	2nd Class	no
4	5	19	Male	1st Class	no

Figure : The first five rows of the second DataFrame

Student_id is common to both datasets. Perform data integration on both the DataFrames with respect to the Student_id column using the pd.merge() function, and then print the first 10 values of the new DataFrame:

```
df = pd.merge(df1, df2, on = 'Student_id')
df.head(10)
```

	Student_id	Mark	City	Age	Gender	Grade	Employed
0	1	95	Chennai	19	Male	1st Class	yes
1	2	70	Delhi	20	Female	2nd Class	no
2	3	98	Mumbai	18	Male	1st Class	no
3	4	75	Pune	21	Female	2nd Class	no
4	5	89	Kochi	19	Male	1st Class	no
5	6	69	Gwalior	20	Male	2nd Class	yes
6	7	52	Bhopal	19	Female	3rd Class	yes
7	8	54	Chennai	21	Male	3rd Class	yes
8	9	55	Delhi	22	Female	3rd Class	yes
9	10	94	Mumbai	21	Male	1st Class	no

Figure : First 10 rows of the merged DataFrame

Here, the data of the df1 DataFrame is merged with the data of the df2 DataFrame. The merged data is stored inside a new DataFrame called df.

We have now learned how to perform data integration. In the next section, we'll explore another pre-processing task, data transformation.

3.Data transformation

Previously, we saw how we can combine data from different sources into a unified dataframe. Now, we have a lot of columns that have different types of data. Our goal is to transform the data into a machine-learning-digestible format. All machine learning algorithms are based on mathematics. So, we need to convert all the columns into numerical format. Before that, let's see all the different types of data we have.

Taking a broader perspective, data is classified into numerical and categorical data:

- Numerical: As the name suggests, this is numeric data that is quantifiable.
- Categorical: The data is a string or non-numeric data that is qualitative in nature.

Numerical data is further divided into the following:

- Discrete: To explain in simple terms, any numerical data that is countable is called discrete, for example, the number of people in a family or the number of students in a class. Discrete data can only take certain values (such as 1, 2, 3, 4, etc).
- Continuous: Any numerical data that is measurable is called continuous, for example, the height of a person or the time taken to reach a destination. Continuous data can take virtually any value (for example, 1.25, 3.8888, and 77.1276).

Categorical data is further divided into the following:

- Ordered: Any categorical data that has some order associated with it is called ordered categorical data, for example, movie ratings (excellent, good, bad, worst) and feedback (happy, not bad, bad). You can think of ordered data as being something you could mark on a scale.
- Nominal: Any categorical data that has no order is called nominal categorical data. Examples include gender and country.

From these different types of data, we will focus on categorical data. In the next section, we'll discuss how to handle categorical data.

Handling Categorical Data

There are some algorithms that can work well with categorical data, such as decision trees. But most machine learning algorithms cannot operate directly with categorical data. These algorithms require the input and output both to be in numerical form. If the output to be predicted is categorical, then after prediction we convert them back to categorical data from numerical data. Let's discuss some key challenges that we face while dealing with categorical data:

- High cardinality: Cardinality means uniqueness in data. The data column, in this case, will have a lot of different values. A good example is User ID – in a table of 500 different users, the User ID column would have 500 unique values.
- Rare occurrences: These data columns might have variables that occur very rarely and therefore would not be significant enough to have an impact on the model.
- Frequent occurrences: There might be a category in the data columns that occurs many times with very low variance, which would fail to make an impact on the model.
- Won't fit: This categorical data, left unprocessed, won't fit our model.

Encoding

To address the problems associated with categorical data, we can use encoding. This is the process by which we convert a categorical variable into a numerical form. Here, we will look at three simple methods of encoding categorical data.

Replacing

This is a technique in which we replace the categorical data with a number. This is a simple replacement and does not involve much logical processing. Let's look at an exercise to get a better idea of this.

Exercise 6: Simple Replacement of Categorical Data with a Number

In this exercise, we will use the student dataset that we saw earlier. We will load the data into a pandas dataframe and simply replace all the categorical data with numbers. Follow these steps to complete this exercise:

In AirQuality dataset, applying Simple Replacement of Categorical Data with a Number.

```
df['type'].value_counts()
RRO      179013
I         148069
RO        86791
S         15010
RIRUO     1304
R          158
```

The column type in the dataframe has 6 unique values, which we will be replacing with numbers.

```
df['type'].replace({"RRO":1, "I":2, "RO":3, "S":4, "RIRUO":5, "R":6},
inplace= True)
df['type']
0         1.0
1         2.0
2         1.0
```

```

3          1.0
4          2.0
...
435734     5.0
435735     5.0
435736     5.0
435737     5.0
435738     5.0
Name: type, Length: 435735, dtype: float64

```

Converting Categorical Data to Numerical Data Using Label Encoding

This is a technique in which we replace each value in a categorical column with numbers from 0 to N-1. For example, say we've got a list of employee names in a column. After performing label encoding, each employee name will be assigned a numeric label. But this might not be suitable for all cases because the model might consider numeric values to be weights assigned to the data. Label encoding is the best method to use for ordinal data. The scikit-learn library provides `LabelEncoder()`, which helps with label encoding.

```

df['state'].value_counts()

```

Maharashtra	60382
Uttar Pradesh	42816
Andhra Pradesh	26368
Punjab	25634
Rajasthan	25589
Kerala	24728
Himachal Pradesh	22896
West Bengal	22463
Gujarat	21279
Tamil Nadu	20597
Madhya Pradesh	19920
Assam	19361
Odisha	19278
Karnataka	17118
Delhi	8551
Chandigarh	8520
Chhattisgarh	7831
Goa	6206
Jharkhand	5968
Mizoram	5338
Telangana	3978
Meghalaya	3853
Puducherry	3785

```

Haryana          3420
Nagaland         2463
Bihar            2275
Uttarakhand      1961
Jammu & Kashmir   1289
Daman & Diu       782
Dadra & Nagar Haveli 634
Uttaranchal      285
Arunachal Pradesh 90
Manipur          76
Sikkim           1
Name: state, dtype: int64

```

```

from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
df["state"]=labelencoder.fit_transform(df["state"])
df.head(5)

```

	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	0	Hyderabad	1.0	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990
1	0	Hyderabad	2.0	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01	1990
2	0	Hyderabad	1.0	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01	1990
3	0	Hyderabad	1.0	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01	1990
4	0	Hyderabad	2.0	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01	1990

One Hot Encoding

In label encoding, categorical data is converted to numerical data, and the values are assigned labels (such as 1, 2, and 3). Predictive models that use this numerical data for analysis might sometimes mistake these labels for some kind of order (for example, a model might think that a label of 3 is "better" than a label of 1, which is incorrect). In order to avoid this confusion, we can use one-hot encoding. Here, the label-encoded data is further divided into n number of columns. Here, n denotes the total number of unique labels generated while performing label encoding. For example, say that three new labels are generated through label encoding. Then,

while performing one-hot encoding, the columns will be divided into three parts. So, the value of n is 3.

```
dfAndhra=df[(df['state']==0)]
```

```
dfAndhra
```

	state	location	type	so2	no2	rspm	spm	pm2_5	date	year
0	0	Hyderabad	1.0	4.8	17.4	108.833091	220.78348	40.791467	1990-02-01	1990
1	0	Hyderabad	2.0	3.1	7.0	108.833091	220.78348	40.791467	1990-02-01	1990
2	0	Hyderabad	1.0	6.2	28.5	108.833091	220.78348	40.791467	1990-02-01	1990
3	0	Hyderabad	1.0	6.3	14.7	108.833091	220.78348	40.791467	1990-03-01	1990
4	0	Hyderabad	2.0	4.7	7.5	108.833091	220.78348	40.791467	1990-03-01	1990
...
26363	0	Rajahmundry	2.0	7.0	13.0	71.000000	220.78348	40.791467	2015-12-13	2015
26364	0	Rajahmundry	2.0	7.0	18.0	77.000000	220.78348	40.791467	2015-12-16	2015
26365	0	Rajahmundry	2.0	8.0	23.0	64.000000	220.78348	40.791467	2015-12-19	2015
26366	0	Rajahmundry	2.0	7.0	19.0	61.000000	220.78348	40.791467	2015-12-22	2015

```
dfAndhra['location'].value_counts()
```

Hyderabad	7764
Visakhapatnam	7108
Vijayawada	2093
Chittoor	1003
Tirupati	986
Kurnool	857
Patancheru	698
Guntur	629
Nalgonda	618
Ramagundam	554
Nellore	408
Khammam	385
Warangal	336
Ananthapur	324
Ongole	317
Kadapa	316
Srikakulam	315
Rajahmundry	311
Eluru	300
Vishakhapatnam	297
Kakinada	288
Vizianagaram	282
Sangareddy	85
Karimnagar	67
Nizamabad	27

Name: location, dtype: int64

```

from sklearn.preprocessing import OneHotEncoder
onehotencoder=OneHotEncoder(sparse=False,handle_unknown='error',drop='first')
pd.DataFrame(onehotencoder.fit_transform(dfAndhra[["location"]]))

```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...
26363	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26364	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26365	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26366	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
26367	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

26368 rows × 24 columns

You have successfully converted categorical data to numerical data using the OneHotEncoder method.

4.Error Correction

In heart dataset it can be observed that feature 'ca' should range from 0–3, however, df.nunique() listed 0–4. So let's find the '4' and change them to NaN.

```
df['ca'].unique()
```

```
array([0, 2, 1, 3, 4], dtype=object)
```

```
df[df['ca']==4]
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
92	52	1	2	138	223	0	1	169	0	0.0	2	4	2	1
158	58	1	1	125	220	0	1	144	0	0.4	1	4	3	1
163	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1
251	43	1	0	132	247	1	0	143	1	0.1	1	4	3	0

```
df.loc[df['ca']==4, 'ca']=np.NaN
```

Similarly, Feature ‘thal’ ranges from 1–3, however, df.nunique() listed 0–3. There are two values of ‘0’. They are also changed to NaN using above mentioned technique.

Now, we can replace changed NaN values(missing values).

```
df.isna().sum()
```

```
age      0
sex      0
cp       0
trestbps 0
chol     0
fbs      0
restecg  0
thalach  0
exang    0
oldpeak  0
slope    0
ca       5
thal     2
target   0
dtype: int64
```

```
df = df.fillna(df.median())
```

```
df.isnull().sum()
```



```
age          0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
target       0
dtype: int64
```

5.Data model building

Once you've pre-processed your data into a format that's ready to be used by your model, you need to split up your data into train and test sets. This is because your machine learning algorithm will use the data in the training set to learn what it needs to know. It will then make a prediction about the data in the test set, using what it has learned. You can then compare this prediction against the actual target variables in the test set in order to see how accurate your model is. The exercise in the next section will give more clarity on this.

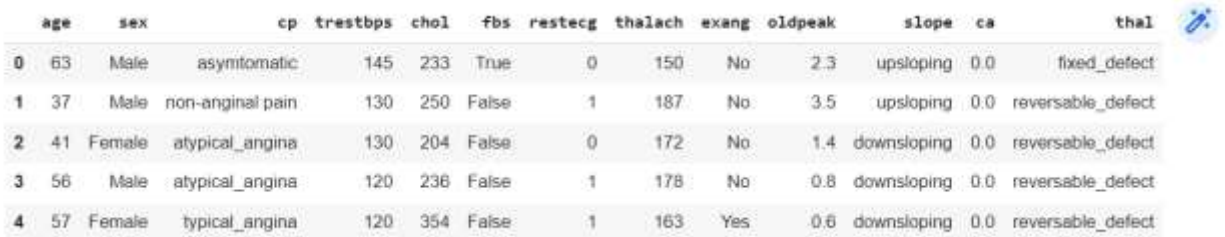
We will do the train/test split in proportions. The larger portion of the data split will be the train set and the smaller portion will be the test set. This will help to ensure that you are using enough data to accurately train your model.

In general, we carry out the train-test split with an 80:20 ratio, as per the Pareto principle. The Pareto principle states that "for many events, roughly 80% of the effects come from 20% of the causes." But if you have a large dataset, it really doesn't matter whether it's an 80:20 split or 90:10 or 60:40. (It can be better to use a smaller split set for the training set if our process is computationally intensive, but it might cause the problem of overfitting – this will be covered later in the book.)

Create a variable called X to store the independent features. Use the drop() function to include all the features, leaving out the dependent or the target variable, which in this case is named 'target' for heart dataset. Then, print out the top five instances of the variable. Add the following code to do this:

```
X = df.drop('target', axis=1)
```

```
X.head()
```



	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	Male	asymptomatic	145	233	True	0	150	No	2.3	upsloping	0.0	fixed_defect
1	37	Male	non-anginal pain	130	250	False	1	187	No	3.5	upsloping	0.0	reversable_defect
2	41	Female	atypical_angina	130	204	False	0	172	No	1.4	downsloping	0.0	reversable_defect
3	56	Male	atypical_angina	120	236	False	1	178	No	0.8	downsloping	0.0	reversable_defect
4	57	Female	typical_angina	120	354	False	1	163	Yes	0.6	downsloping	0.0	reversable_defect

Figure : Dataframe consisting of independent variables

1. Print the shape of your new created feature matrix using the X.shape command:

```
X.shape
```

The preceding code generates the following output:

```
(284, 13)
```

Figure : Shape of the X variable

In the preceding figure, the first value indicates the number of observations in the dataset (284), and the second value represents the number of features (13).

2. Similarly, we will create a variable called y that will store the target values. We will use indexing to grab the target column. Indexing allows us to access a section of a larger element. In this case, we want to grab the column named Price from the df dataframe and print out the top 10 values. Add the following code to implement this:

```
y = df['target']
```

y.head(10)

The preceding code generates the following output:

```
0    Disease
1    Disease
2    Disease
3    Disease
4    Disease
5    Disease
6    Disease
7    Disease
9    Disease
10   Disease
Name: target, dtype: object
```

Figure : Top 10 values of the y variable

3. Print the shape of your new variable using the y.shape command:

y.shape

The preceding code generates the following output:

```
(284,)
```

Figure : Shape of the y variable

The shape should be one-dimensional, with a length equal to the number of observations (284).

4. Make train/test sets with an 80:20 split. To do so, use the train_test_split() function from the sklearn.model_selection package. Add the following code to do this:

```
from sklearn.model_selection import train_test_split
```

```
from sklearn import preprocessing
```

```
df=df.apply(preprocessing.LabelEncoder().fit_transform)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
```

In the preceding code, test_size is a floating-point value that defines the size of the test data. If the value is 0.2, then it is an 80:20 split. train_test_split splits the arrays or matrices into train and test subsets in a random way. Each time we run the code without random_state, we will get a different result.

5. Print the shape of X_train, X_test, y_train, and y_test. Add the following code to do this:

```
print("X_train : ",X_train.shape)
print("X_test : ",X_test.shape)
print("y_train : ",y_train.shape)
print("y_test : ",y_test.shape)
```

The preceding code generates the following output:

```
X_train :  (227, 13)
X_test   :  (57, 13)
y_train  :  (227,)
y_test   :  (57,)
```

Figure : Shape of train and test datasets

You have successfully split the data into train and test sets.

In the next section, you will complete an activity wherein you'll perform pre-processing on a dataset.

Supervised Learning

Supervised learning is a learning system that trains using labeled data (data in which the target variables are already known). The model learns how patterns in the feature matrix map to the target variables. When the trained machine is fed with a new dataset, it can use what it has learned to predict the target variables. This can also be called predictive modeling.

Supervised learning is broadly split into two categories. These categories are as follows:

Classification mainly deals with categorical target variables. A classification algorithm helps to predict which group or class a data point belongs to.

When the prediction is between two classes, it is known as binary classification. An example is predicting whether or not a person has a heart disease (in this case, the classes are yes and no).

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
clf = DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(y_pred)
```

```
[0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1
 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 0 0 0 1 1]
```

If the prediction involves more than two target classes, it is known as multi-classification; for example, predicting all the items that a customer will buy.

Regression deals with numerical target variables. A regression algorithm predicts the numerical value of the target variable based on the training dataset.

Linear regression measures the link between one or more predictor variables and one outcome variable. For example, linear regression could help to enumerate the relative impacts of age, gender, and diet (the predictor variables) on height (the outcome variable).

Time series analysis, as the name suggests, deals with data that is distributed with respect to time, that is, data that is in a chronological order. Stock market prediction and customer churn prediction are two examples of time series data. Depending on the requirement or the necessities, time series analysis can be either a regression or classification task.

Unsupervised Learning

Unlike supervised learning, the unsupervised learning process involves data that is neither classified nor labeled. The algorithm will perform analysis on the data without guidance. The job of the machine is to group unclustered information according to similarities in the data. The aim is for the model to spot patterns in the data in order to give some insight into what the data is telling us and to make predictions.

An example is taking a whole load of unlabeled customer data and using it to find patterns to cluster customers into different groups. Different products could then be marketed to the different groups for maximum profitability.

Unsupervised learning is broadly categorized into two types:\

Clustering: A clustering procedure helps to discover the inherent patterns in the data.

Association: An association rule is a unique way to find patterns associated with a large amount of data, such as the supposition that when someone buys product 1, they also tend to buy product 2.

References:

[1][What is Data Cleaning? How to Process Data for Analytics and Machine Learning Modeling? | by Awan-Ur-Rahman | Towards Data Science](#)

CONCLUSION:

This assignment covers the most crucial step in Machine learning is data preprocessing.

Assignment: 6

Aim: Integrate Python and Hadoop and perform the following operations on forest fire dataset

- a. Data analysis using the Map Reduce in PyHadoop
- b. Data mining in Hive

Theory:

- a. Data analysis using the Map Reduce in PyHadoop

Spark – Overview

Apache Spark is a lightning fast real-time processing framework. It does in-memory computations to analyze data in real-time. It came into picture as Apache Hadoop MapReduce was performing batch processing only and lacked a real-time processing feature. Hence, Apache Spark was introduced as it can perform stream processing in real-time and can also take care of batch processing.

Apart from real-time and batch processing, Apache Spark supports interactive queries and iterative algorithms also. Apache Spark has its own cluster manager, where it can host its application. It leverages Apache Hadoop for both storage and processing. It uses HDFS (Hadoop Distributed File system) for storage and it can run Spark applications on YARN as well.

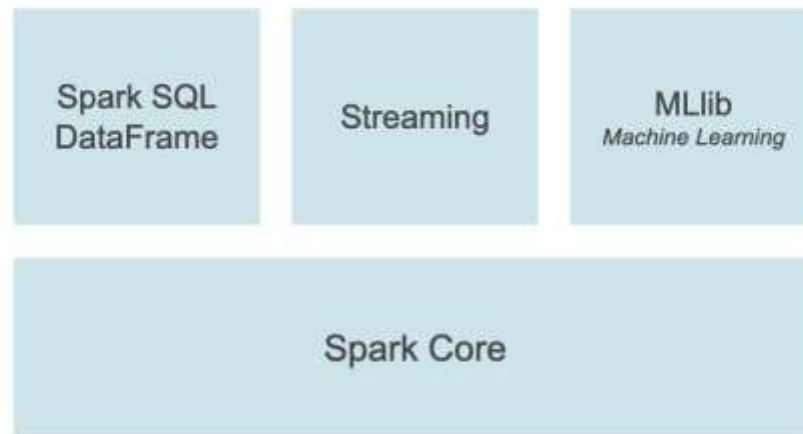
PySpark – Overview

Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark Community released a tool, PySpark. Using PySpark, you can work with RDDs in Python programming language also. It is because of a library called Py4j that they are able to achieve this.

PySpark offers PySpark Shell which links the Python API to the spark core and initializes the Spark context. Majority of data scientists and analytics experts today use Python because of its rich library set. Integrating Python with Spark is a boon to them.

PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing your data in a distributed environment. PySpark supports most of Spark's features such as Spark SQL, DataFrame, Streaming, MLlib (Machine Learning) and Spark Core.

Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark community released a tool, PySpark. Using PySpark, you can work with RDDs in Python programming language also. It is because of a library called Py4j that they are able to achieve this. This is an introductory tutorial, which covers the basics of Data-Driven Documents and explains how to deal with its various components and sub-components.



Spark SQL and DataFrame

Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrame and can also act as distributed SQL query engine.

pandas API on Spark

pandas API on Spark allows you to scale your pandas workload out. With this package, you can:

- Be immediately productive with Spark, with no learning curve, if you are already familiar with pandas.
- Have a single codebase that works both with pandas (tests, smaller datasets) and with Spark (distributed datasets).
- Switch to pandas API and PySpark API contexts easily without any overhead.

Streaming

Running on top of Spark, the streaming feature in Apache Spark enables powerful interactive and analytical applications across both streaming and historical data, while inheriting Spark's ease of use and fault tolerance characteristics.

MLlib

Built on top of Spark, MLlib is a scalable machine learning library that provides a uniform set of high-level APIs that help users create and tune practical machine learning pipelines.

Spark Core

Spark Core is the underlying general execution engine for the Spark platform that all other functionality is built on top of. It provides an RDD (Resilient Distributed Dataset) and in-memory computing capabilities.

Features

Following are the main features of PySpark.

- In-memory computation
- Distributed processing using parallelize
- Can be used with many cluster managers (Spark, Yarn, Mesos e.t.c)
- Fault-tolerant
- Immutable
- Lazy evaluation
- Cache & persistence
- Inbuild-optimization when using DataFrames
- Supports ANSI SQL

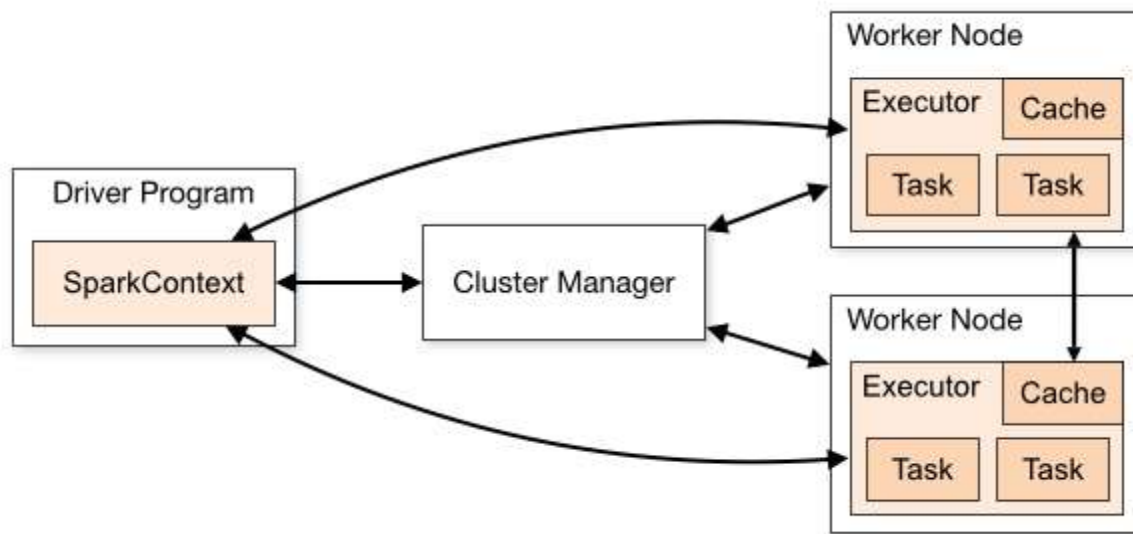
Advantages of PySpark

- PySpark is a general-purpose, in-memory, distributed processing engine that allows you to process data efficiently in a distributed fashion.
- Applications running on PySpark are 100x faster than traditional systems.

- You will get great benefits using PySpark for data ingestion pipelines.
- Using PySpark we can process data from Hadoop HDFS, AWS S3, and many file systems.
- PySpark also is used to process real-time data using Streaming and Kafka.
- Using PySpark streaming you can also stream files from the file system and also stream from the socket.
- PySpark natively has machine learning and graph libraries.

PySpark Architecture

Apache Spark works in a master-slave architecture where the master is called “Driver” and slaves are called “Workers”. When you run a Spark application, Spark Driver creates a context that is an entry point to your application, and all operations (transformations and actions) are executed on worker nodes, and the resources are managed by Cluster Manager.



source: <https://spark.apache.org/>

b. Data mining in Hive

Hive is a datawarehouseing infrastructure for Hadoop. The primary responsibility is to provide data summarization, query and analysis. It supports analysis of large datasets stored in Hadoop's HDFS as well as on the Amazon S3 filesystem. The best part of HIVE is that it supports SQL-

Like access to structured data which is known as HiveQL (or HQL) as well as big data analysis with the help of MapReduce. Hive is not built to get a quick response to queries but it is built for data mining applications. Data mining applications can take from several minutes to several hours to analysis the data and HIVE is primarily used there.

Hive query language provides the basic SQL like operations. Here are few of the tasks which HQL can do easily.

- Create and manage tables and partitions
- Support various Relational, Arithmetic and Logical Operators
- Evaluate functions
- Download the contents of a table to a local directory or result of queries to HDFS directory

Here is the example of the HQL Query:

```
1  SELECT upper(name), salesprice
2  FROM sales;
3  SELECT category, count(1)
4  FROM products
5  GROUP BY category;
```

When you look at the above query, you can see they are very similar to SQL like queries.

What is the data mining in Hive?

Hive is a batch-oriented and data -warehousing layer created on the basic elements of Hadoop , such as HDFS and mapreduce. This layer plays an important role in mining of big data . Hive offers a simple SQL -lite-implementation call hiveQL to SQL users without losing access through mappers and reducers.

Mining Big Data with Hive

Hive organises data using three mechanisms:

Tables: Hive tables are similar to RDBMS tables because they contain rows and columns. Tables are mapped to file system directories since Hive is layered on the Hadoop HDFS. Hive also supports tables saved in different native file systems.

Partitioning: A Hive table can have one or more partitions. These partitions indicate data distribution across the table and are mapped to subdirectories in the underlying file system.

Buckets: Data can be separated into buckets. Buckets are saved as files in the underlying file system's partition directory. The hash of a table column determines the buckets. In the prior example, you might have a bucket called Focus that contains all of the characteristics of a Ford Focus vehicle.

REFERENCES: <https://spark.apache.org/docs/latest/cluster-overview.html>

<https://intellipaat.com/blog/tutorial/spark-tutorial/spark-architecture/>

CONCLUSION: Thus we have studied Spark and data analytics using Hive

Assignment: 7

Aim: Visualize the data using R/Python by plotting the graphs for assignment no. 2 and 3

OBJECTIVE:

1. To understand and apply the analytical concept of Big data using Python.
2. To study detailed data visualization techniques in Python programing.

SOFTWARE REQUIREMENTS:

1. Ubuntu 16.04 / 18.04
2. Python

THEORY:

INTRODUCTION TO VISUALIZATION

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible wayto see and understand trends, outliers, and patterns in data.

We can use Matplotlib and Seaborn library for data Visualization

```
import matplotlib.pyplot as plt
```

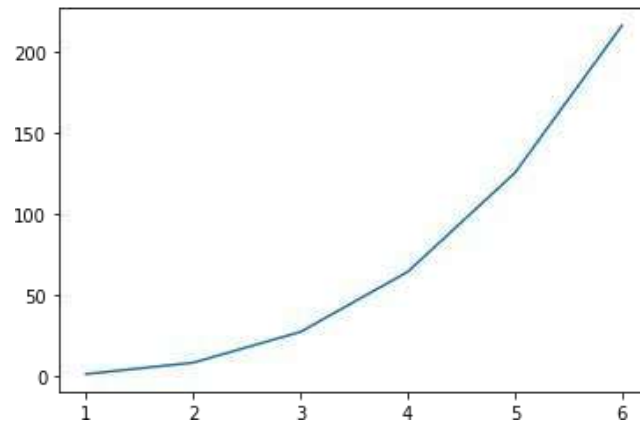
Matplotlib is one of the most popular Python packages used for data visualization. It is across-platform library for making 2D plots from data in arrays.

Line Chart

A line chart is a type of chart that displays information as a series of data points connectedby straight line segments. A line chart is a way of visually representing an asset's price history using a single, continuous line.

```
x = [1,2,3,4,5,6]
y = [1,8,27,64,125,216]
```

```
plt.plot(x,  
y)
```



```
plt.show()
```

BAR PLOT

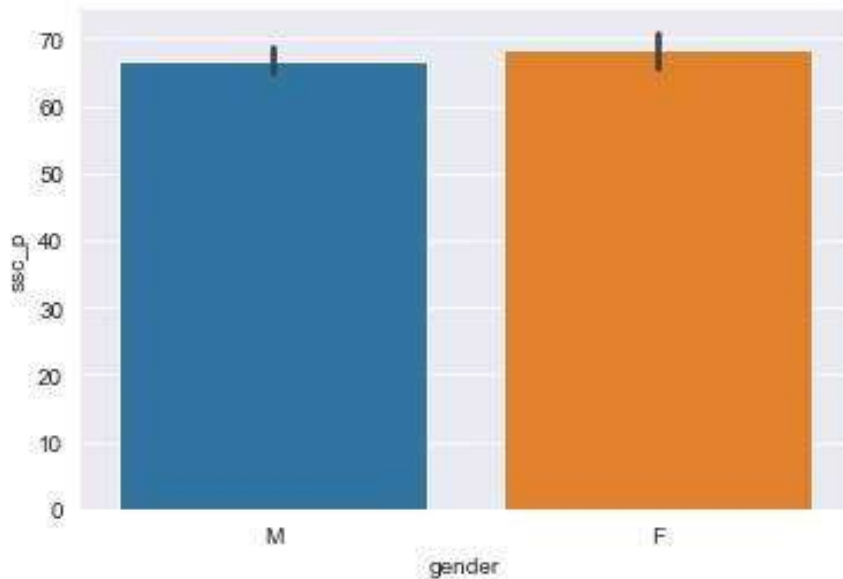
Takes the average of the categorical data. & Count plot gives the count of the unique values in the data.

We have confidence intervals via bootstrapping.

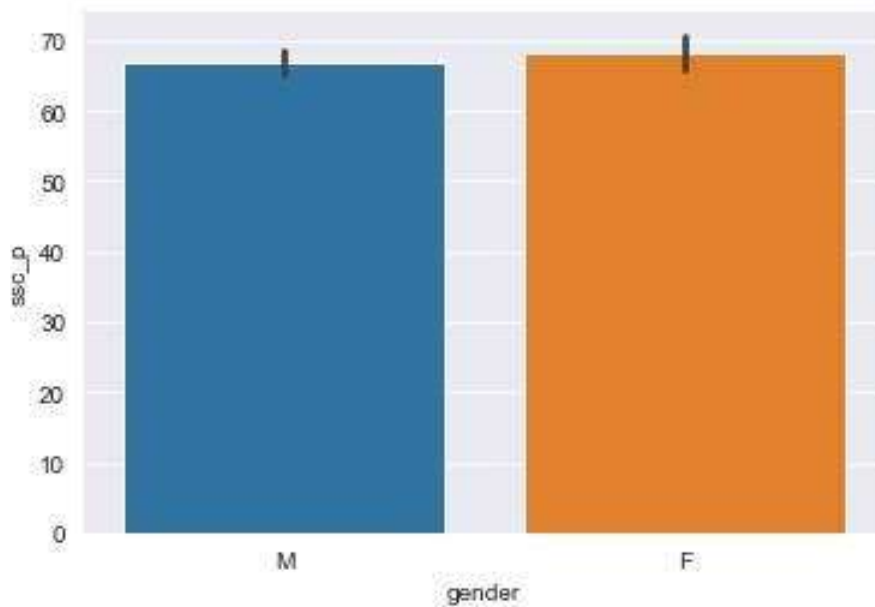
This is the error bar. Error bars are graphical representations of the variability of data and used on graphs to indicate the error or uncertainty in a reported measurement.

```
sns.barplot(x='gender', y='ssc_p', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2570637b8c8>
```



```
sns.barplot(x='gender', y='ssc_p', data=df, estimator=np.mean)
<matplotlib.axes._subplots.AxesSubplot at 0x25706435d48>
```



HEATMAP

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter

colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the `seaborn.heatmap()` function.

we can find correlation between columns by using `corr()` function and plot a heatmap based on the correlation

```
df1 = df[['ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p']]
```

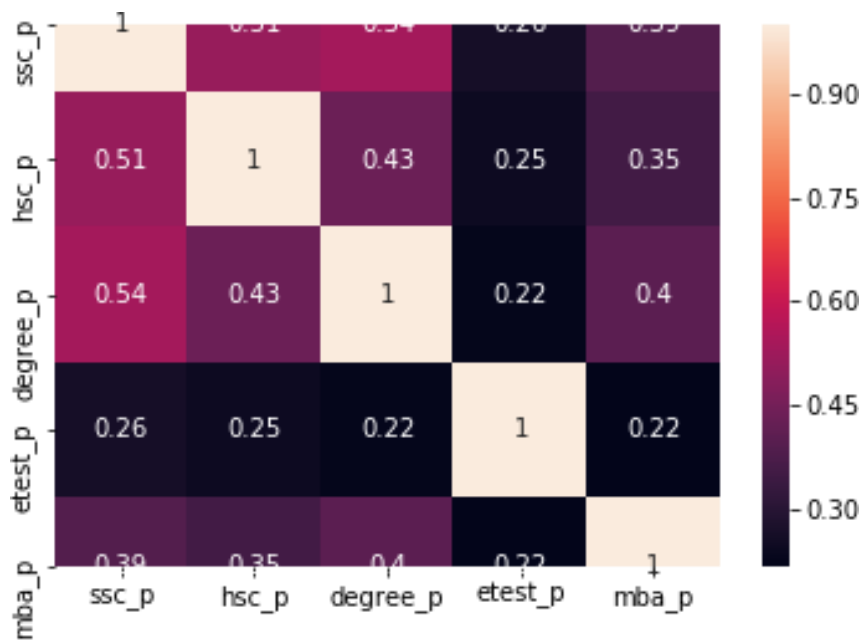
```
df1
```

	ssc_p	hsc_p	degree_p	etest_p	mba_p
0	67.00	91.00	58.00	55.0	58.80
1	79.33	78.33	77.48	86.5	66.28
2	65.00	68.00	64.00	75.0	57.80
3	56.00	52.00	52.00	66.0	59.43
4	85.80	73.60	73.30	96.8	55.50
..
210	80.60	82.00	77.60	91.0	74.49
211	58.00	60.00	72.00	74.0	53.62
212	67.00	67.00	73.00	59.0	69.72
213	74.00	66.00	58.00	70.0	60.23
214	62.00	58.00	53.00	89.0	60.22

```
[215 rows x 5 columns]
```

```
sns.heatmap(df1.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x26dbf2994c8>
```



HISTOGRAM

A histogram is a graphical representation that organizes a group of data points into user- specified ranges. Similar in appearance to a bar graph, the histogram condenses a data series into an easily interpreted visual by taking many data points and grouping them into logical ranges or bins.

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df =
```

```
pd.read_csv('tested.csv') df
```

	PassengerId	Surv:
0	892	
1	893	
2	894	
3	895	
4	896	
..	...	
413	1305	
414	1306	
415	1307	
416	1308	
416	Ware, Mr.	Frederick male NaN 0

```

0
417 Peter, Master.           Michael J    male    NaN    1
1

```

```

      Ticket      Fare Cabin Embarked
0      330911      7.8292   NaN        Q
1      363272      7.0000   NaN        S
2      240276      9.6875   NaN        Q
3      315154      8.6625   NaN        S
4      3101298     12.2875   NaN        S
..      ...      ...      ...      ...
413      A.5. 3236      8.0500   NaN        S
414      PC 17758     108.9000  C105        C
415  SOTON/O.Q. 3101262      7.2500   NaN        S
416      359309      8.0500   NaN        S
417      2668      22.3583   NaN        C

```

```
[418 rows x 12 columns]
```

```
plt.hist(df['Age'])
```

```

C:\Users\omkar\anacondanew\lib\site-packages\numpy\lib\
histograms.py:824: RuntimeWarning: invalid value encountered in
greater_equal

```

```
    keep = (tmp_a >= first_edge)
```

```

C:\Users\omkar\anacondanew\lib\site-packages\numpy\lib\
histograms.py:825: RuntimeWarning: invalid value encountered
inless_equal

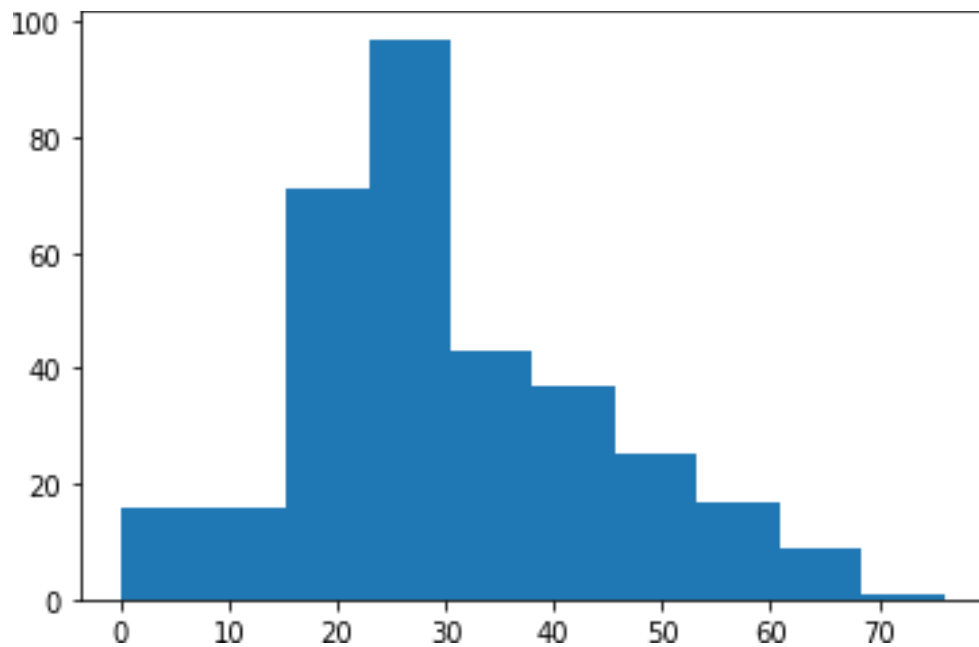
```

```
    keep &= (tmp_a <= last_edge)
```

```

(array([16., 16., 71., 97., 43., 37., 25., 17., 9., 1.]),
 array([ 0.17 ,  7.753, 15.336, 22.919, 30.502, 38.085, 45.668,
53.251,
        60.834, 68.417, 76.   ])),
<a list of 10 Patch objects>)

```



PIE CHARTS

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

PIE CHART

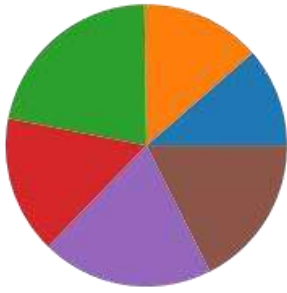
Pie chart is a graph where a circle is divided in parts to show the percentage of each unique values in each column.

It represents what a quantity contributes to the whole

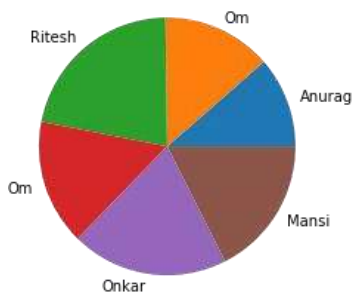
SYNTAX :-

```
plt.pie(<seq>, labels=None, explode=None,
        autopct=None,
        colors=None, shadow=None)

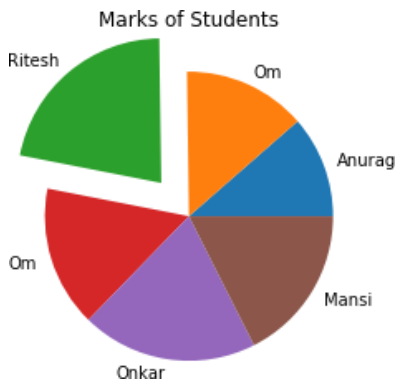
marks = [52, 63, 99, 72, 90, 80]
plt.pie(marks)
plt.show()
```



```
Student_name = ['Anurag','Om','Ritesh','Om','Onkar','Mansi']
plt.pie(marks, labels = Student_name)
plt.show()
```



```
# Explode = Separate the one we need, out from the pie chart, HERE
max marks
exp = [0,0,0.3,0,0,0]
plt.pie(marks, labels = Student_name,
explode=exp)plt.title('Marks of Students')
plt.show()
```



SCATTER PLOT

Use a scatter plot to determine whether or not two variables have a relationship or correlation. Are you trying to see if your two variables might mean something when put together? Plotting a scattergram with your data points can help you to determine whether there's a potential relationship between them.

We can detect outliers with scatter

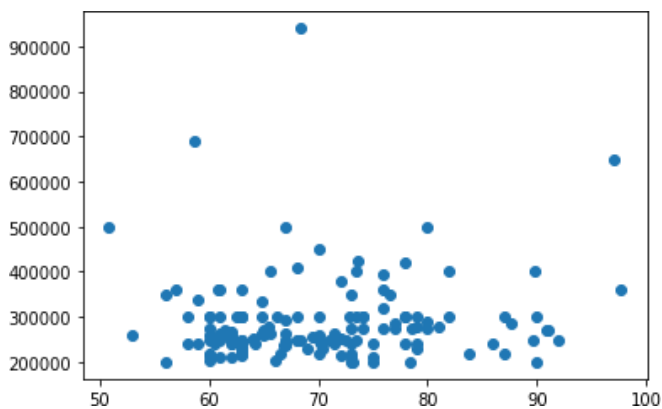
plot. We can find different trends.

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('Placement_Data_Full_Class.csv')

plt.scatter(df['hsc_p'], df['salary'])

<matplotlib.collections.PathCollection at 0x1b9281c0f88>
```



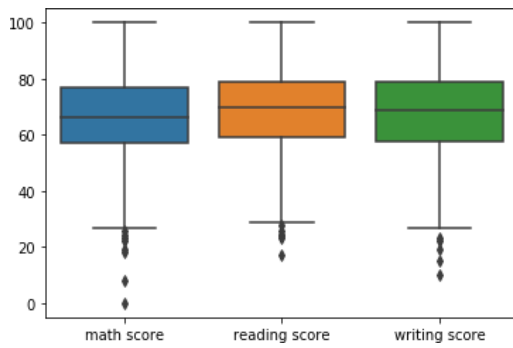
BOX PLOT

are used to show that distribution of categorical data

And it shows the distribution of quantitative data in a way that hopefully facilitates comparisons between the variables.

```
sns.boxplot(data=df, orient='v')

<matplotlib.axes._subplots.AxesSubplot at 0x22f55388c88>
```



REFERENCES:

<https://seaborn.pydata.org/introduction.html>

<https://matplotlib.org/>

CONCLUSION: Visualization of Datasets used in 4th and 5th assignment is completed in python. We can use Matplotlib, Seaborn, Plotly tools in python for Data visualization.

Assignment: 8

AIM: Perform different data visualization operations using Tableau

PROBLEM STATEMENT /DEFINITION

Study following data visualization operations using Tableau on Adult, Iris datasets or retail dataset.

- 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

Perform the data visualization operations using Tableau to get answers to various business questions on Retail dataset.

- a. Find and Plot top 10 products based on total sale
- b. Find and Plot product contribution to total sale
- c. Find and Plot the month wise sales in year 2010 in descending order
- d. Find and Plot most loyal customers based on purchase order
- e. Find and Plot yearly sales comparison
- f. Find and Plot country wise total sales price and show on Geospatial graph
- g. Find and Plot country wise popular product
- h. Find and Plot bottom 10 products based on total sale
- i. Find and Plot top 5 purchase order
- j. Find and Plot most popular products based on sales
- k. Find and Plot half yearly sales for the year 2011
- l. Find and Plot country wise total sales quantity and show on Geospatial graph

OBJECTIVE:

To learn specialized data visualization tools.

To learn different types of data visualization techniques.

SOFTWARE REQUIREMENTS:

1. Ubuntu 16.04 / 18.04

2. Tableau

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**

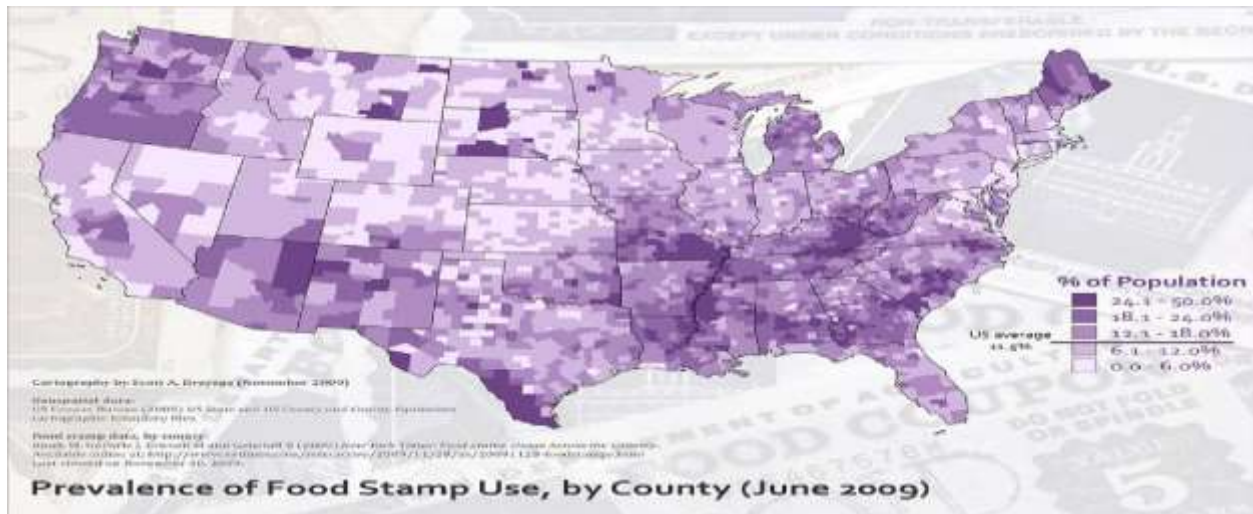


Fig. 1. Geospatial Graph

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 visualization.

Temporal

Examples:

- **Timeline**

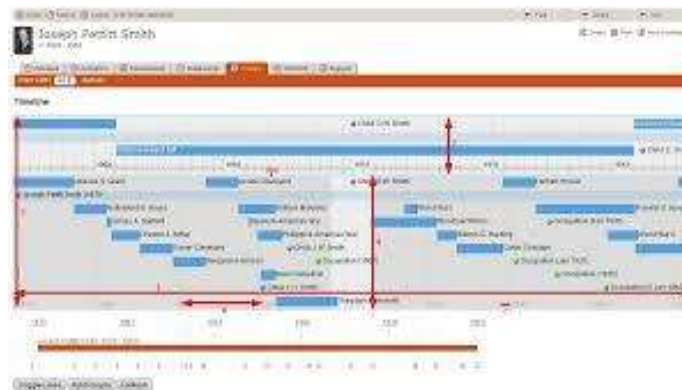


Fig.2 Timeline Graph

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:**



Fig.3Time Series

Multidimensional:

Examples (category proportions, counts):

- **Histogram**

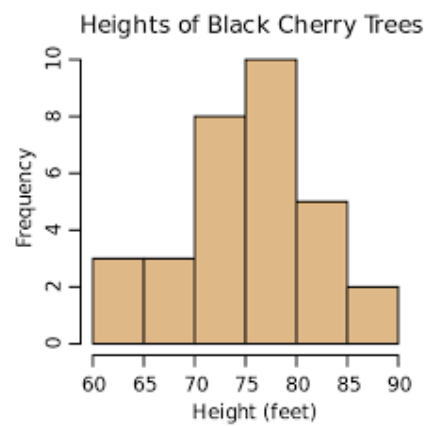


Fig. 4 Histogram

- **pie chart**

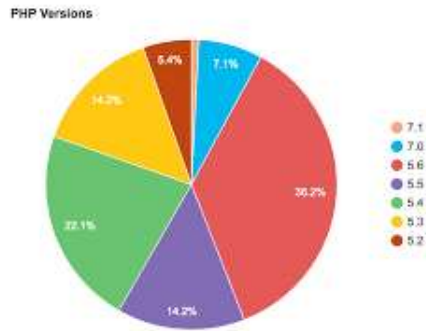


Fig. 5 Pie Chart

Tree/Hierarchical

Examples:

- general tree visualization



Fig. 6 Tree Graph

- Hierarchical Visualization:

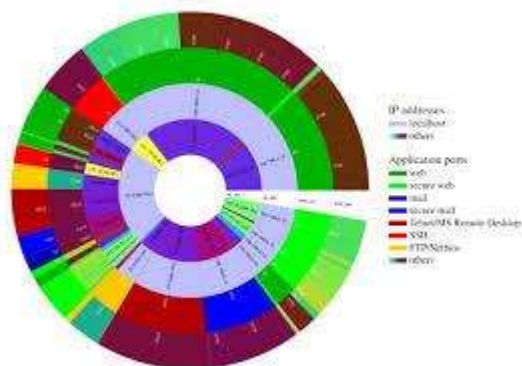


Fig. 7 Hierarchical Graph

- **Dendrogram:**

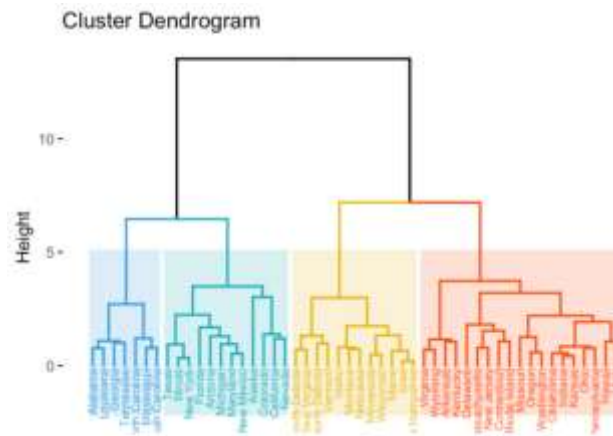


Fig. 8 Dendrogram

- **Network:**

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

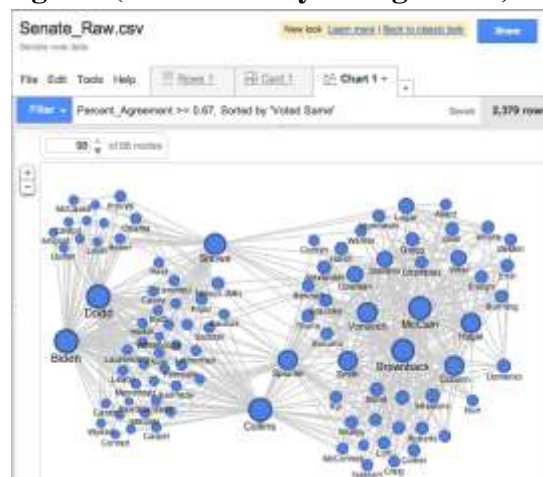


Fig. 9 Network Graph

- **matrix:**

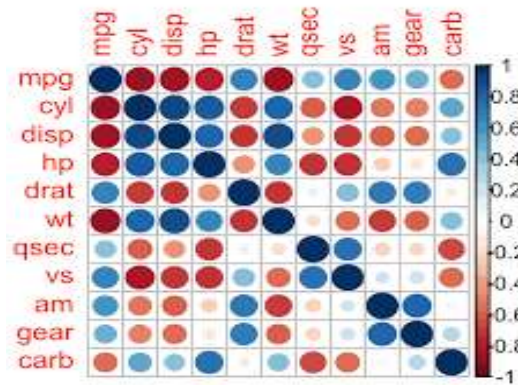


Fig. 10 Matrix Graph

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

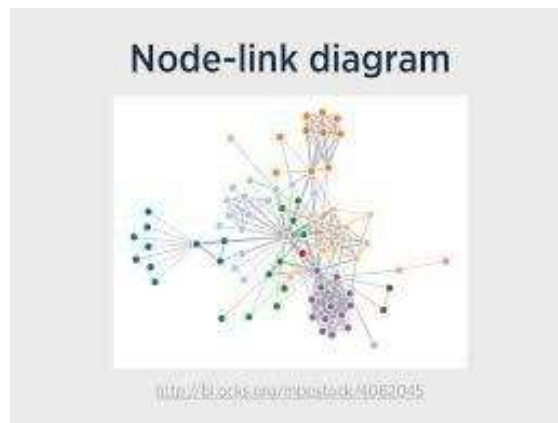


Fig. 11 Node Link Graph

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.

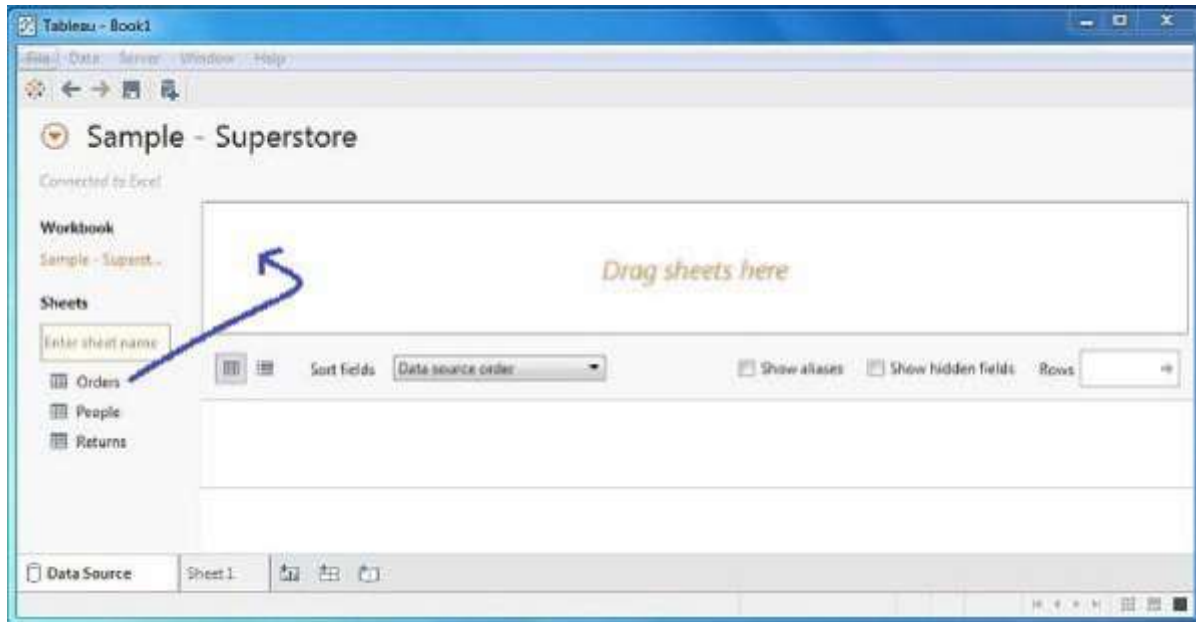
These three steps are –

- **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**.

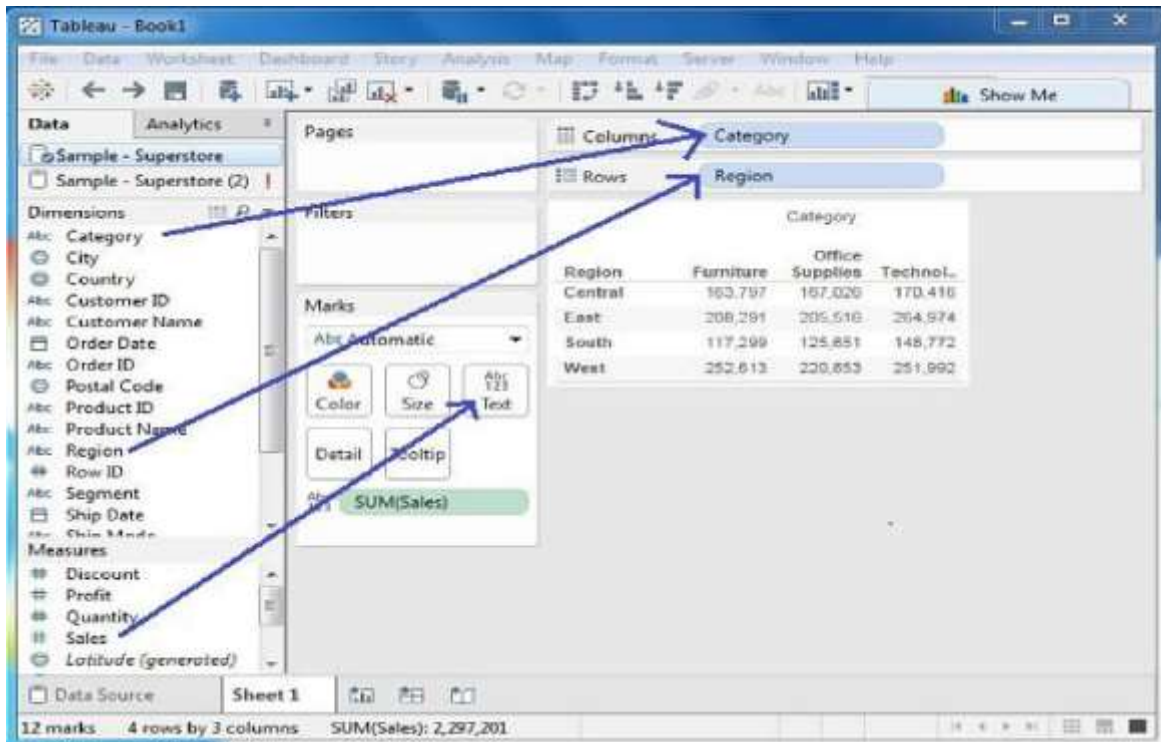
Connect to a Data Source

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

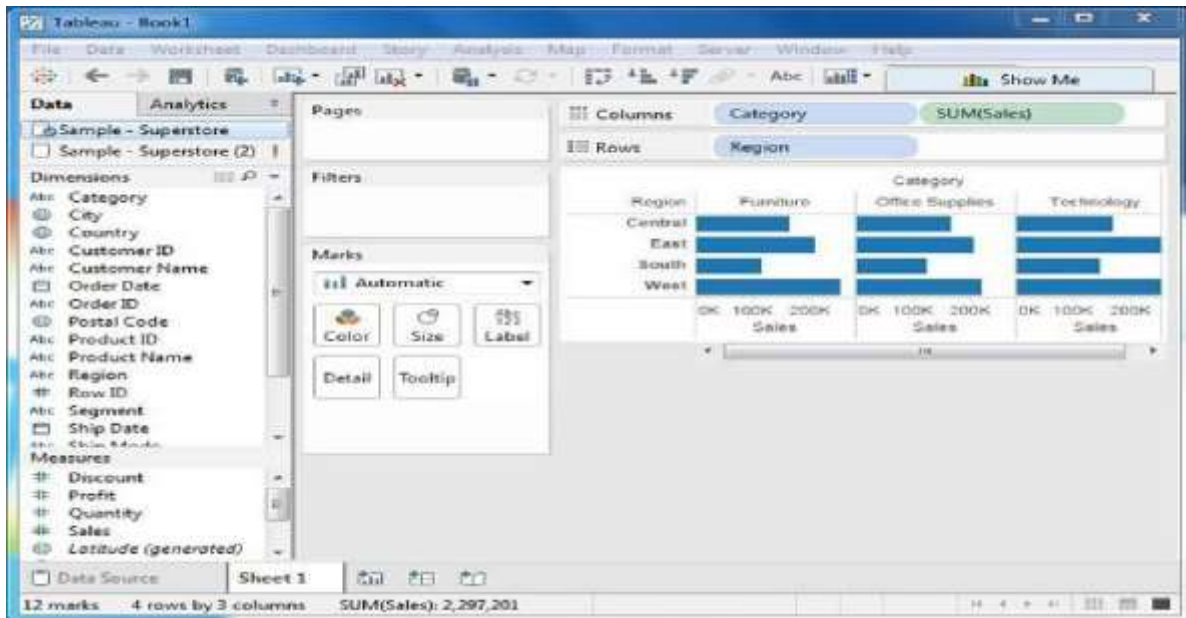
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region**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



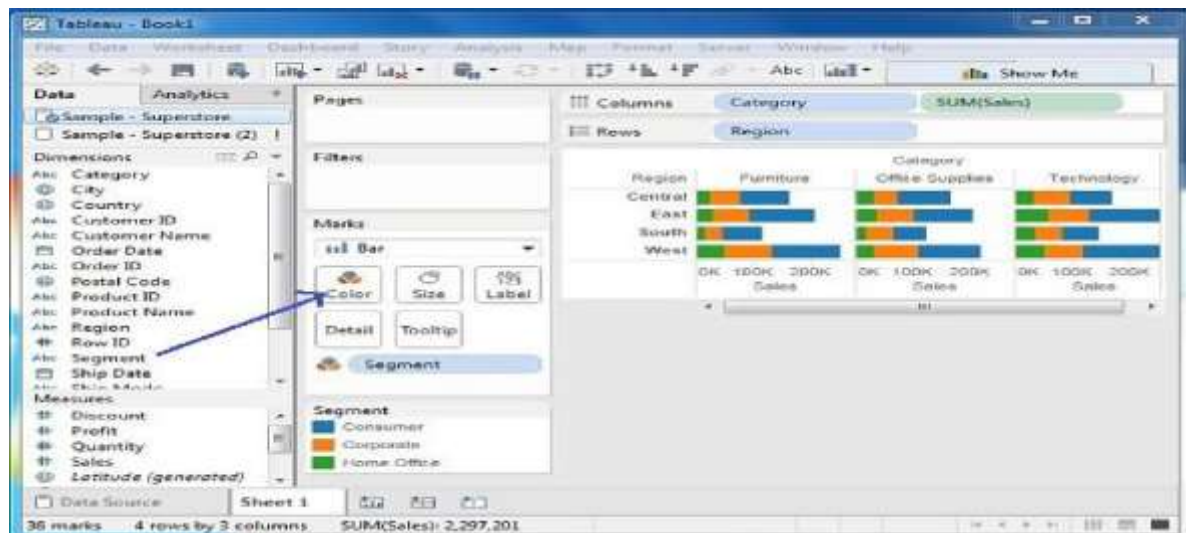
Apply Visualization Technique

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 (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

Model Implementation

Assignment: 9

AIM : To create a review scrapper for any ecommerce website

PROBLEM STATEMENT /DEFINITION

Create a review scrapper for any ecommerce website to fetch real time comments, reviews, ratings, comment tags, customer name using Python

OBJECTIVE

- To understand the concept of Web scraping
- To understand the methods used in web scraping.

THEORY:

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites. The web scraping software may directly access the World Wide Web using the Hypertext Transfer Protocol or a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis. Web scraping a web page involves fetching it and extracting from it. [1]

Beautifulsoup4 library is used here to scrape the data.the required Python libraries:

The request library to make network requests

To scrape data from a website, we need to extract the content of the webpage. Once the request is made to a website, the entire content of the webpage is available, and we can then evaluate the web content to extract data out from it. The content is made available in the form of plain text.

2. Thehtml5lib library for parsing HTML

Once the content is available, we need to specify the library that represents the parsing logic for the text available. We'll be using the html5lib library to parse the text content to HTML DOM-based representation.

3. The beautifulsoup4 library for navigating the HTML tree structure

beautifulsoup4 takes the raw text content and parsing library as the input parameters. In our example, we have exposed html5lib as a parsing library. It can then be used to navigate and search for elements from the

parsed HTML nodes. It can pull data out from the HTML nodes and extract/search required nodes from HTML structure.

Making the Request for the Web Content

Let's make the web request for the website to be scraped. We will be using the requests library. To start using the requests library, we need to install the third-party library using the following command

```
pip install requests
```

We will be scrapping the amazon e-commerce website for customer name, ratings and reviews. Let's first make a request to extract the content for the specified website. request.get makes a request to the webpage, which returns back the raw HTML content.

CODE:

```
from bs4 import BeautifulSoup as bs

import requests

link='https://www.amazon.in/OnePlus-Mirror-Black-128GB-Storage/product-reviews/B07DJHV6VZ/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews'

page = requests.get(link)

page

soup = bs(page.content, 'html.parser')

print(soup.prettify())

names = soup.find_all('span', class_='a-profile-name')

names

[<span class="a-profile-name">Tanmay Shukla</span>,
 <span class="a-profile-name">Surbhi Garg</span>,
 <span class="a-profile-name">Tanmay Shukla</span>,
 <span class="a-profile-name">Surbhi Garg</span>,
 <span class="a-profile-name">Saroj N.</span>,
 <span class="a-profile-name">klknow</span>,
```

```

<span class="a-profile-name">abdulkadir garari</span>,
<span class="a-profile-name">Mani</span>,
<span class="a-profile-name">Anshu K.</span>,
<span class="a-profile-name">Sneha</span>,
<span class="a-profile-name">nagaraj s.</span>,
<span class="a-profile-name">Aakash Sinha</span>]

```

```
#Extracting customer names
```

```
cust_name = []
```

```
for i in range(0,len(names)):
```

```
    cust_name.append(names[i].get_text())
```

```
Cust_name
```

```

['Tanmay Shukla',
 'Surbhi Garg',
 'Tanmay Shukla',
 'Surbhi Garg',
 'Saroj N.',
 'klknow',
 'abdulkadir garari',
 'Mani',
 'Anshu K.',
 'Sneha',
 'nagaraj s.',
 'Aakash Sinha']

```

```
#Extracting Review title.
```

```
title = soup.find_all('a',class_='review-title-content')
```

```
title
```

```

[<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/RG52NAY3E12BR?ASIN=B07DJHV6VZ">
  <span>Flagship Killer</span>

```



```

</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R1EE0UTB5657D6?ASIN=B07DJHV6VZ">
<span>Camera quality is very poor.</span>
</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R15DF987OXR3ES?ASIN=B07DJHV6VZ">
<span>Worst phone</span>
</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R33GOMNOW0H21X?ASIN=B07DJHV6VZ">
<span>Dead on arrival</span>
</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R22HPI22GRD028?ASIN=B07DJHV6VZ">
<span>Not worth to buy 6T</span>
</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R1EJO3TGNM3R9X?ASIN=B07DJHV6VZ">
<span>Battery problem and disappointing customer support.</span>
</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/RP84LWZU2465S?ASIN=B07DJHV6VZ">
<span>Beautiful phone</span>
</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R2LUMMIRPSJBC0?ASIN=B07DJHV6VZ">
<span>Only one side of speakers is working</span>
</a>,

```

```

<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R17C4CZRJA22GP?ASIN=B07DJHV6VZ">
  <span>Awesome value for the money</span>
</a>,
<a class="a-size-base a-link-normal review-title a-color-base review-
title-content a-text-bold" data-hook="review-title" href="/gp/customer-
reviews/R3E54TMP6XTNA4?ASIN=B07DJHV6VZ">
  <span>OnePlust 6T McLaren Edition - Salute to Speed!</span>
</a>]

```

#Similarly all the required fields are scraped and dataframe is formed as given below

```

import pandas as pd

df = pd.DataFrame()

df['Customer Name']=cust_name

df['Review title']=review_title

df['Ratings']=rate

df['Reviews']=review_content

df

```

	Customer Name	Review title	Ratings	Reviews
0	Tanmay Shukla	Flagship Killer	5.0 out of 5 stars	I got this phone on Friday evening.Pros.Great ...
1	Surbhi Garg	Camera quality is very poor.	2.0 out of 5 stars	Camera quality is not upto the mark. I visited...
2	Saroj N.	Worst phone	1.0 out of 5 stars	1. The Battery lasts max of 6 hours.2. Major a...
3	kiknow	Dead on arrival	1.0 out of 5 stars	I charged the phone completely out of the box ...
4	abdulkadir garari	Not worth to buy 6T	1.0 out of 5 stars	One plus 6 was costing 28k during the big bill...
5	Mani	Battery problem and disappointing customer sup...	1.0 out of 5 stars	OnePlus 6T turned out to be an utter disappoin...
6	Anshu K.	Beautiful phone	4.0 out of 5 stars	Good build.amazing battery life. (Minimum 3 d...
7	Sneha	Only one side of speakers is working	1.0 out of 5 stars	Only one side of speakers is working. Right si...
8	nagaraj s.	Awesome value for the money	5.0 out of 5 stars	Phone is simply superb in all aspects...low li...
9	Aakash Sinha	OnePlust 6T McLaren Edition - Salute to Speed!	5.0 out of 5 stars	This has by far been the best phone I have eve...

#dataframe is converted to csv file

```
df.to_csv(r'E:\reviews.csv',index=True)
```

CONCLUSION:

Beautiful Soup was used to scrape customer reviews from amazon Website.

REFERENCES :

[1]Web scraping,Wikipedia

Assignment: 10

AIM: Design and Implement Mini Project in Data Science

PROBLEM STATEMENT /DEFINITION

Design and Implement any Data science Application using R/Python. Obtain Data, Scrub data (Data cleaning), Explore data, Prepare and validate data model and Interpret data (Data Visualization). Visualize data using any visualization tool like Matplotlib, ggplot, Tableau etc. Prepare Project Report.

OBJECTIVE:

1. To explore the Data science project life cycle.
2. To identify need of project and define problem statement.
3. To extract and process data.
4. To interpret and analyze results using data visualization.

Mini Project Report Format:

Abstract

Acknowledgement

List of Tables & Figures

Contents

1. Introduction

1.1 Purpose, Problem statement

1.2 Scope, Objective

1.3 Definition, Acronym, and Abbreviations

1.4 References

2. Literature Survey

2.1 Introduction

2.2 Detail Literature survey

2.4 Findings of Literature survey

3. System Architecture and Design

3.1 Detail Architecture

3.2 Dataset Description

3.3 Detail Phases

3.4 Algorithms

4. Experimentation and Results

4.1 Phase-wise results

4.2 Explanation with example

4.3 Comparison of result with standard

4.4 Accuracy

4.5 Visualization

4.6 Tools used

5. Conclusion and Future scope

5.1 Conclusion

5.2 Future scope

References

Annexure:

- A. GUIs / Screen Snapshot of the System Developed
- B. Implementation /code