DATA SCIENCE AND BIG DATA ANALYTICS LAB

SUBJECT CODE: 314457

CLASS: THIRD YEAR

SEMESTER - VI

LAB MANUAL

DEPARTMENT OF INFORMATION TECHNOLOGY,

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

DEPARTMENT OF INFORMATION TECHNOLOGY

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

DATA SCIENCE & BIG DATA ANALYTICS LAB (314457)

BEIT (2019 Course)

Teaching Scheme :-	Practical : 02 Hrs. / Week		
Examination Scheme:-	Practical 25 Marks	TW 25 Marks	**Credit :-** 01

Prerequisites :

1. Discrete mathematics
2. Database Management Systems, Data warehousing, Data mining
3. Programming in Python

Course Objectives (CO):

1. To understand Big data primitives and fundamentals.
2. To understand the different Big data processing techniques.
3. To understand and apply the Analytical concept of Big data using Python.
4. To understand different data visualization techniques for Big Data.
5. To understand the application and impact of Big Data.
6. To understand emerging trends in Big data analytics.

Course Outcomes (CO's) :

CO1: Apply Big data primitives and fundamentals for application development.

CO2: Explore different Big data processing techniques with use cases.

CO3: Apply the Analytical concept of Big data using Python.

CO4: Visualize the Big Data using Tableau.

CO5: Design algorithms and techniques for Big data analytics.

CO6: Design and develop Big data analytic application for emerging trends.

INDEX

10	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- admissionparameters)	95

REALIZATION OF CO'S AND CSO'S:

Sr. No	TITLE	CO
1	Single node/Multiple node Hadoop Installation.	
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.	
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.	
4	Perform the following operations using Python on the Facebook metrics data sets a. Create data subsets b. Merge Data c. Sort Data d. Transposing Data e. Shape and reshape Data	
5	Perform the following operations using Python on the Air quality and Heart Diseases data sets a. Data cleaning b. Data integration c. Data transformation d. Error correcting e. Data model building	
6	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	
7	Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 2 and 3 (Group B)	
8	Perform the following data visualization operations using Tableau on Adult and Iris datasets. 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	

9	Create a review scrapper for any ecommerce website to fetch real time comments, reviews, ratings, comment tags, customer name using Python.	
10	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-dmissionparameters)	

Reference Books

1. Big Data, Black Book, DT Editorial services, 2015 edition.

2. Data Analytics with Hadoop, Jenny Kim, Benjamin Bengfort, OReilly Media, Inc.

3. Python for Data Analysis by Wes McKinney published by O' Reilly media, ISBN : 978-1-449-31979-3.

4. Python Data Science Handbook by Jake VanderPlas
https://tanthiamhuat.files.wordpress.com/2018/04/pythondatasciencehandbook.pdf

5. Alex Holmes, Hadoop in practice, Dreamtech press.

6. Online References for data set http://archive.ics.uci.edu/ml/
https://www.kaggle.com/tanmoyie/us-graduate-schools-admission-parameters
https://www.kaggle.com

Assignment No 1 (Part A)

Aim :- Hadoop Installation on a) Single Node b) Multiple Node

a) Single Node:

Objective(s)

1. To understand and practice installation and configuration of Hadoop.

Theory:

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.

Prerequisites :

Supported Platforms:

- GNU/Linux is supported as a development and production platform.
- Windows is also a supported platform. Need virtual box with linux operating system. (In case you have an OS other than Linux, you can install a Virtual box software in it and have Linux inside the Virtual box

Required software :

 Java must be installed.

 ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons.

If your cluster doesn't have the requisite software you will need to install it.

*To install software internet connectivity is required. First update System .

> sudo apt-get update

Software installation Steps:

[1] Java Installation: Java is the main prerequisite for Hadoop. First of all, you should verify the existence of java in your system using the command

>"java -version".

If java is not installed in your system, then install java:

> sudo apt-get install default-jdk

create group first

>sudo addgroup hadoop

>sudo adduser --ingroup hadoop hduser

>sudo adduser hduser sudo

[2] Install ssh (secure shell): Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. ssh setup is must for different operation on a cluster like starting, stopping, distributed demon shell operations. This provides secure authentication for different users of hadoop and also provides public/private key pair for Hadoop user and shares it with different users.

ssh has two main components:

1) ssh : The command we use to connect to remote machines - the client.

2) sshd : The daemon that is running on the server and allows clients to connect to the server.

[2.1] Install ssh:

> sudo apt-get install openssh-server

[2.2] If we get something similar to the following, we can think it is setup properly:

```
> which ssh
        /usr/bin/shh
```

Then login as hduser -

>su hduser (login as hduser)

>cd

>pwd

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

>ssh-keygen -t rsa -P "" (then press enter)

[2.4] Add the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

>cat $home/.ssh/id_rsa.pub >> /$home/.ssh/authorized_keys

[2.5] We can check if ssh works:

>ssh localhost

>exit

[3] Download a recent stable release from one of the Apache Download Mirrors (http://hadoop.apache.org)

[3.1] Download: (keep the downloaded file on desktop)

```
> cd /usr/local
> wget http://www.eu.apache.org/dist/hadoop/common/stable/hadoop-2.9.0.tar.gz
```

[3.2] copy and extract hadoop-2.9.0.tar.gz in home folder

>pwd

>cd /home/rajendra/Desktop/

> tar -xvzf hadoop-2.9.0.tar.gz

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

```
> mv  hadoop-2.9.0  /usr/local/hadoop
>ls
```

[3.3]Give ownership to hadoop user

> sudo chown –R hduser /usr/local

[4] Hadoop Configuration:

There are three modes in which you can start Hadoop cluster:

1) Local (Standalone)Mode

2) Pseudo-Distributed Mode

3) Fully Distributed Mode

Hadoop is configured by default to run in a non-distributed mode, asa single Java process, Which is useful for debugging. Hadoop deamon runs in a separate Java process in single-node in a psudo-distributed mode. **Fully distributed mode runs with two or more machines in a cluster**.

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

1) ~/.bashrc

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

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

4) /usr/local/hadoop/etc/hadoop/mapred-site.xml.template

5) /usr/local/hadoop/etc/hadoop/hdfs-site.xml

Prepare to Start the Hadoop Cluster:

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

> readlink -f /usr/bin/javac

On 64 bit machine , the path might be /usr/lib/jvm/java-8-openjdk-amd64/bin/javac

On 32 bit machine /usr/lib/jvm/java-8-openjdk-i386/bin/javac

[4.1.1] **~/.bashrc**: Edit ~/.bashrc file and add Java path, hadoop path

*Note that the JAVA_HOME should be set as the path just before the '.../bin/'

*Note : See the configuration file

>sudo nano ~/.bashrc

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME =/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME /bin
export PATH=$PATH:$HADOOP_HOME / sbin
export HADOOP_MAPRED_HOME = $HADOOP_HOME
export HADOOP_COMMON_HOME =$HADOOP_HOME
export HADOOP_HDFS_HOME = $HADOOP_HOME
export YARN_HOME=$ HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_LOCAL/lib"
```

[4.2] We need to set JAVA_HOME by modifying hadoop-env.sh file.

Edit the file etc/hadoop/hadoop-env.sh to define JAVA_HOME parameter as follows:

>sudo gedit usr/local/hadoop/etc/hadoop/hadoop-env.sh

Try the following command:

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

Adding the above statement in the hadoop-env.sh file ensures that the value of JAVA_HOME variable will be available to Hadoop whenever it is started up.

[4.3] Edit core-site.xml

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

Create a temporary directory to store App-data and give rights to hadoop user 'hduser' to make changes into it

sudo gedit */usr/local/hadoop/etc/hadoop/core-site.xml*

Open the core-site.xml file and enter the code in between the <configuration></configuration> tag: (see configuration file)

[4.4] hdfs-site.xml:

sudo gedit */usr/local/hadoop/etc/hadoop/core-site.xml (*(see configuration file)

[4.5] YARN on a Single Node:

You can run a MapReduce job on YARN in a pseudo-distributed mode by setting a few parameters and running Resource Manager daemon and Node Manager daemon in addition.

sudo gedit */usr/local/hadoop/etc/hadoop/yarn-site.xml (*(see configuration file)

[4.6] Edit mapred-site.xml

By default, the /usr/local/hadoop/etc/hadoop/ folder contains mapred-site.xml.template file which has to be renamed/copied with the name mapred-site.xml:

sudo gedit */usr/local/hadoop/etc/hadoop/mapred-site.xml (*(see configuration file)

```
> sudo cp  /usr/local/hadoop/etc/hadoop/mapred-site.xml.template  /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

```
sudo mkdir -p /usr/local/hadoop_tmp
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
assign ownership to hduser
sudo chown -R hduser /usr/local/hadoop_tmp
```

[5] Format Namenode

```
>cd
>hdfs namenode -format
>start-dfs.sh
>start-yarn.sh
```

[6] Check all process are started

```
>jps
9273 Nodemanager
21603 Namenode
21787 SecondaryNamenode
29728 Datanode
```

```
29567 ResourceManager
21922 jps
```

Word Count Program

```
cd /home/rajendra/Desktop/
sudo mkdir data
cd data
sudo gedit sample.txt
cd /usr/local/hadoop
bin/hdfs dfs -mkdir /user
bin/hdfs dfs -mkdir /user/ypm
bin/hdfs dfs -put /home/rajendra/Desktop/data /user/input
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-example-2.9.0.jar wordcount /user/input output
bin/hdfs dfs -cat output/*
for user interface
localhost:50070
stop-all.sh
```

b) Multiple Node

Install a Multi Node Hadoop Cluster on Ubuntu 14.04

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
# Updata 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.bindv6only\ =\ 1/net.ipv6.bindv6only\ =\ 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. Lets'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, 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 the 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, lets 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 30 100
```

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.

Conclusion: We have studied Hadoop installation on single node & Multi node & Hadoop configured on Ubuntu.

Assignment No 2 (Part A)

Aim :- 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:

Students should be able to understand:-

1. MapReduce
2. Working of MapReduce

THEORY:

What is MapReduce?

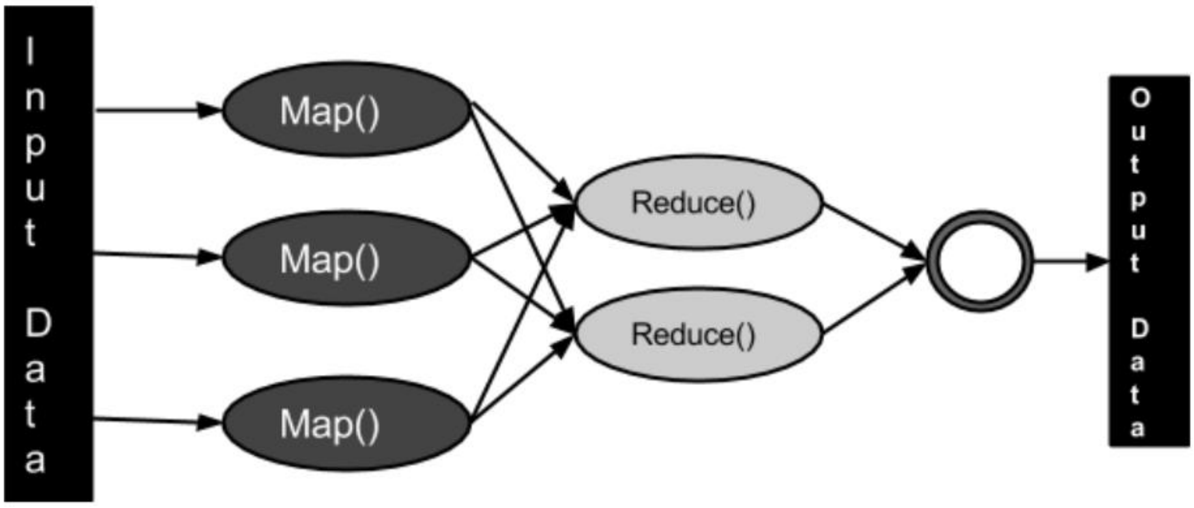
MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job. The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

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

• 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.



Inserting Data into HDFS

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

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework.

Input and Output types of a MapReduce job: (Input) <k1,v1> -> map -> <k2, v2>-> reduce -> <k3, v3> (Output).

	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.

IMPLEMENTATION:-

Cyber.java

```
/* 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.

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 Cyber
{
        //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) throws IOException
        {
                String line = value.toString();
                String lasttoken = null;
                StringTokenizer s = new StringTokenizer(line,"\t");
                String name = s.nextToken();
                while(s.hasMoreTokens())
                        {
                        lasttoken += (Integer.parseInt(s.nextToken()); //add all the elements
                        }
        int avgtime = lasttoken/7;  // calculate average
        output.collect(new Text(name), new IntWritable(avgtime));
        }
}
//Reducer class
public static class E_EReduce extends MapReduceBase implements
Reducer< Text, IntWritable, Text, IntWritable >
{
        //Reduce function
        public void reduce(
        Text key,
        Iterator <IntWritable> values,
        OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException
        {
                int avg=5;              //threshold value for max time internet usage
                int val=0;
                while (values.hasNext())
                {
                        if((val=values.next().get())>avg)
                        {
                                output.collect(key, new IntWritable(val));
                        }
                }
        }
}
//Main function
public static void main(String args[])throws Exception
{
        JobConf conf = new JobConf(Cyber.class);
        conf.setJobName("Internet Log");
        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, new Path(args[0]));
                FileOutputFormat.setOutputPath(conf, new Path(args[1]));
                JobClient.runJob(conf);
        }
}
```

Sample.txt

```
Rahul     2    3    4    5    2    0    8
Tanmay    3    0    0    0    3    6    6
Ajay      5    6    1    1    4    2    1
Akshat    2    2    2    2    2    4    2
Rohan     10   8    7    8    10   1    3
Srenik    5    5    5    6    2    1    1
Anahat    3    5    2    10   1    10   5
Ajita     3    2    2    4    3    4    6
Raghav         8    3    1    0    7    8    8
Pramod    0    0    0    3    1    2    2
Arjun     5    3    4    2    3    2    4
Pratap    9    8    7    6    7    8    10
Nitin     8    5    4    3    4    3    4
Vikas     1    2    8    6    5    4    4
Kunda     9    6    7    8    9    8    11
Amod      6    7    4    5    6    6    7
Siddhi    1    2    0    0    0    4    5
Ashok     9    7    5    6    7    5    5
Heena     9    6    7    5    4    5    7
Raman     6    7    8    8    8    9    9
```

Compilation and Execution:-
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 internet

• 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 Cyber.java program and creating a jar for the program.

```
$ javac -classpath hadoop-core-1.2.1.jar internet/Cyber.java
$ jar -cvf cyber.jar -C internet/ .
```

Step 4

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

```
$hadoop fs -mkdir input_dir
```

• **Step 5**

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

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

• **Step 6**

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

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

Step 7

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

```
$hadoop jar internet.jar Cyber input_dir/output_dir/
```

Wait for a while until the file is executed. After execution, the output will contain the number of input splits, the number of Map tasks, the number of reducer tasks, etc. The output directory must not be existing already.

Step 8

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

```
$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 fs -cat output_dir/part-00000
```

- **Step 10**

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

```
$hadoop fs -get output_dir/part-00000
```

CONLCUSION: In this assignment, we have studied to design a distributed application using MapReduce which processes a log file of a system.

FAQ:

- What is Hadoop Map Reduce ?
- How Hadoop MapReduce works?
- Explain what is shuffling in MapReduce ?
- What is NameNode in Hadoop?

ASSIGNMENT NO. 3 (Part A)

TITLE: HIVEQL

AIM:

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:

Students should be able to understand:-

1. HiveQL

PRE-REQUISITES:-

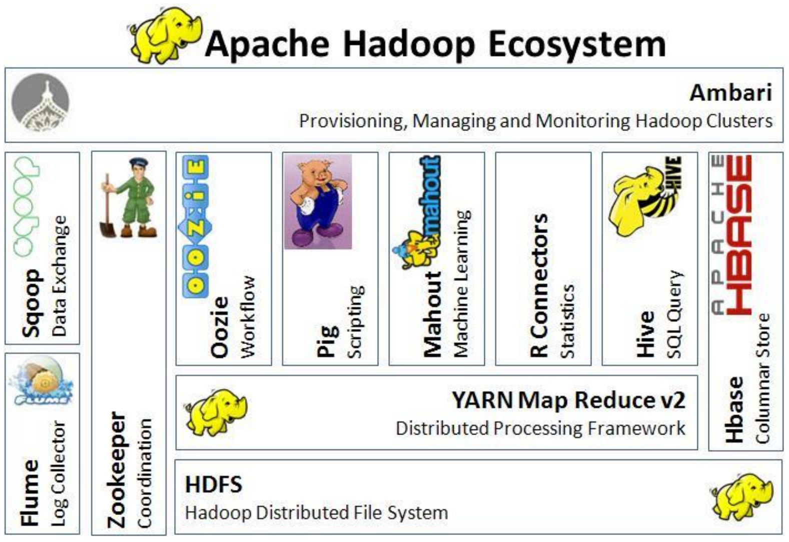
• Ubuntu 14.04 LTS
• Hadoop 2.6.*
• Java with jdk1.7 onwards

THEORY:

Hadoop Ecosystem:-

The Hadoop ecosystem contains different subprojects (tools) such as Sqoop, Pig, and Hive that are used to help Hadoop modules.

- Sqoop: It is used to import and export data to and fro between HDFS and RDBMS.
- Pig: It is a procedural language platform used to develop a script for MapReduce operations.
- Hive: It is a platform used to develop SQL type scripts to do MapReduce operations.



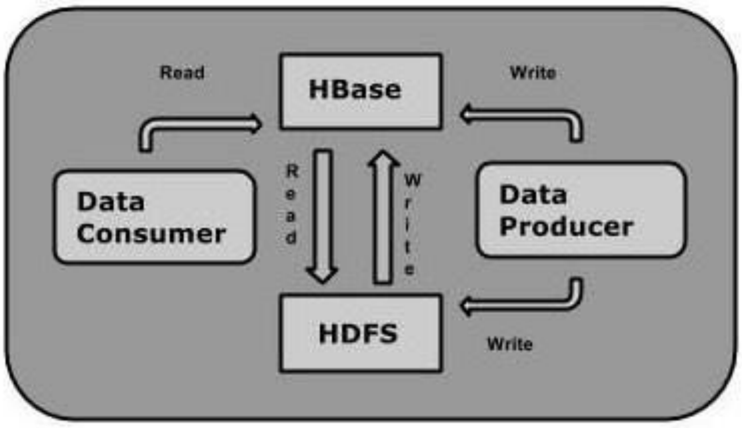
What is HBase?

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable.

HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS).

It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access.



Hive:-

Hive:

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

- **Prerequisites:** Core Java,

  Database concepts of SQL,

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

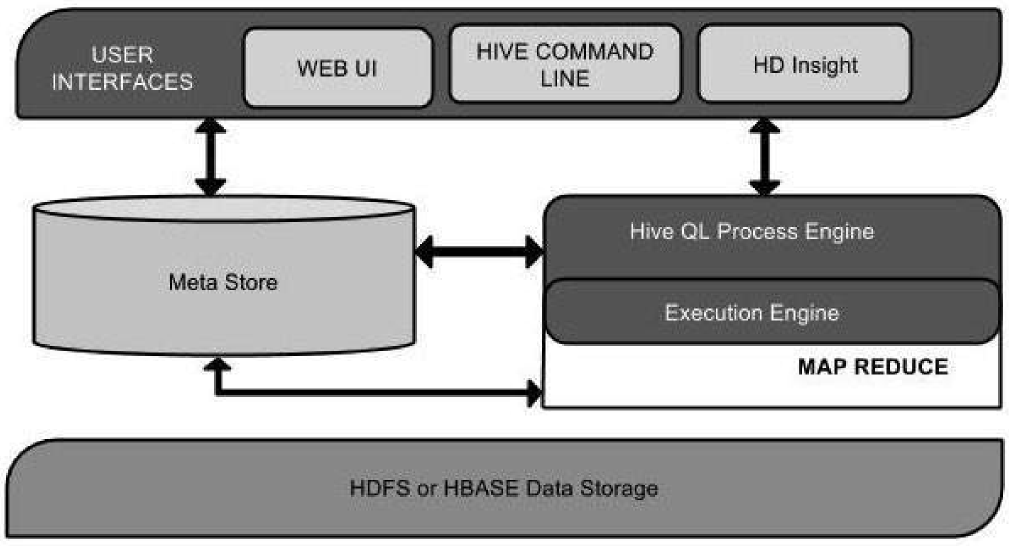


Figure 1: Hive Architecture

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

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

	MapReduce results. It uses the flavor of MapReduce.
HDFS or HBASE	Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

Table 1: Hive components and their operations.

Download and Copy:-
Download hive on below path (nearly 93 MB):

```
http://www.apache.org/dyn/closer.cgi/hive/
```

Extract the .tar.gz file in Downloads/ and rename it to hive/ and move the folder to /usr/lib/ path:

```
sudo mv Downloads/hive /usr/lib
```

Change the owner
Provide access to hive path by changing the owners and groups to hduser and hadoop respectively.

```
sudo chown -R hduser:hadoop /usr/lib/hive
```

Configure environment variables
Configure environment variables in .bashrc file.

```
su - hduser
vim ~/.bashrc
```

Add following lines at the end of file

```
export HIVE_HOME=/usr/lib/hive/
export PATH=$PATH:$HIVE_HOME/bin
export HADOOP_USER_CLASSPATH_FIRST=true
```

Apply the changes:

```
source ~/.bashrc
```

Make directories
Create temporary and folder for data warehouse of hive in HDFS as well as change the permissions.

```
hadoop fs -mkdir /tmp
hadoop fs -mkdir -p /user/hive/warehouse
hadoop fs -chmod g+w /tmp
hadoop fs -chmod -R g+w /user/hive/warehouse
```

Configure Hive
To configure Hive with Hadoop, you need to edit the hiveenv. sh file, which is placed in the $HIVE_HOME/conf directory. The following commands redirect to Hive config folder and copy the template file:

```
cd $HIVE_HOME/conf
cp hive-env.sh.template hive-env.sh
```

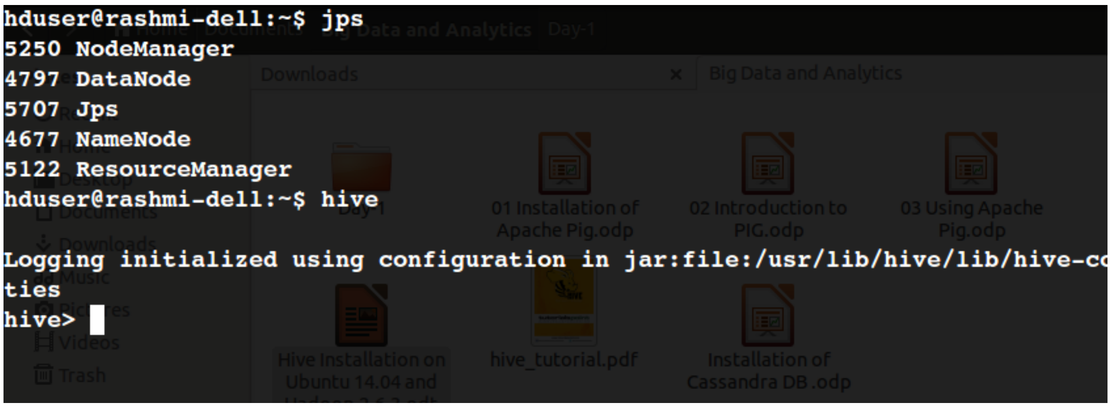
Edit the hive-env.sh file by appending the following line:

export HADOOP_HOME=/usr/local/hadoop

Run the Hive

Make sure that Hadoop services are running. Then type

Hive



```
hduser@rashmi-dell:~$ jps
5250 NodeManager
4797 DataNode
5707 Jps
4677 NameNode
5122 ResourceManager
hduser@rashmi-dell:~$ hive

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-co
ties
hive>
```

Create Database:-

Create Database is a statement used to create a database in Hive.
A database in Hive is a namespace or a collection of tables. The syntax for this statement is as follows:
CREATE DATABASE|SCHEMA [IF NOT EXISTS]
<database name>;
Here, IF NOT EXISTS is an optional clause, which notifies the user that a database with the same name already exists. We can use SCHEMA in place of DATABASE in this command.
The following query is executed to create a database named
mydb:

```
hive> CREATE DATABASE [IF NOT EXISTS] mydb;
        or
hive> CREATE SCHEMA mydb;
```

The following query is used to verify a databases list:

```
hive> SHOW DATABASES;
        default
        mydb
```

Drop Database

Drop Database is a statement that drops all the tables and deletes the database.
Its syntax is as follows:

```
DROP DATABASE StatementDROP
        (DATABASE|SCHEMA) [IF EXISTS]
        database_name [RESTRICT|CASCADE];
```

The following queries are used to drop a database. Let us assume that the database name is mydb.

```
hive> DROP DATABASE IF EXISTS mydb;
```

The following query drops the database using CASCADE. It means dropping respective tables before dropping the database.

```
hive> DROP DATABASE IF EXISTS userdb CASCADE;
```

The following query drops the database using SCHEMA.

```
hive> DROP SCHEMA userdb;
```

This clause was added in Hive 0.6.

Create Table

Create Table is a statement used to create a table in Hive. The syntax and example are as follows:
Syntax:

```
        CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]
table_name
                        [(col_name data_type [COMMENT col_comment], ...)]
                        [COMMENT table_comment]
                        [ROW FORMAT row_format]
                        [STORED AS file_format]
```

Create Table : Example

Sr. No	Field Name	Data type
1	Eid	Int
2	Name	String
3	Salary	Float
4	Designation	String

The following query creates a table named employee using the above data.

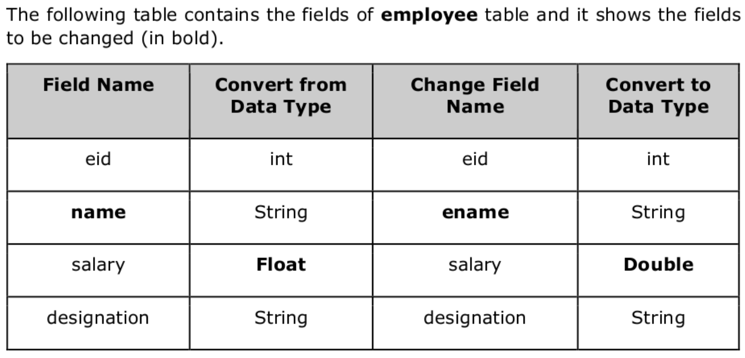
```
hive> CREATE TABLE IF NOT EXISTS
employee ( eid int, name String,
> salary String, destination String)
> COMMENT ‘Employee details’
> ROW FORMAT DELIMITED
> FIELDS TERMINATED BY ‘\t’
> LINES TERMINATED BY ‘\n’
> STORED AS TEXTFILE;
```

Alter Table

```
ALTER TABLE name RENAME TO new_name
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
ALTER TABLE name DROP [COLUMN] column_name
ALTER TABLE name CHANGE column_name new_name new_type
ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])
ALTER TABLE employee RENAME TO emp;

ALTER TABLE employee RENAME TO emp;
```

Change statement



The following table contains the fields of **employee** table and it shows the fields to be changed (in bold).

Field Name	Convert from Data Type	Change Field Name	Convert to Data Type
eid	int	eid	int
name	String	**ename**	String
salary	**Float**	salary	**Double**
designation	String	designation	String

```
hive> ALTER TABLE employee CHANGE name ename String;
hive> ALTER TABLE employee CHANGE salary salary Double;
```

Add column statement

```
hive> ALTER TABLE employee ADD COLUMNS
                        ( dept STRING COMMENT 'Department name');
```

Replace statement

```
hive> ALTER TABLE employee REPLACE COLUMNS
        ( eid INT empid Int, ename STRING name String);
```

Drop table statement

The syntax is as follows:

```
DROP TABLE [IF EXISTS] table_name;
```

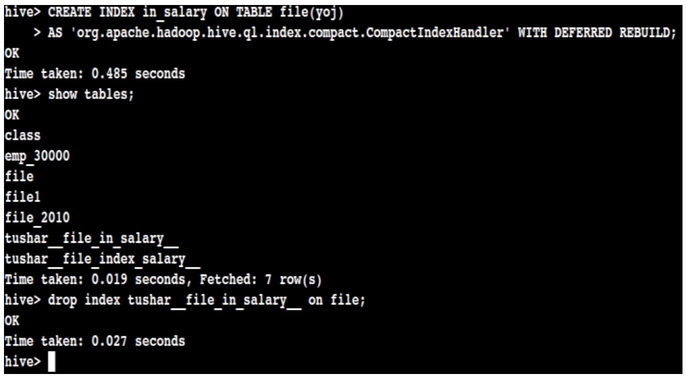
The following query drops a table named employee:

```
hive> DROP TABLE IF EXISTS employee;
```

Index

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

```
hive> CREATE INDEX index_yoj ON TABLE file(yoj)
> AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler'
WITH DEFERRED REBUILD;
```



```
hive> CREATE INDEX in_salary ON TABLE file(yoj)
    > AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;
OK
Time taken: 0.485 seconds
hive> show tables;
OK
class
emp_30000
file
file1
file_2010
tushar__file_in_salary__
tushar__file_index_salary__
Time taken: 0.019 seconds, Fetched: 7 row(s)
hive> drop index tushar__file_in_salary__ on file;
OK
Time taken: 0.027 seconds
hive> 
```

Drop index

The following syntax is used to drop an index:

```
DROP INDEX <index_name> ON <table_name>
```

The following query drops an index named index_salary:

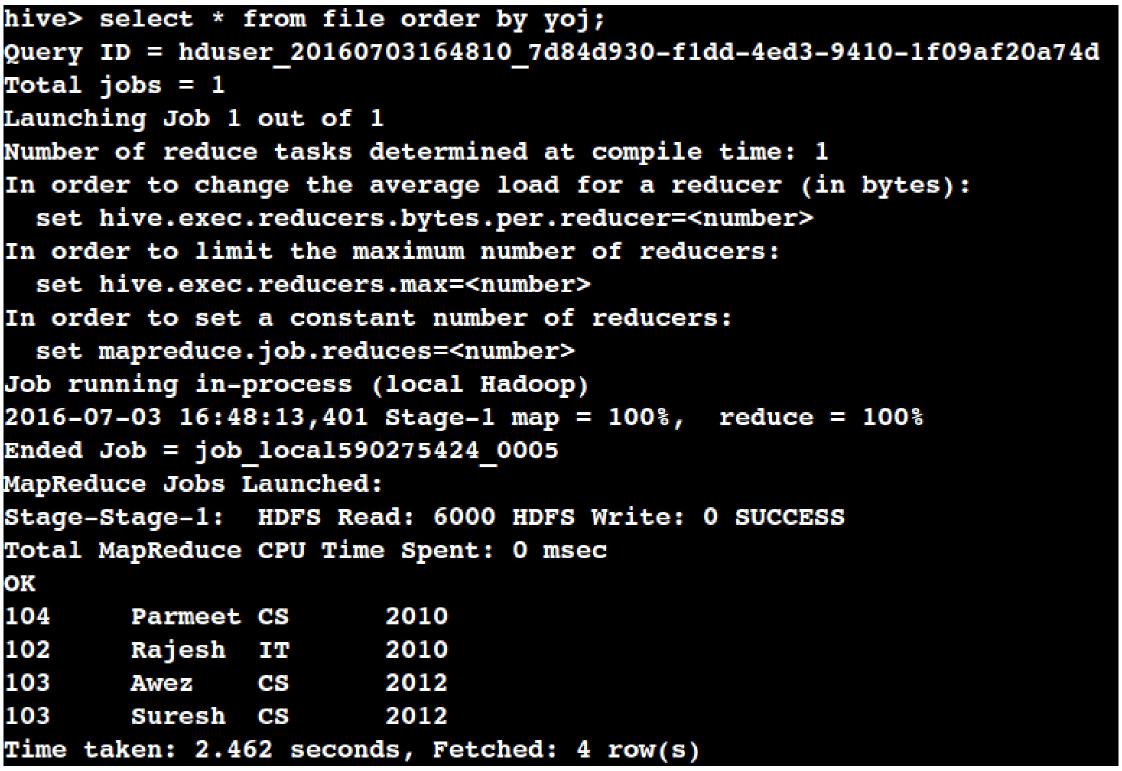
```
hive> DROP INDEX index_salary ON employee;
```

Select … order by

The ORDER BY clause is used to retrieve the details based on one column and sort the result set by ascending or descending order.

Syntax:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list]]
[LIMIT number];
```



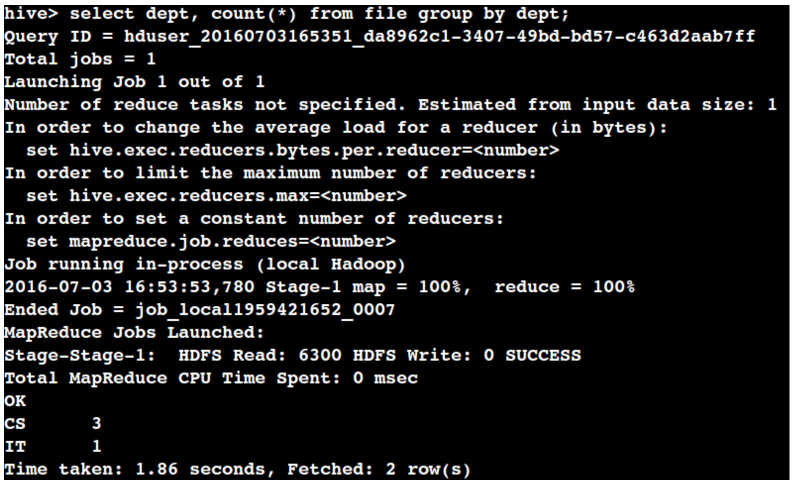
```
hive> select * from file order by yoj;
Query ID = hduser_20160703164810_7d84d930-f1dd-4ed3-9410-1f09af20a74d
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2016-07-03 16:48:13,401 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local590275424_0005
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 6000 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
104     Parmeet CS      2010
102     Rajesh  IT      2010
103     Awez    CS      2012
103     Suresh  CS      2012
Time taken: 2.462 seconds, Fetched: 4 row(s)
```

Select… group by

The GROUP BY clause is used to group all the records in a result set using a particular collection column. It is used to query a group of records.

Syntax:

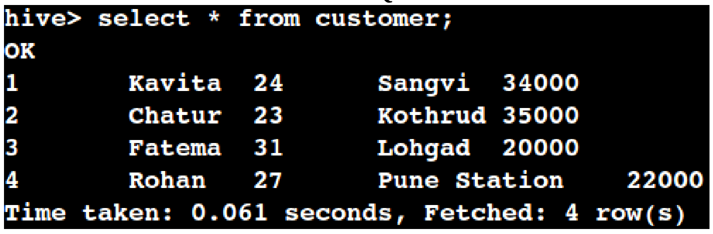
```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list]]
[LIMIT number];
```



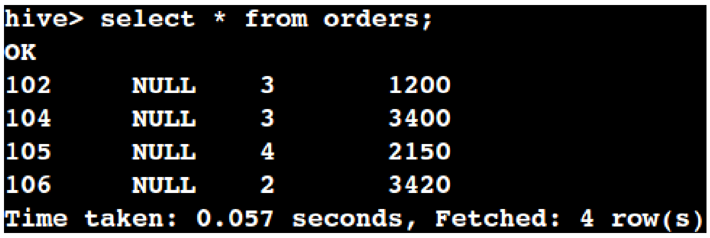
```
hive> select dept, count(*) from file group by dept;
Query ID = hduser_20160703165351_da8962c1-3407-49bd-bd57-c463d2aab7ff
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  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>
Job running in-process (local Hadoop)
2016-07-03 16:53:53,780 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1959421652_0007
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 6300 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
CS      3
IT      1
Time taken: 1.86 seconds, Fetched: 2 row(s)
```

Joins

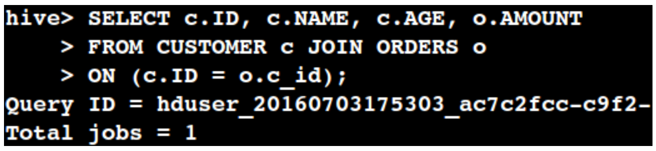
JOINS is a clause that is used for combining specific fields from two tables by using values common to each one.
It is used to combine records from two or more tables in the database.
It is more or less similar to SQL JOINS.



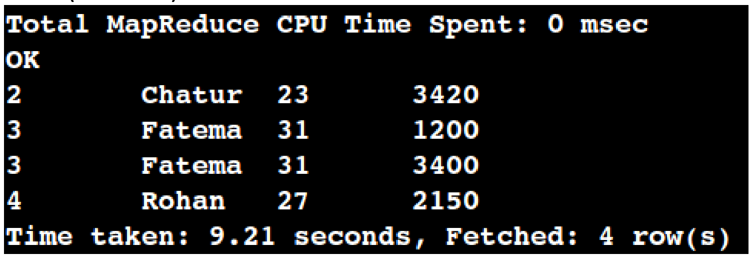
```
hive> select * from customer;
OK
1       Kavita  24      Sangvi  34000
2       Chatur  23      Kothrud 35000
3       Fatema  31      Lohgad  20000
4       Rohan   27      Pune Station    22000
Time taken: 0.061 seconds, Fetched: 4 row(s)
```



```
hive> select * from orders;
OK
102     NULL    3       1200
104     NULL    3       3400
105     NULL    4       2150
106     NULL    2       3420
Time taken: 0.057 seconds, Fetched: 4 row(s)
```



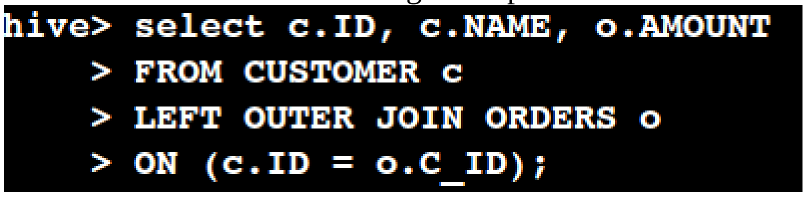
```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT
    > FROM CUSTOMER c JOIN ORDERS o
    > ON (c.ID = o.c_id);
Query ID = hduser_20160703175303_ac7c2fcc-c9f2-
Total jobs = 1
```



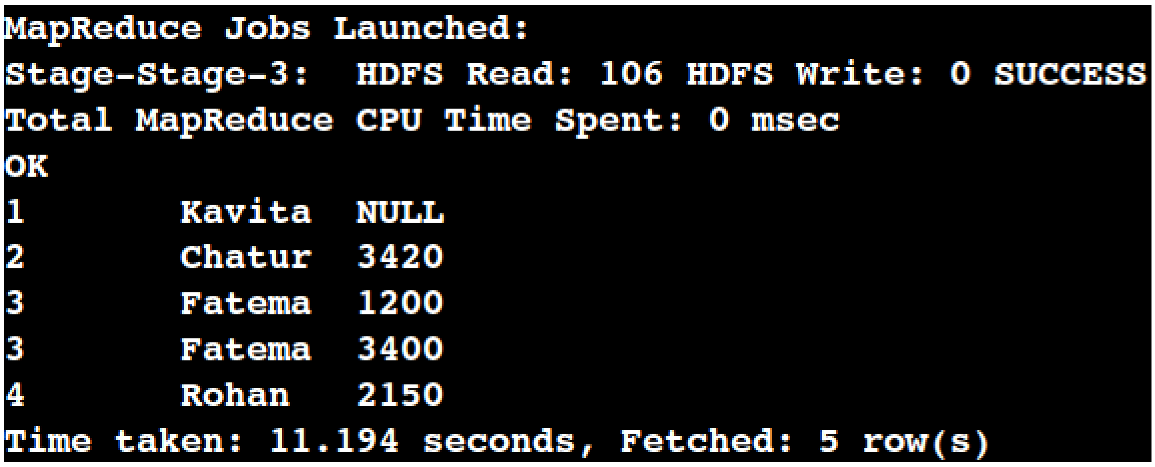
```
Total MapReduce CPU Time Spent: 0 msec
OK
2       Chatur  23      3420
3       Fatema  31      1200
3       Fatema  31      3400
4       Rohan   27      2150
Time taken: 9.21 seconds, Fetched: 4 row(s)
```

Left outer join

The HiveQL LEFT OUTER JOIN returns all the rows from the left table, even if there are no matches in the right table. This means, if the ON clause matches 0 (zero) records in the right table, the JOIN still returns a row in the result, but with NULL in each column from the right table. A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate.



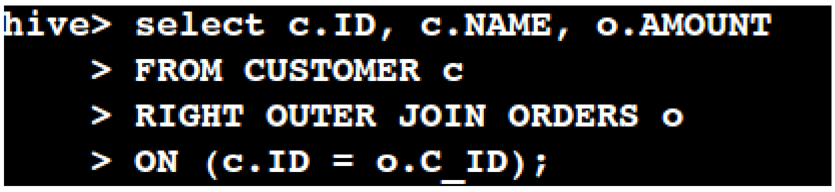
```
hive> select c.ID, c.NAME, o.AMOUNT
    > FROM CUSTOMER c
    > LEFT OUTER JOIN ORDERS o
    > ON (c.ID = o.C_ID);
```



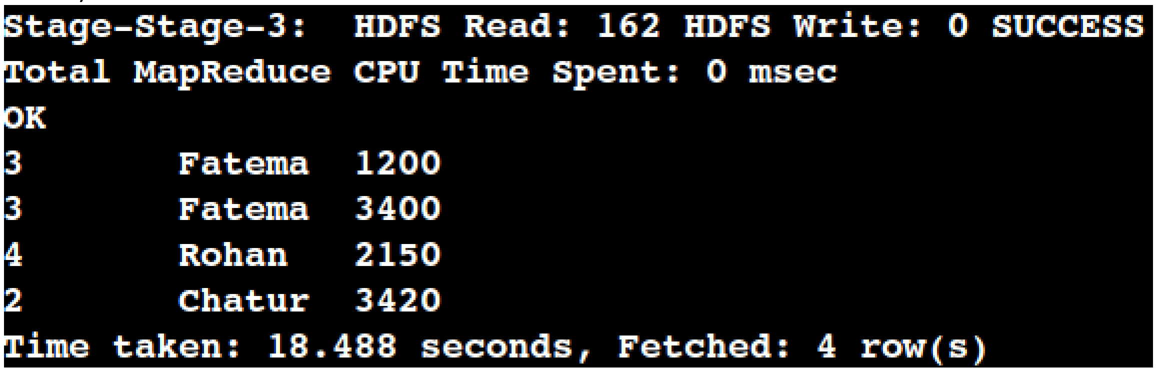
```
MapReduce Jobs Launched:
Stage-Stage-3:  HDFS Read: 106 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
1       Kavita  NULL
2       Chatur  3420
3       Fatema  1200
3       Fatema  3400
4       Rohan   2150
Time taken: 11.194 seconds, Fetched: 5 row(s)
```

Right outer join

The HiveQL RIGHT OUTER JOIN returns all the rows from the right table, even if there are no matches in the left table. If the ON clause matches 0 (zero) records in the left table, the JOIN still returns a row in the result, but with NULL in each column from the left table. A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate.



```
hive> select c.ID, c.NAME, o.AMOUNT
    > FROM CUSTOMER c
    > RIGHT OUTER JOIN ORDERS o
    > ON (c.ID = o.C_ID);
```



```
Stage-Stage-3:  HDFS Read: 162 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
3       Fatema  1200
3       Fatema  3400
4       Rohan   2150
2       Chatur  3420
Time taken: 18.488 seconds, Fetched: 4 row(s)
```

IMPLEMENTATION:-
(students should attach code here)

CONLCUSION: In this assignment, we have studied to design an application using HiveQL.

Assignment 4 (Part-B)

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

a. Create data subsets

b. Merge Data

c. Sort Data

d. Transposing Data

e. Shape and reshape Data

Objective : Student should be able to do : Create data subsets, Merge Data , Sort Data , Transposing Data , Shape and reshape Data

Theory:

a. Create data subsets

How to Subset a DataFrame in Python?

If you are importing data into Python then you must be aware of Data Frames. A DataFrame is a **two-dimensional data structure**, i.e., data is aligned in a tabular fashion in rows and columns.Subsetting a data frame is the process of **selecting a set of desired rows and columns from the data frame.**

You can select:

- all rows and limited columns
- all columns and limited rows
- limited rows and limited columns.

Subsetting a data frame is important as it allows you to access only a certain part of the data frame. This comes in handy when you want to reduce the number of parameters in your data frame.

Importing the Data to Build the Dataframe

Let's start with importing the data into a data frame using pandas.

Import pandas as pd

df = pd.read_csv("D:/dataset_Facebook.csv",sep = ';')

Select a Subset of a Dataframe using the Indexing Operator

Indexing Operator is just a fancy name for **square brackets.** You can select columns, rows, and a combination of rows and columns using just the square brackets. Let's see this in action.

1. Selecting Only Columns

To select a column using indexing operator use the following line of code.

print(df['Type'])

This line of code selects the column with label as 'Type' and displays all row values corresponding to that.

You can also select multiple columns using indexing operator.

print(df[['like','share']])

To subset a dataframe and store it, use the following line of code :

df_subset = df[['like','share']]

This creates a separate data frame as a subset of the original one.

2. Selecting Rows

You can use the indexing operator to select specific rows based on certain conditions.

For example to select rows having likes greater than 100 you can use the following line of code.

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

You can also further subset a data frame.

Subset a Dataframe using Python .loc()

We can select specific ranges of our data in both the row and column directions using either label or integer-based indexing.

- loc is primarily *label* based indexing. *Integers* may be used but they are interpreted as a *label*.
- iloc is primarily *integer* based indexing

.loc indexer is an effective way to select rows and columns from the data frame. It can also be used to select rows and columns simultaneously.

An important thing to remember is that **.loc() works on the labels of rows and columns.** After this, we will look at .iloc() that is based on an index of rows and columns.

1. Selecting Rows with loc()

To select a single row using .loc() use the following line of code.

print(df.loc[1])

To select multiple rows use :

print(df.loc[[1,5,7]])

You can also slice the rows between a starting index and ending index.

df.loc[1:7]

2. Selecting rows and columns

To select specific rows and specific columns out of the data frame, use the following line of code :

df.loc[1:7,['Type', 'likes']]

This line of code selects rows from 1 to 7 and columns corresponding to the labels ‘Type’ and ‘likes’.

Subset a Dataframe using Python iloc()

iloc() function is short for **integer location**. It works entirely on integer indexing for both rows and columns.

To select a subset of rows and columns using iloc() use the following line of code:

df.iloc[[2,3,6], [3, 5]]

This line of code selects row number **2, 3 and 6** along with column number **3 and 5.**

Using iloc saves you from writing the complete labels of rows and columns.

You can also use iloc() to select rows or columns individually just like loc() after replacing the labels with integers.

REMEMBER

- When selecting subsets of data, square brackets [] are used.

- Inside these brackets, you can use a single column/row label, a list of column/row labels, a slice of labels, a conditional expression or a colon.
- Select specific rows and/or columns using loc when using the row and column names
- Select specific rows and/or columns using iloc when using the positions in the table
- You can assign new values to a selection based on loc/iloc.

Python Syntax

You can use the syntax below when querying data by criteria from a DataFrame. Experiment with selecting various subsets of the "rainfall" data.

- Equals: ==
- Not equals: !=
- Greater than, less than: > or <
- Greater than or equal to >=
- Less than or equal to <=

Missing Data Values - NaN

By now you probably wondered about the NaNs in the data, e.g. in the data column. NaN stands for **N**ot **aN**umber. NaN values are undefined values that cannot be represented mathematically. Pandas, for example, will read an empty cell in a CSV or Excel sheet as a NaN. NaNs have some desirable properties: if we were to average the data column without replacing our NaNs, Python would know to skip over those cells.

```
rainfall_df['data'].mean()
0.37230130486357743
```

We can replace all NaN values with zeroes using the .fillna() method (after making a copy of the data so we don't lose our work):

```
df1['data'] = df1['data'].fillna(0)
df1.tail()
```

However NaN and 0 yield different analysis results. The mean value when NaN values are replaced with 0 is different from when NaN values are simply thrown out or ignored.

We can fill NaN values with any value that we chose. The code below fills all NaN values with a mean for all rainfall values.

```
 df1['data'] = df['data'].fillna(likes_df['data'].mean())
df1.tail()
```

We could also chose to create a subset of our data, only keeping rows that do not contain NaN values. .dropna() removes all rows with NaNs.

```
df_na = df.dropna()
```

B. Merge Data

Combining Datasets: Merge and Join

One essential feature offered by Pandas is its high-performance, in-memory join and merge

Relational Algebra

The behavior implemented in pd.merge() is a subset of what is known as *relational algebra*, which is a formal set of rules for manipulating relational data, and forms the conceptual foundation of operations available in most databases. Pandas implements several fundamental building-blocks in the pd.merge() function and the related join() method of Series and Dataframes.

Categories of Joins

The pd.merge() function implements a number of types of joins: the *one-to-one*, *many-to-one*, and *many-to-many* joins. All three types of joins are accessed via an identical call to the pd.merge() interface; the type of join performed depends on the form of the input data.

One-to-one joins

Perhaps the simplest type of merge expresion is the one-to-one join, which is in many ways very similar to the column-wise concatenation seen in Combining Datasets: Concat & Append. As a concrete example, consider the following two DataFrames which contain information on several employees in a company:

```
df1 = pd.DataFrame({'employee': ['Bob', 'Jake', 'Lisa', 'Sue'],
                    'group': ['Accounting', 'Engineering', 'Engineering', 'HR']})
df2 = pd.DataFrame({'employee': ['Lisa', 'Bob', 'Jake', 'Sue'],
                    'hire_date': [2004, 2008, 2012, 2014]})
```

To combine this information into a single DataFrame, we can use the pd.merge() function:

```
df3 = pd.merge(df1, df2)
df3
```

	employee	**group**	**hire_date**
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

The pd.merge() function recognizes that each DataFrame has an "employee" column, and automatically joins using this column as a key. The result of the merge is a new DataFrame that combines the information from the two inputs. Notice that the order of entries in each column is not necessarily maintained: in this case, the order of the "employee" column differs between df1 and df2, and the pd.merge() function correctly accounts for this. Additionally, keep in mind that the merge in general discards the index, except in the special case of merges by index

Many-to-one joins

Many-to-one joins are joins in which one of the two key columns contains duplicate entries. For the many-to-one case, the resulting DataFrame will preserve those duplicate entries as appropriate. Consider the following example of a many-to-one join:

```
df4 = pd.DataFrame({'group': ['Accounting', 'Engineering', 'HR'],
                'supervisor': ['Carly', 'Guido', 'Steve']})
Print('df3', 'df4', 'pd.merge(df3, df4)')
```

```
pd.merge(df3, df4)
```

	employee	group	hire_date	supervisor
0	Bob	Accounting	2008	Carly
1	Jake	Engineering	2012	Guido
2	Lisa	Engineering	2004	Guido
3	Sue	HR	2014	Steve

The resulting DataFrame has an aditional column with the "supervisor" information, where the information is repeated in one or more locations as required by the inputs.

Many-to-many joins

Many-to-many joins are a bit confusing conceptually, but are nevertheless well defined. If the key column in both the left and right array contains duplicates, then the result is a many-to-many merge. This will be perhaps most clear with a concrete example. Consider the following, where we have a DataFrame showing one or more skills associated with a particular group. By performing a many-to-many join, we can recover the skills associated with any individual person:

```
df5 = pd.DataFrame({'group': ['Accounting', 'Accounting',
                              'Engineering', 'Engineering', 'HR', 'HR'],
                'skills': ['math', 'spreadsheets', 'coding', 'linux',
                              'spreadsheets', 'organization']})
print('df1', 'df5', "pd.merge(df1, df5)")
```

```
pd.merge(df1, df5)
```

	employee	group	skills
0	Bob	Accounting	math
1	Bob	Accounting	spreadsheets
2	Jake	Engineering	coding
3	Jake	Engineering	linux
4	Lisa	Engineering	coding
5	Lisa	Engineering	linux
6	Sue	HR	spreadsheets
7	Sue	HR	organization

These three types of joins can be used with other Pandas tools to implement a wide array of functionality.

Specification of the Merge Key

We've already seen the default behavior of pd.merge(): it looks for one or more matching column names between the two inputs, and uses this as the key. However, often the column names will not match so nicely, and pd.merge() provides a variety of options for handling this.

The on keyword

Most simply, you can explicitly specify the name of the key column using the on keyword, which takes a column name or a list of column names:

```
print('df1', 'df2', "pd.merge(df1, df2, on='employee')")
pd.merge(df1, df2, on='employee')
```

	employee	group	hire_date
0	Bob	Accounting	2008
1	Jake	Engineering	2012
2	Lisa	Engineering	2004
3	Sue	HR	2014

This option works only if both the left and right DataFrames have the specified column name.

The left_on and right_on keywords

At times you may wish to merge two datasets with different column names; for example, we may have a dataset in which the employee name is labeled as "name" rather than "employee". In this case, we can use the left_on and right_on keywords to specify the two column names:

```
df3 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],
                    'salary': [70000, 80000, 120000, 90000]})
display('df1', 'df3', 'pd.merge(df1, df3, left_on="employee", right_on="name")')
pd.merge(df1, df3, left_on="employee", right_on="name")
```

	employee	group	name	salary
0	Bob	Accounting	Bob	70000
1	Jake	Engineering	Jake	80000

	employee	group	name	salary
2	Lisa	Engineering	Lisa	120000
3	Sue	HR	Sue	90000

The result has a redundant column that we can drop if desired–for example, by using the drop() method of DataFrames:

pd.merge(df1, df3, left_on="employee", right_on="name").drop('name', axis=1)

	employee	group	salary
0	Bob	Accounting	70000
1	Jake	Engineering	80000
2	Lisa	Engineering	120000
3	Sue	HR	90000

The left_index and right_index keywords

Sometimes, rather than merging on a column, you would instead like to merge on an index. For example, your data might look like this:

df1a = df1.set_index('employee')

df2a = df2.set_index('employee')

display('df1a', 'df2a')

df1a

	group
employee	
Bob	Accounting
Jake	Engineering
Lisa	Engineering
Sue	HR

df2a

	hire_date
employee	
Lisa	2004
Bob	2008
Jake	2012
Sue	2014

You can use the index as the key for merging by specifying the left_index and/or right_index flags in pd.merge():

```
display('df1a', 'df2a',
        "pd.merge(df1a, df2a, left_index=True, right_index=True)")
```

For convenience, DataFrames implement the join() method, which performs a merge that defaults to joining on indices:

display('df1a', 'df2a', 'df1a.join(df2a)')

df1adf1a.join(df2a)

If you'd like to mix indices and columns, you can combine left_index with right_on or left_on with right_index to get the desired behavior:

display('df1a', 'df3', "pd.merge(df1a, df3, left_index=True, right_on='name')")

df1a

df3

pd.merge(df1a, df3, left_index=True, right_on='name')

	group	name	salary
0	Accounting	Bob	70000
1	Engineering	Jake	80000
2	Engineering	Lisa	120000
3	HR	Sue	90000

Concatenating objects

The **concat()** function (in the main pandas namespace) does all of the heavy lifting of performing concatenation operations along an axis while performing optional set logic (union or intersection) of the indexes (if any) on the other axes. Note that I say "if any" because there is only a single possible axis of concatenation for Series.

```
In [1]: df1 = pd.DataFrame(
   ...:     {
   ...: "A": ["A0", "A1", "A2", "A3"],
   ...: "B": ["B0", "B1", "B2", "B3"],
   ...: "C": ["C0", "C1", "C2", "C3"],
   ...: "D": ["D0", "D1", "D2", "D3"],
   ...:     },
   ...:     index=[0, 1, 2, 3],
   ...: )
   ...:

In [2]: df2 = pd.DataFrame(
   ...:     {
   ...: "A": ["A4", "A5", "A6", "A7"],
   ...: "B": ["B4", "B5", "B6", "B7"],
   ...: "C": ["C4", "C5", "C6", "C7"],
   ...: "D": ["D4", "D5", "D6", "D7"],
   ...:     },
   ...:     index=[4, 5, 6, 7],
   ...: )
   ...:

In [3]: df3 = pd.DataFrame(
   ...:     {
   ...: "A": ["A8", "A9", "A10", "A11"],
   ...: "B": ["B8", "B9", "B10", "B11"],
   ...: "C": ["C8", "C9", "C10", "C11"],
   ...: "D": ["D8", "D9", "D10", "D11"],
   ...:     },
   ...:     index=[8, 9, 10, 11],
   ...: )
   ...:
```

```
In [4]: frames = [df1, df2, df3]

In [5]: result = pd.concat(frames)
```

C. Sort Data

Python Pandas - Sorting

There are two kinds of sorting available in Pandas. They are −

- By label
- By Actual Value

By Label

Using the **sort_index()** method, by passing the axis arguments and the order of sorting,

Order of Sorting

By passing the Boolean value to ascending parameter, the order of the sorting can be controlled.

Sort the Columns

By passing the axis argument with a value 0 or 1, the sorting can be done on the column labels. By default, axis=0, sort by row. Let us consider the following example to understand the same.

By Value

Like index sorting, **sort_values()** is the method for sorting by values. It accepts a 'by' argument which will use the column name of the DataFrame with which the values are to be sorted.

As a quick reminder, a **DataFrame** is a data structure with labeled axes for both rows and columns. You can sort a DataFrame by row or column value as well as by row or column index.Both rows and columns have **indices**, which are numerical representations of where the data is in your DataFrame. You can retrieve data from specific rows or columns using the DataFrame's index locations. By default, index numbers start from zero. You can also manually assign your own index.

Getting Familiar With .sort_values()

You use .sort_values() to sort values in a DataFrame along either axis (columns or rows). Typically, you want to sort the rows in a DataFrame by the values of one or more columns:

Getting Familiar With .sort_index()

You use .sort_index() to sort a DataFrame by its row index or column labels. The difference from using .sort_values() is that you're sorting the DataFrame based on its row index or column names, not by the values in these rows or columns:

Sorting Your DataFrame on a Single Column

To sort the DataFrame based on the values in a single column, you'll use .sort_values(). By default, this will return a new DataFrame sorted in ascending order. It does not modify the original DataFrame.

Sorting by a Column in Ascending Order

To use .sort_values(), you pass a single argument to the method containing the name of the column you want to sort by.

```
>>>df.sort_values(by = "like")
```

By default, .sort_values() sorts your data in **ascending order**.

Changing the Sort Order

Another parameter of .sort_values() is ascending. By default .sort_values() has ascending set to True. If you want the DataFrame sorted in **descending order**, then you can pass False to this parameter:

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

Choosing a Sorting Algorithm

It's good to note that pandas allows you to choose different **sorting algorithms** to use with both .sort_values() and .sort_index(). The available algorithms are quicksort, mergesort, and heapsort.

The algorithm used by default when sorting on a single column is quicksort. To change this to a stable sorting algorithm, use mergesort. You can do that with the kind parameter in .sort_values() or .sort_index(), like this:

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

Using kind, you set the sorting algorithm to mergesort..

Note: In pandas, kind is ignored when you sort on more than one column or label.

When you're sorting multiple records that have the same key, a **stable sorting algorithm(**mergesort) will maintain the original order of those records after sorting. For that reason, using a stable sorting algorithm is necessary if you plan to perform multiple sorts.

Sorting Your DataFrame on Multiple Columns

In data analysis, it's common to want to sort your data based on the values of multiple columns. Imagine you have a dataset with people's first and last names. It would make sense to sort by last name and then first name, so that people with the same last name are arranged alphabetically according to their first names.

```
>>>df.sort_values(by=["like", "share])
```

you sort the DataFrame on two columns using .sort_values

Changing the Column Sort Order

Since you're sorting using multiple columns, you can specify the order by which your columns get sorted. If you want to change the logical sort order from the previous example, then you can change the order of the column names in the list you pass to the by parameter:

```
>>>df.sort_values(by=["share","like"])
```

Your DataFrame is now sorted by the model column in ascending order, then sorted by make if there are two or more of the same model. You can see that changing the order of columns also changes the order in which the values get sorted.

Sorting by Multiple Columns in Descending Order

Up to this point, you've sorted only in ascending order on multiple columns. In the next example, you'll sort in descending order based on the make and model columns. To sort in descending order, set ascending to False:

```
>>>df.sort_values(by=["share","like"],ascending=False)
```

The values in the share column are in reverse alphabetical order, and the values in the like column are in descending order. With textual data, the sort is **case sensitive**, meaning capitalized text will appear first in ascending order and last in descending order.

Sorting by Multiple Columns With Different Sort Orders

You might be wondering if it's possible to sort using multiple columns and to have those columns use different ascending arguments. With pandas, you can do this with a single method call. If you want to sort some columns in ascending order and some columns in descending order, then you can pass a list of Booleans to ascending.

In this example, you sort your DataFrame by the make, model, and city08 columns, with the first two columns sorted in ascending order and city08 sorted in descending order. To do so, you pass a list of column names to by and a list of Booleans to ascending:

```
>>>df.sort_values(by=["share","like"],ascending=[True,False])
```

Now your DataFrame is sorted by share in ascending order, but with the like column in descending order.

Sorting Your DataFrame on Its Index

Before sorting on the index, it's a good idea to know what an index represents. A DataFrame has an **.index** property, which by default is a numerical representation of its rows' locations. You can think of the index as the row numbers. It helps in quick row lookup and identification.

Sorting by Index in Ascending Order

You can sort a DataFrame based on its row index with .sort_index(). Sorting by column values like you did in the previous examples reorders the rows in your DataFrame, so the index becomes disorganized. This can also happen when you filter a DataFrame or when you drop or add rows.

To illustrate the use of .sort_index(), start by creating a new sorted DataFrame using .sort_values():

```
>>> sorted_df = df.sort_values(by=["like", "share"])
>>>print(sorted_df)
```

You've created a DataFrame that's sorted using multiple values. Notice how the row index is in no particular order. To get your new DataFrame back to the original order, you can use .sort_index():

```
>>> sorted_df.sort_index()
```

Now the index is in ascending order. Just like .sort_values(), the default argument for ascending in .sort_index() is True, and you can change to descending order by passing False. Sorting on the index has no impact on the data itself as the values are unchanged.

This is particularly useful when you've assigned a custom index with **.set_index()**. If you want to set a custom index using the like and share columns, then you can pass a list to .set_index():

```
>>> assigned_index_df = df.set_index(["like", "share"])
>>> assigned_index_df
```

Using this method, you replace the default integer-based row index with two axis labels. This is considered a MultiIndex or a **hierarchical index**. Your DataFrame is now indexed by more than one key, which you can sort on with .sort_index():

```
>>> assigned_index_df.sort_index()
```

Sorting by Index in Descending Order

For the next example, you'll sort your DataFrame by its index in descending order. Remember from sorting your DataFrame with .sort_values() that you can reverse the sort order by setting ascending to False. This parameter also works with .sort_index(), so you can sort your DataFrame in reverse order like this:

```
>>> assigned_index_df.sort_index(ascending=False)
```

Sorting the Columns of Your DataFrame

You can also use the column labels of your DataFrame to sort row values. Using .sort_index() with the optional parameter axis set to 1 will sort the DataFrame by the column labels. The sorting algorithm is applied to the **axis labels** instead of to the actual data. This can be helpful for visual inspection of the DataFrame.

Working With the DataFrame axis

When you use .sort_index() without passing any explicit arguments, it uses axis=0 as a default argument. The **axis** of a DataFrame refers to either the index (axis=0) or the columns (axis=1). You can use both axes for indexing and selecting data in a DataFrame as well as for sorting the data.

Using Column Labels to Sort

You can also use the column labels of a DataFrame as the sorting key for .sort_index(). Setting axis to 1 sorts the columns of your DataFrame based on the column labels:

```
>>> df.sort_index(axis=1)
```

The columns of your DataFrame are sorted from left to right in ascending alphabetical order. If you want to sort the columns in descending order, then you can use ascending=False:

```
>>> df.sort_index(axis=1, ascending=False)
```

D. Transposing Data

Transpose DataFrame (swap rows and columns)

Use the T attribute or the transpose() method to swap (= transpose) the rows and columns of pandas.DataFrame.

Neither method changes the original object but returns a new object with the rows and columns swapped (= transposed object).

Note that depending on the data type dtype of each column, a view is created instead of a copy, and changing the value of one of the original and transposed objects will change the other.

Syntax

dataframe.transpose(args, copy)

Parameters

The parameters are keyword arguments.

Parameter	Value	Description
args	Tuple	Optional. arguments that can be used in NumPy functions
copy	True False	Optional, default False. Specifies whether to copy the data or not

Return Value

A DataFrame where the columns have been rows and vice versa.

This method does not change the original DataFrame.

E. Shape and reshape Data

The shape property returns a tuple containing the shape of the DataFrame.

The shape is the number of rows and columns of the DataFrame

dataframe.shape

Return Value

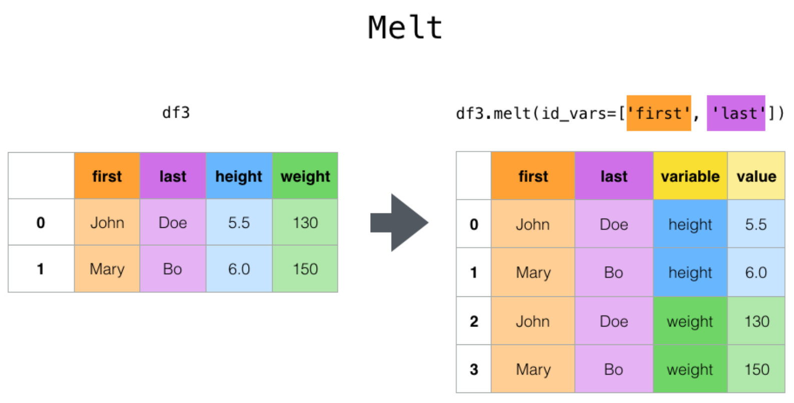
a Python Tuple showing the number of rows and columns.

reshaping means the transformation of the structure of a table or vector (i.e. DataFrame or Series) to make it suitable for further analysis..

Reshaping by melt

To make analysis of data in table easier, we can reshape the data into a more computer-friendly form using Pandas in Python. Pandas.melt() is one of the function to do so.. Pandas.melt() unpivots a DataFrame from wide format to long format. **melt()** function is useful to message a DataFrame into a format where one or more columns are identifier variables, while all other columns, considered measured variables, are unpivoted to the row axis, leaving just two non-identifier columns, variable and value. Syntax :

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



	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150

```
df3.melt(id_vars=['first', 'last'])
```

	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150

The top-level **melt()** function and the corresponding **DataFrame.melt()** are useful to massage a **DataFrame** into a format where one or more columns are *identifier variables*, while all other columns, considered *measured variables*, are "unpivoted" to the row axis, leaving just two non-identifier columns, "variable" and "value". The names of those columns can be customized by supplying the var_name and value_name parameters.

For instance,

When transforming a DataFrame using **melt()**, the index will be ignored. The original index values can be kept around by setting the ignore_index parameter to False (default is True). This will however duplicate them.

Pivot

The pivot function is used to create a new derived table out of a given one. Pivot takes 3 arguements with the following names: index, columns, and values. As a value for each of these parameters you need to specify a column name in the original table. Then the pivot function will create a new table, whose row and column indices are the unique values of the respective parameters. The cell values of the new table are taken from column given as the values parameter. Assume that we are given the following small table:

In [1]:

```
fromcollectionsimport OrderedDict
frompandasimport DataFrame
importpandasaspd
importnumpyasnp

table = OrderedDict((
    ("Item", ['Item0', 'Item0', 'Item1', 'Item1']),
    ('CType',['Gold', 'Bronze', 'Gold', 'Silver']),
    ('USD',  ['1$', '2$', '3$', '4$']),
    ('EU',   ['1€', '2€', '3€', '4€'])
))
d = DataFrame(table)
d
```

In [2]:

```
p = d.pivot(index='Item', columns='CType', values='USD')
p
```

Out[2]:

CType	Bronze	Gold	Silver
Item			
Item0	2$	1$	None
Item1	None	3$	4$

Pivoting By Multiple Columns

Now what if we want to extend the previous example to have the EU cost for each item on its row as well? This is actually easy - we just have to omit the values parameter as follows:

In [4]:

```
p = d.pivot(index='Item', columns='CType')
p
```

Out[4]:

	USD			**EU**		
CType	**Bronze**	**Gold**	**Silver**	**Bronze**	**Gold**	**Silver**

Item						
Item0	2$	1$	None	2€	1€	None
Item1	None	3$	4$	None	3€	4€

We can use this hierarchical column index to filter the values of a single column from the original table. For example p.USD returns a pivoted DataFrame with the USD values only and it is equivalent to the pivoted DataFrame from the previous section.

In [5]:

```
p.USD
```

Out[5]:

CType	Bronze	Gold	Silver
Item			
Item0	2$	1$	None
Item1	None	3$	4$

In [6]:

```
p.USD.Bronze
```

Out[6]:

```
Item
Item0     2$
Item1    None
Name: Bronze, dtype: object
```

In [7]:

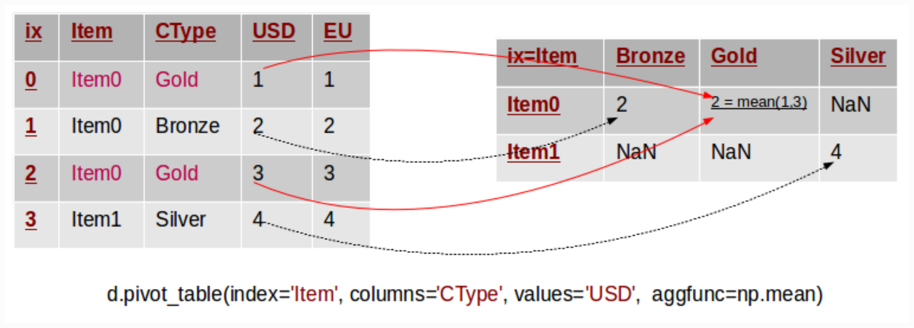
```
# Original DataFrame: Access the USD cost of Item0 for Gold customers
print(d[(d.Item=='Item0') & (d.CType=='Gold')].USD.values)

# Pivoted DataFrame: p.USD gives a "sub-DataFrame" with the USD values only
print(p.USD[p.USD.index=='Item0'].Gold.values)
```

```
['1$']
['1$']
```

Pivot Table

The pivot_table method comes to solve this problem. It works like pivot, but it aggregates the values from rows with duplicate entries for the specified columns. In other words, in the previous example we could have used the mean, the median or another aggregation function to compute a single value from the conflicting entries. This is depicted in the example below.



ix	Item	CType	USD	EU
0	Item0	Gold	1	1
1	Item0	Bronze	2	2
2	Item0	Gold	3	3
3	Item1	Silver	4	4

ix=Item	Bronze	Gold	Silver
Item0	2	2 = mean(1,3)	NaN
Item1	NaN	NaN	4

```
d.pivot_table(index='Item', columns='CType', values='USD',  aggfunc=np.mean)
```

In [8]:

```
table = OrderedDict((
    ("Item", ['Item0', 'Item0', 'Item0', 'Item1']),
    ('CType',['Gold', 'Bronze', 'Gold', 'Silver']),
    ('USD',  [1, 2, 3, 4]),
    ('EU',   [1.1, 2.2, 3.3, 4.4])
))
d = DataFrame(table)
p = d.pivot_table(index='Item', columns='CType', values='USD', aggfunc=np.sum)
p.fillna(value='--',inplace=True)
p
```

Out[8]:

CType	Bronze	Gold	Silver
Item			
Item0	2	4	--
Item1	--	--	4

In essence pivot_table is a generalisation of pivot, which allows you to aggregate multiple values with the same destination in the pivoted table.

While **pivot()** provides general purpose pivoting with various data types (strings, numerics, etc.), pandas also provides **pivot_table()** for pivoting with aggregation of numeric data.

The function **pivot_table()** can be used to create spreadsheet-style pivot tables. It takes a number of arguments:

- data: a DataFrame object.
- values: a column or a list of columns to aggregate.
- index: a column, Grouper, array which has the same length as data, or list of them. Keys to group by on the pivot table index. If an array is passed, it is being used as the same manner as column values.
- columns: a column, Grouper, array which has the same length as data, or list of them. Keys to group by on the pivot table column. If an array is passed, it is being used as the same manner as column values.
- aggfunc: function to use for aggregation, defaulting to numpy.mean.

We can produce pivot tables from this data very easily:

```
In [67]: pd.pivot_table(df, values="D", index=["A", "B"], columns=["C"])
Out[67]:
C             bar       foo
A     B
one   A  1.120915 -0.514058
      B -0.338421  0.002759
      C -0.538846  0.699535
three A -1.181568       NaN
      B       NaN  0.433512
      C  0.588783       NaN
two   A       NaN  1.000985
      B  0.158248       NaN
      C       NaN  0.176180
```

Note that pivot_table() is also available as an instance method on DataFrame, i.e. **DataFrame.pivot_table()**.

Conclusion :

We have learned to :

a. Create data subsets
b. Merge Data
c. Sort Data
d. Transposing Data
e. Shape and reshape Data

Assignment – 5 (Part B)

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

a. Data cleaning

b. Data integration

c. Data transformation

d. Error correcting

e. Data model building

Objective : students are able to perform Data cleaning , Data integration , Data transformation, Error correcting , Data model building

Theory:

Data cleaning Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

1. **Data cleansing** may be performed interactively with data wrangling tools, or as batch processing through scripting.
2. **Data integration** : Data integration involves combining data residing in different sources and providing users with a unified view of them. This process becomes significant in a variety of situations, which include both commercial (such as when two similar companies need to merge their databases) and scientific (combining research results from different bioinformatics repositories, for example) domains.

Data integration appears with increasing frequency as the volume and the need to share existing data explodes.[3] It has become the focus of extensive theoretical work, and numerous open problems remain unsolved.

3. **Data transformation** : In computing, data transformation is the process of converting data from one format or structure into another format or structure. It is a fundamental aspect of most data integration and data management tasks such as data wrangling, data warehousing, data integration and application integration.
4. **Error correcting** : Error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data in many cases.
5. **Data Modeling :** python has been the language of choice for predictive analysis due to its innumerable packages and strong developer community. Stages of Predictive Modeling Predictive Modeling is the process of building a model to predict future outcomes using statistics techniques. In order to generate the model, historical data of prior occurrences needs to be analyzed, classified and validated.

a. Data cleaning

Data cleaning or cleansing is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

how to find and clean:

Missing Data –

- There are 4 ways to find the null values if present in the dataset.

Using isnull() function (data.isnull()**) :** This function provides the boolean value for the complete dataset to know if any null value is present or not.

Using isna() function:data.isna() : This is the same as the isnull() function. Ans provides the same output.

Using isna().any()

This function also gives a boolean value if any null value is present or not, but it gives results column-wise, not in tabular format.

Using isna(). sum()

This function gives the sum of the null values preset in the dataset column-wise.

Using isna().any().sum()

This function gives output in a single value if any null is present or not.

if there are any null values preset we can fill those places with any other value using the fillna() function of DataFrame.

Following is the syntax of fillna() function:

DataFrame_name.fillna(*value=None*, *method=None*, *axis=None*, *inplace=False*, *limit=None*, *downcast=None*)

Cleaning / Filling Missing Data

Pandas provides various methods for cleaning the missing values. The fillna function can "fill in" NA values with non-null data in a couple of ways.

Replace NaN with a Scalar Value

The following program shows how you can replace "NaN" with "0".

```
df.fillna(0)
```

Drop Missing Values

If you want to simply exclude the missing values, then use the **dropna** function along with the **axis** argument. By default, axis=0, i.e., along row, which means that if any value within a row is NA then the whole row is excluded.

```
df.dropna()
```

In statistics, this method is called the listwise deletion technique. In this solution, we drop the entire observation as long as it contains a missing value.

Only if we are sure that the missing data is not informative, we perform this. Otherwise, we should consider other solutions.

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

Drop the Observation

Here, we consider that only a minimal amount of observations have over 5 features missing altogether. We may create a new dataset *df_less_missing_rows* deleting observations with over 5 missing features.

```
# drop rows with a lot of missing values
ind_missing=df[df['num_missing'] >5].index
df_less_missing_rows=df.drop(ind_missing, axis=0)
```

Drop the Feature

we *only* do this when we are confident that this feature doesn't provide useful information.

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.

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

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.

Notice that the 'type' column has values such as 'Industrial Area' and 'Industrial Areas'—both actually mean the same, so let's remove them and make it uniform

```
df["type"].unique()
types = {
   "Residential": "R",
   "Residential and others": "RO",
   "Residential, Rural and other Areas": "RRO",
   "Industrial Area": "I",
   "Industrial Areas": "I",
   "Industrial": "I",
   "Sensitive Area": "S",
   "Sensitive Areas": "S",
   "Sensitive": "S",
   "NaN": "RRO"
}
df.type = df.type.replace(types)
```

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.

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.

```
df['date'] = pd.to_datetime(df['date'], errors='coerce')
df.head(5)
df['year'] = df.date.dt.year
df.head(5)
```

Handling Missing Values

The column such as SO2, NO2, rspm, spm, pm2_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']
import numpy as np
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])
```

Data Integration

```
>sport=c("hockey","baseball","football")
 > league=c("L1","L2","L3")
 > trophy=c("SACH","SAU","YUV")
>trophies1=cbind(sport,league,trophy)
```

 > trophies1 sport league trophy [1,] "hockey""L1""SACH" [2,] "baseball""L2""SAU" [3,] "football""L3""YUV" > trophies2=data.frame(sport=("Swiming"),league=("Lee"),trophy=("GAV"),stringsAsFactors = FALSE) > trophies=rbind(trophies1,trophies2) > trophies sport league trophy 1 hockey L1 SACH 2 baseball L2 SAU 3 football L3 YUV 4 Swiming Lee GAV

Data Transformation

All machine learning algorithms are based on mathematics. So, we need to convert all the columns into numerical format.

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

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

Discovering Duplicates

Duplicate rows are rows that have been registered more than one time.

By taking a look at our test data set, we can assume that row 11 and 12 are duplicates.

To discover duplicates, we can use the duplicated() method.

The duplicated() method returns a Boolean values for each row:

Return True for every row that is a duplicate, otherwise False.

df.duplicated()

df.duplicated().sum()

Removing Duplicates

To remove duplicates, use the drop_duplicates() method.

df.drop_duplicates(inplace = True)

Simple Replacement of Categorical Data with a Number

When we look at the categorical data, the first question that arises to anyone is how to handle those data, because machine learning is always good at dealing with numeric values. We could make machine learning models by using text data. So, to make predictive models we have to convert categorical data into numeric form.

1. 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.

2. Replacement: This is the technique in which we replace the categorical data with a number. This is a simple replacement and does not involve much logical processing.

Using replace() method

Replacing is one of the methods to convert categorical terms into numeric.

df.head()

df['type'].value_counts()

df['type'].replace({ 'MO':1, 'I':2, 's':3 , 'RO':4, 'K':5, 'RIRUO':6 }, inplace=True)

df.info()

df['type']

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Example :

Suppose we have a column *Height* in some dataset.

Height
Tall
Medium
Short

After applying label encoding, the Height column is converted into:

Height
0
1
2

where 0 is the label for tall, 1 is the label for medium, and 2 is a label for short height.

Exam. from sklearn.preprocessing import LabelEncoder

```
# label_encoder object knows how to understand word labels.
labelencoder = LabelEncoder()
# Encode labels in column 'state'.
df['state'] =labelencoder.fit_transform(df['state'])
```

Limitation of label Encoding

Label encoding converts the data in machine-readable form, but it assigns a unique number(starting from 0) to each class of data. This may lead to the generation of priority issues in the training of data sets. A label with a high value may be considered to have high priority than a label having a lower value.

One-Hot Encoder

# One-hot encoding is used to convert categorical variables into a format that can be readily used by machine learning algorithms.

Though label encoding is straight but it has the disadvantage that the numeric values can be misinterpreted by algorithms as having some sort of hierarchy/order in them. This ordering issue is addressed in another common alternative approach called 'One-Hot Encoding'. In this strategy, each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column.

Using sci-kit learn library approach:

OneHotEncoder from SciKit library only takes numerical categorical values, hence any value of string type should be label encoded before one hot encoded.

```
dfAndhra = df[df['state']==0]
dfAndhra
dfAndhra['location'].value_counts()
from sklearn.preprocessing import OneHotEncoder
# creating instance of one-hot-encoder
onehotencoder = OneHotEncoder(sparse=False, handle_unknown='error', drop='first')
#passing dfAndhra column (label encoded values of dfAndhra)
pd.DataFrame(onehotencoder.fit_transform(dfAndhra[['location']]))
dfAndhra['location'].value_counts()
```

Error correction

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

# Detecting and Filtering Outliers

```
df.describe()
df[df['so2']>100]=0
```

Detect Outliers

Outliers are numerical values that lie significantly outside of the statistical norm. Cutting that down from unnecessary science garble – they are data points that are so out of range they are likely misreads. They, like duplicates, need to be removed.

How to find out?

Depending on whether the feature is numeric or categorical, we can use different techniques to study its distribution to detect outliers.

- **Technique #1: Histogram/Box Plot**

When the feature is numeric, we can use a histogram and box plot to detect outliers.

- **Technique #2: Descriptive Statistics**

Also, for numeric features, the outliers could be too distinct that the box plot can't visualize them. Instead, we can look at their descriptive statistics.

For example, for the feature *life_sq* again, we can see that the maximum value is 7478, while the 75% quartile is only 43. The 7478 value is an outlier.

Data Model building : (write answer for following in journal)

Explain SVM

Explain performance measure like precision, recall , Accuracy

Explain Data model building

Conclusion :

FAQ :

- How data analytics performed using python?
- What is the need of data analytics?
- Explain different types of data analytics.
- Why data transformation is important in Big Data Analytics?
- Explain data cleaning process.
- Why data is dirty?
- Explain data analytics life cycle.
- What is data analytics?

Assignment 7 (Part B)

Aim:

Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 2 and 3

Objective :

Students are able to draw different graphs.

Prerequisites:

Ensure that python programming is installed, configured and is running.

Theory

What is Data Visualization in Python?

Data visualization the visual representation of data in the form of graphs and plots and is particularly useful as non technical people often understand data and analysis presented in a visual form much better than with complicated numbers and tables.

Data visualization enables us to identify patterns or trends easily, as well as help to visualize data distribution, correlation and causality.

1. Pie Charts

A pie-chart is a representation of values as slices of a circle with different colors. The slices are labeled and the numbers corresponding to each slice is also represented in the chart.

The pie chart is created using the **pie()** function which takes positive numbers as a vector input. The additional parameters are used to control labels, color, title etc.

Syntax

The basic syntax for creating a pie-chart using the R is −

pie(x, labels, radius, main, col, clockwise)

Following is the description of the parameters used −

- **x**is a vector containing the numeric values used in the pie chart.
- **Labels** is used to give description to the slices.
- **Radius** indicates the radius of the circle of the pie chart.(value between −1 and +1).
- **main**indicates the title of the chart.
- **Col** indicates the color palette.
- Clockwise is a logical value indicating if the slices are drawn clockwise or anti clockwise.

2. Bar Charts

A bar graph is a graphical representation of data in which we can highlight the category with particular shapes like a rectangle. The length and heights of the bar chart represent the data distributed in the dataset. In a bar chart, we have one axis representing a particular category of a column in the dataset and another axis representing the values or counts associated with it. Bar charts can be plotted vertically or horizontally. A vertical bar chart is often called a column chart. When we arrange bar charts in a high to low-value counts manner, we called them Pareto charts.The function **barplot()** to create bar charts. We can draw both vertical and Horizontal bars in the bar chart. In bar chart each of the bars can be given different colors.

Matplotlib is a maths library widely used for data exploration and visualization. It is simple and provides us with the API to access functions like the ones used in MATLAB. We import the library as plt and use:

plt.bar(x, height, width, bottom, align)

Seaborn:

Seaborn is also a visualization library based on matplotlib and is widely used for presenting data. We can import the library as sns and use the following syntax:

seaborn.barplot(x=' ', y=' ',data=df)

Unstacked bar plots:

Unstacked bar plots are used when we cant to compare a particular category over time with different samples. It can be used to deduct some facts from the pattern we observe through the comparison. Like in the figure below, we can see the players' ratings over the years in FIFA. We can see that Django and Gafur have been increasing ratings over years. This shows us their progression and so a club can now decide if they want to sign Django or Gafur.

Stacked bar plots:

As the name suggests, stacked bar plots have each plot stacked one over them. As we saw earlier that we had used an unstacked bar chart for the comparison of each group, we can use a stacked plot for the comparison of each individual. In pandas, this is easy to implement using the stacked keyword.

Seaborn.countplot()

seaborn.countplot() method is used to Show the counts of observations in each categorical bin using bars.

Syntax : seaborn.countplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, dodge=True, ax=None, **kwargs)

Parameters : This method is accepting the following parameters that are described below:

- **x, y:** This parameter take names of variables in data or vector data, optional, Inputs for plotting long-form data.

- **hue :** (optional) This parameter take column name for colour encoding.
- **data :** (optional) This parameter take DataFrame, array, or list of arrays, Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.
- **order, hue_order :** (optional) This parameter take lists of strings. Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.
- **orient :** (optional)This parameter take "v" | "h", Orientation of the plot (vertical or horizontal). This is usually inferred from the dtype of the input variables but can be used to specify when the "categorical" variable is a numeric or when plotting wide-form data.
- **color :** (optional) This parameter take matplotlib color, Color for all of the elements, or seed for a gradient palette.
- **palette :** (optional) This parameter take palette name, list, or dict, Colors to use for the different levels of the hue variable. Should be something that can be interpreted by color_palette(), or a dictionary mapping hue levels to matplotlib colors.
- **saturation :** (optional) This parameter take float value, Proportion of the original saturation to draw colors at. Large patches often look better with slightly desaturated colors, but set this to 1 if you want the plot colors to perfectly match the input color spec.
- **dodge :** (optional) This parameter take bool value, When hue nesting is used, whether elements should be shifted along the categorical axis.
- **ax :** (optional) This parameter take matplotlib Axes, Axes object to draw the plot onto, otherwise uses the current Axes.
- **kwargs :** This parameter take key, value mappings, Other keyword arguments are passed through to matplotlib.axes.Axes.bar().

Returns: Returns the Axes object with the plot drawn onto it.

What is a Seaborn Distplot?

A Distplot or distribution plot, depicts the variation in the data distribution. Seaborn Distplot represents the overall distribution of continuous data variables.

The Seaborn module along with the Matplotlib module is used to depict the distplot with different variations in it. The Distplot depicts the data by a histogram and a line in combination to it.

Creating a Seaborn Distplot

Python Seaborn module contains various functions to plot the data and depict the data variations. The seaborn.distplot() is used to plot the distplot. The distplot represents the univariate distribution of data i.e. data distribution of a variable against the density distribution.

Syntax:

seaborn.distplot()

The seaborn.distplot() function accepts the data variable as an argument and returns the plot with the density distribution.

seaborn.Implot() method

seaborn.lmplot() method is used to draw a scatter plot onto a FacetGrid.

Syntax : seaborn.lmplot(x, y, data, hue=None, col=None, row=None, palette=None, col_wrap=None, height=5, aspect=1, markers='o', sharex=True, sharey=True, hue_order=None, col_order=None, row_order=None, legend=True, legend_out=True, x_estimator=None, x_bins=None, x_ci='ci', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, seed=None, order=1, logistic=False, lowest=False, robust=False, logx=False, x_partial=None, y_partial=None, truncate=True, x_jitter=None, y_jitter=None, scatter_kws=None, line_kws=None, size=None)

Parameters : This method is accepting the following parameters that are described below:

- **x, y**: (optional) This parameters are column names in data.
- **data :** This parameter is DataFrame .
- **hue, col, row :** This parameters are define subsets of the data, which will be drawn on separate facets in the grid. See the *_order parameters to control the order of levels of this variable.
- **palette**: (optional) This parameter is palette name, list, or dict, Colors to use for the different levels of the hue variable. Should be something that can be interpreted by color_palette(), or a dictionary mapping hue levels to matplotlib colors.
- **col_wrap :** (optional) This parameter is of int type, "Wrap" the column variable at this width, so that the column facets span multiple rows. Incompatible with a row facet.
- **height :** (optional) This parameter is Height (in inches) of each facet.
- **aspect :** (optional) This parameter is Aspect ratio of each facet, so that aspect * height gives the width of each facet in inches.
- **markers :** (optional) This parameter is matplotlib marker code or list of marker codes, Markers for the scatterplot. If a list, each marker in the list will be used for each level of the hue variable.
- **share{x, y} :** (optional) This parameter is of bool type, 'col', or 'row', If true, the facets will share y axes across columns and/or x axes across rows.
- **{hue, col, row}_order :** (optional) This parameter is lists, Order for the levels of the faceting variables. By default, this will be the order that the levels appear in data or, if the variables are pandas categoricals, the category order.
- **legend :** (optional) This parameter accepting bool value, If True and there is a hue variable, add a legend.
- **legend_out :** (optional) This parameter accepting bool value, If True, the figure size will be extended, and the legend will be drawn outside the plot on the center right.

- **x_estimator :** (optional)This parameter is callable that maps vector -> scalar, Apply this function to each unique value of x and plot the resulting estimate. This is useful when x is a discrete variable. If x_ci is given, this estimate will be bootstrapped and a confidence interval will be drawn.
- **x_bins :** (optional) This parameter is int or vector, Bin the x variable into discrete bins and then estimate the central tendency and a confidence interval. This binning only influences how the scatter plot is drawn; the regression is still fit to the original data. This parameter is interpreted either as the number of evenly-sized (not necessary spaced) bins or the positions of the bin centers. When this parameter is used, it implies that the default of x_estimator is numpy.mean.
- **x_ci :** (optional) This parameter is "ci", "sd", int in [0, 100] or None, Size of the confidence interval used when plotting a central tendency for discrete values of x. If "ci", defer to the value of the ci parameter. If "sd", skip bootstrapping and show the standard deviation of the observations in each bin.
- **scatter :** (optional) This parameter accepting bool value . If True, draw a scatterplot with the underlying observations (or the x_estimator values).
- **fit_reg :** (optional) This parameter accepting bool value . If True, estimate and plot a regression model relating the x and y variables.
- **ci :** (optional) This parameter is int in [0, 100] or None, Size of the confidence interval for the regression estimate. This will be drawn using translucent bands around the regression line. The confidence interval is estimated using a bootstrap; for large datasets, it may be advisable to avoid that computation by setting this parameter to None.
- **n_boot :** (optional) This parameter is Number of bootstrap resamples used to estimate the ci. The default value attempts to balance time and stability; you may want to increase this value for "final" versions of plots.
- **units :** (optional) This parameter is variable name in data, If the x and y observations are nested within sampling units, those can be specified here. This will be taken into account when computing the confidence intervals by performing a multilevel bootstrap that resamples both units and observations (within unit). This does not otherwise influence how the regression is estimated or drawn.
- **seed :** (optional) This parameter is int, numpy.random.Generator, or numpy.random.RandomState, Seed or random number generator for reproducible bootstrapping.
- **order :** (optional) This parameter, order is greater than 1, use numpy.polyfit to estimate a polynomial regression.
- **logistic :** (optional) This parameter accepting bool value, If True, assume that y is a binary variable and use statsmodels to estimate a logistic regression model. Note that this is substantially more computationally intensive than linear regression, so you may wish to decrease the number of bootstrap resamples (n_boot) or set ci to None.

- **lowest :** (optional) This parameter accepting bool value, If True, use statsmodels to estimate a non-parametric lowest model (locally weighted linear regression). Note that confidence intervals cannot currently be drawn for this kind of model.
- **robust :** (optional) This parameter accepting bool value, If True, use statsmodels to estimate a robust regression. This will de-weight outliers. Note that this is substantially more computationally intensive than standard linear regression, so you may wish to decrease the number of bootstrap resamples (n_boot) or set ci to None.
- **logx :** (optional) This parameter accepting bool value. If True, estimate a linear regression of the form y ~ log(x), but plot the scatterplot and regression model in the input space. Note that x must be positive for this to work.
- **{x, y}_partial :** (optional) This parameter is strings in data or matrices, Confounding variables to regress out of the x or y variables before plotting.
- **truncate :** (optional) This parameter accepting bool value.If True, the regression line is bounded by the data limits. If False, it extends to the x axis limits.
- **{x, y}_jitter :** (optional) This parameter is Add uniform random noise of this size to either the x or y variables. The noise is added to a copy of the data after fitting the regression, and only influences the look of the scatterplot. This can be helpful when plotting variables that take discrete values.
- **{scatter, line}_kws :** (optional)dictionaries

Returns : This method returns the FacetGrid object with the plot on it for further tweaking.

3. Histogram

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

A histogram is used to summarize continuous or discrete data. For example, you have the notes of the introductory computer science course of the students of X universities. You want to see which grade has been taken the most, with less than average grade fields or more grade fields. So you want to briefly see the distribution of the grades. Histograms are used to view these summary statistics.

Often, histograms are confused with bar charts. Histograms are similar to bar charts, but a histogram groups numbers into intervals. Histograms are used to show the distribution of the variable, while bar charts are used to compare variables. For example, we can use bar graphs to compare the average of students in a class over 5 lessons. However, histograms are used to see summary information of students' performance in all lessons in a class.

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. It is a kind of bar graph.

To construct a histogram, follow these steps −

- **Bin** the range of values.
- Divide the entire range of values into a series of intervals.
- Count how many values fall into each interval.

The bins are usually specified as consecutive, non-overlapping intervals of a variable.

The **matplotlib.pyplot.hist()** function plots a histogram. It computes and draws the histogram of x.

Parameters

The following table lists down the parameters for a histogram −

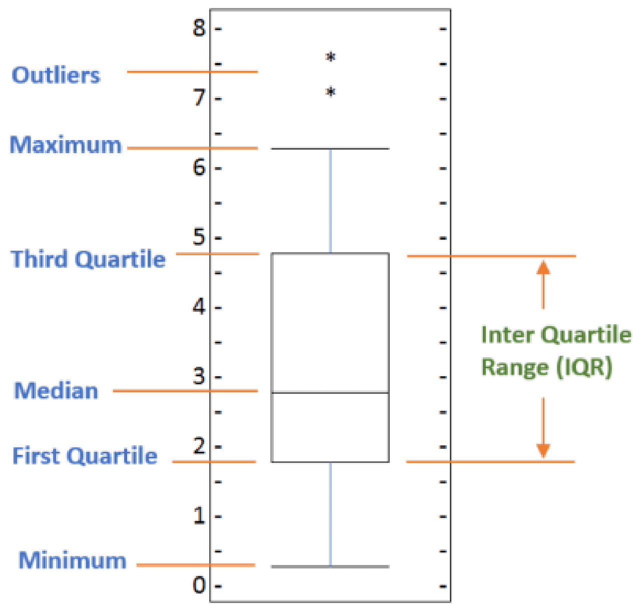
x	array or sequence of arrays
bins	integer or sequence or 'auto', optional
optional parameters	
range	The lower and upper range of the bins.
density	If True, the first element of the return tuple will be the counts normalized to form a probability density
cumulative	If True, then a histogram is computed where each bin gives the counts in that bin plus all bins for smaller values.
histtype	The type of histogram to draw. Default is 'bar' • 'bar' is a traditional bar-type histogram. If multiple data are given the bars are arranged side by side. • 'barstacked' is a bar-type histogram where multiple data are stacked on top of each other. • 'step' generates a lineplot that is by default unfilled. • 'stepfilled' generates a lineplot that is by default filled.

4. Box Plots

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

A box plot is a way of statistically representing the distribution of the data through five main dimensions:

- **Minimun:** Smallest number in the dataset.
- **First quartile:** Middle number between the minimum and the median.
- **Second quartile (Median):** Middle number of the (sorted) dataset.
- **Third quartile:** Middle number between median and maximum.
- **Maximum:** Highest number in the dataset.



A box plot helps to maintain the distribution of quantitative data in such a way that it facilitates the comparisons between variables or across levels of a categorical variable. The main body of the box plot showing the quartiles and the median's confidence intervals if enabled. The medians have horizontal lines at the median of each box and while whiskers have the vertical lines extending to the most extreme, non-outlier data points and caps are the horizontal lines at the ends of the whiskers.

Syntax: seaborn.boxplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, ax=None, **kwargs)

Parameters:

x, y, hue: Inputs for plotting long-form data.

data: Dataset for plotting. If x and y are absent, this is interpreted as wide-form.

color: Color for all of the elements.

Returns: It returns the Axes object with the plot drawn onto it.

5. Scatterplots

Scatterplots show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.One very commonly used tool in exploratory analysis of multivariate data is the scatterplot.
Scatter plots are used to observe relationship between variables and uses dots to represent the relationship between them. The **scatter()** method in the matplotlib library is used to draw a scatter plot. Scatter plots are widely used to represent relation among variables and how change in one affects the other.

Syntax

The syntax for scatter() method is given below:

matplotlib.pyplot.scatter(x_axis_data, y_axis_data, s=None, c=None, marker=None, cmap=None, vmin=None, vmax=None, alpha=None, linewidths=None, edgecolors=None)

The scatter() method takes in the following parameters:

- **x_axis_data-** An array containing x-axis data
- **y_axis_data-** An array containing y-axis data
- **s-** marker size (can be scalar or array of size equal to size of x or y)
- **c-** color of sequence of colors for markers
- marker- marker style
- **cmap-** cmap name
- **linewidths-** width of marker border
- **edgecolor-** marker border color
- **alpha-** blending value, between 0 (transparent) and 1 (opaque)

Except x_axis_data and y_axis_data all other parameters are optional and their default value is None. Below are the scatter plot examples with various parameters.

Code:

Conclusion :

Assignment 8 (Part B)

Aim:

Perform the following data visualization operations using Tableau on Adult and Iris datasets.

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

Objective :

Students are able to draw different graphs.

Prerequisites:

Ensure that Tablue is installed, configured and is running.

Theory:

Install Tableau Desktop.

Link to download tableau :

https://www.tableau.com

Adult dataset and IRIS Dataset

Download it from UCI repository or kaggle.com

https://archive.ics.uci.edu/ml/datasets/adult

https://www.kaggle.com/wenruliu/adult-income-dataset

https://archive.ics.uci.edu/ml/datasets/iris

https://www.kaggle.com/arshid/iris-flower-dataset

The Adult dataset is from the Census Bureau and the task is to predict whether a given adult makes more than $50,000 a year based attributes such as education, hours of work per week, etc.

The dataset provides 14 input variables that are a mixture of categorical, ordinal, and numerical data types. The complete list of variables is as follows:

Age.

Workclass.

Final Weight.

Education.

Education Number of Years.

Marital-status.

Occupation.

Relationship.

Race.

Sex.

Capital-gain.

Capital-loss.

Hours-per-week.

Native-country

There are a total of 48,842 rows of data

Iris Dataset

Anderson collected the data to quantify the morphologic variation of Iris flowers of three related species.

The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica, and Iris versicolor).

Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

This dataset became a typical test case for many statistical classification techniques in machine learning such as support vector machines

The dataset contains a set of 150 records under 5 attributes - Petal Length, Petal Width, Sepal Length, Sepal width and Class(Species).

This dataset is free and is publicly available at the UCI Machine Learning Repository

What is TableAU?

ableau is a powerful and fastest growing data visualization tool used in the Business Intelligence Industry. It helps in simplifying raw data into the very easily understandable format.

Data analysis is very fast with Tableau and the visualizations created are in the form of dashboards and worksheets. The data that is created using Tableau can be understood by professional at any level in an organization. It even allows a non-technical user to create a customized dashboard.

The best feature Tableau are: 1)Data Blending 2)Real time analysis 3)Collaboration of data

Tableau Product Suite

The Tableau Product Suite consists of:

Tableau Desktop

Tableau Public

Tableau Online

Tableau Server

Tableau Reader

For clear understanding, data analytics in tableau can be classified into two section:

- Developer Tools: The Tableau tools that are used for development such as the creation of dashboards, charts, report generation, visualization fall into this category. The Tableau products, under this category, are the Tableau Desktop and the Tableau Public.
- Sharing Tools: As the name suggests, the purpose of the tool is sharing the visualizations, reports, dashboards that were created using the developer tools. Products that fall into this category are Tableau Online, Server, and Reader.

Open Tableau Desktop and begin

The first thing you see after you open Tableau Desktop is the start page. Here, you select the connector (how you will connect to your data) that you want to use.

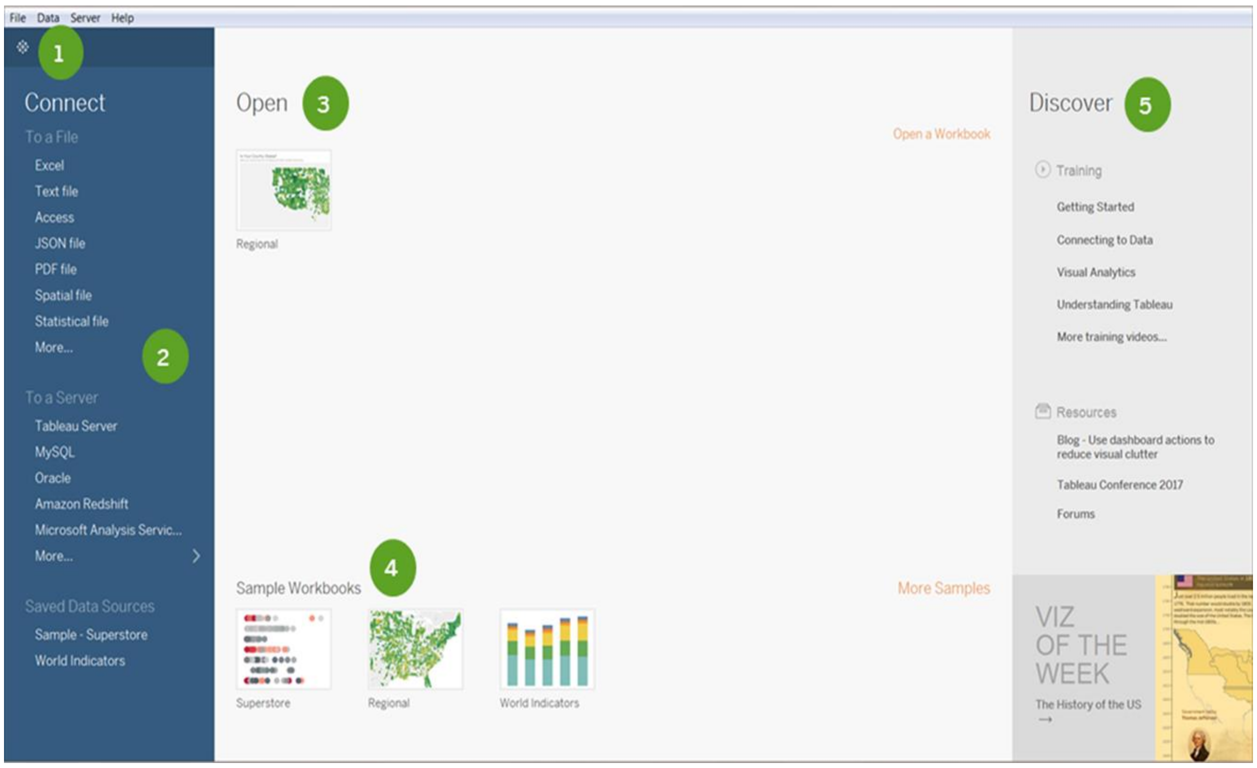
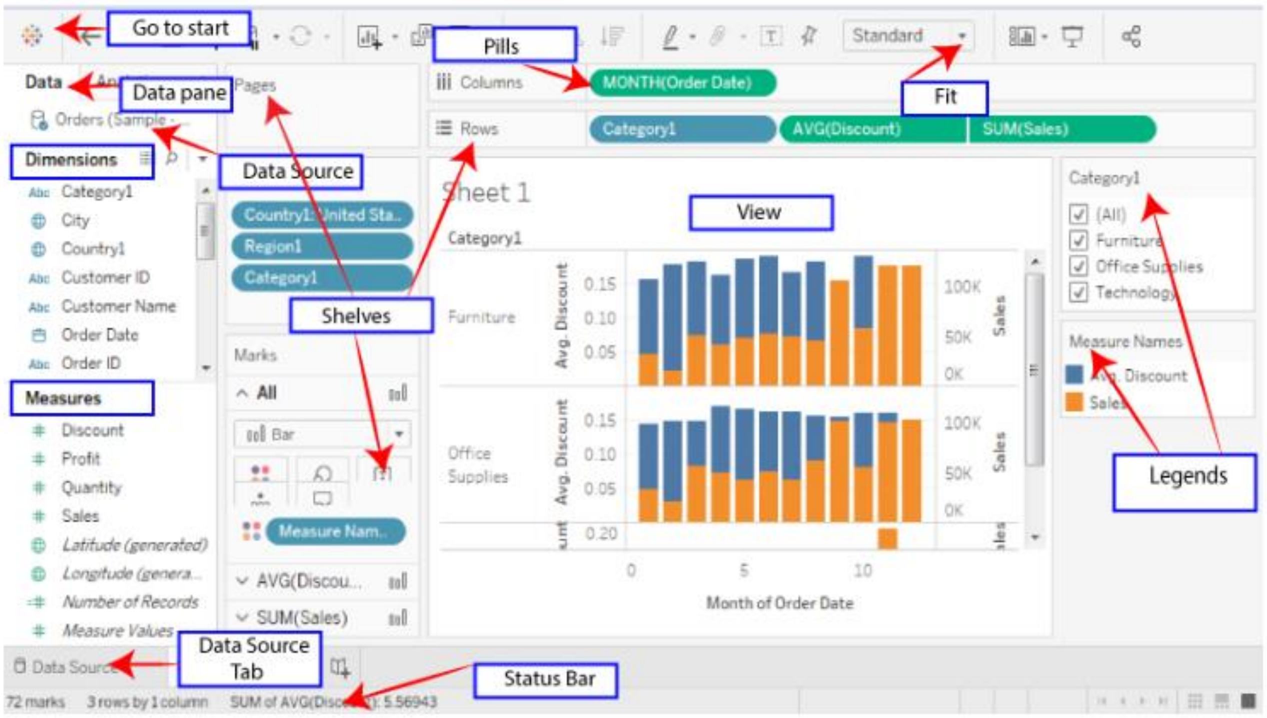


Tableau Data Terminology:

Dimension: Dimension is commonly known as a field of categorical data. Dimensions hold discrete data such as members and hierarchies that cannot be aggregated. It also contains characteristic values such as dates, names, and geographical data. The dimensions used to reveal details of your information.

Measures : contain numeric, quantitative values that you can measure. Measures can be aggregated. When you drag a measure into the view, Tableau applies an aggregation to that measure



1D (Linear) Data visualization

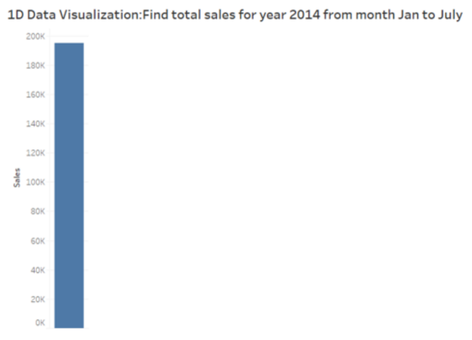
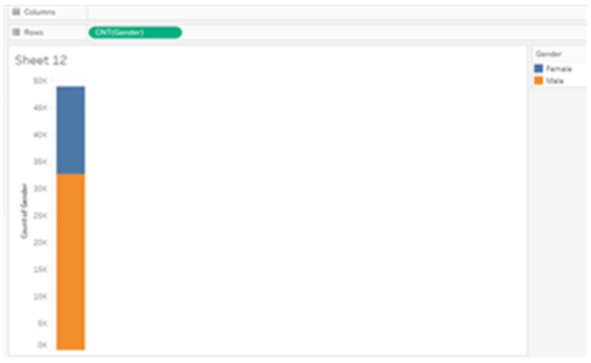
1D/Linear Data Visualization

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

Display one parameter of your data

Example:

Find total sales of United States.

1D Example: Find total sales for year 2014 from Jan to July(Samplesuperstore dataset)	Count of number of Male and Females (ADULT Dataset)
	

2D/Planner Data Visualization

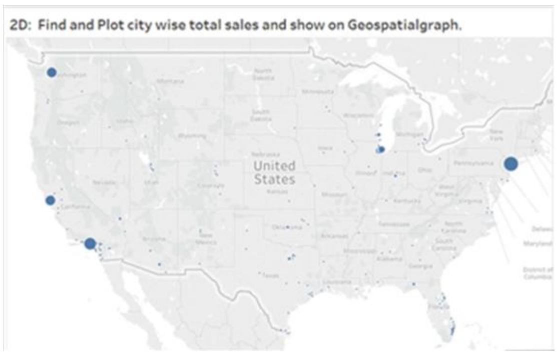
Geospatial or spatial data visualizations relate to real life physical locations, overlaying familiar maps with different data points.

These types of data visualizations are commonly used to display sales or acquisitions over time, and can be most recognizable for their use in political campaigns or to display market penetration in multinational corporations.

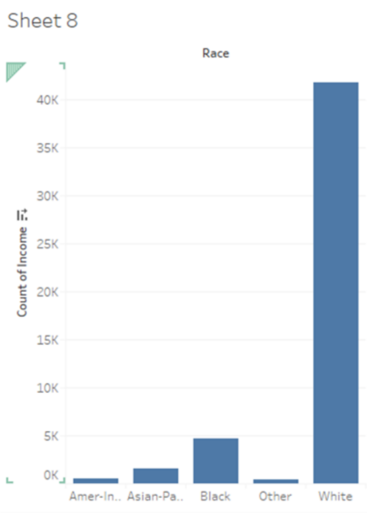
Example:

Plot state wise sales distribution.

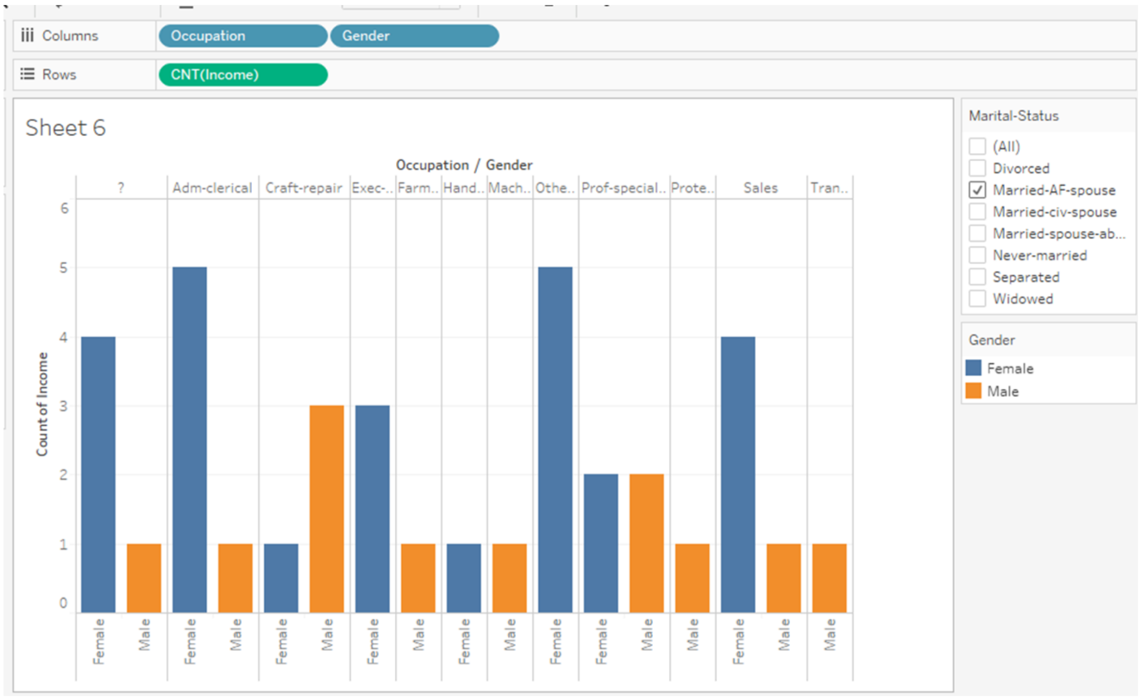
2d example1: superstore dataset


2D: Find and Plot city wise total sales and show on Geospatialgraph.
United States

income based on Race(Adult dataset)



3D Example:Ocuupation wise Income of Male and Female(ADULT dataset)



Multidimensional Visualization

Here we analyze multiple data dimensions or attributes (2 or more)

multidimensional simply means to add additional aspects or variables for consideration.

Multidimensional visualization is not only involves just checking out distributions but also potential relationships, patterns and correlations amongst these attributes.

Types of Multidimensional Visualization

Examples of visualizations that show category proportions or counts:

pie chart,

Wordles,

bar chart,

histogram,

rank plot,

tree map

Examples of visualizations that show relationships between variables:

scatter plot,

line chart,

step chart,

area chart,

heat map,

matrices,

parallel coordinates/sets,

radar/spider chart,

box and whisper plots, mosaic display, waterfall chart, pixel bar chart, tabular comparison of charts

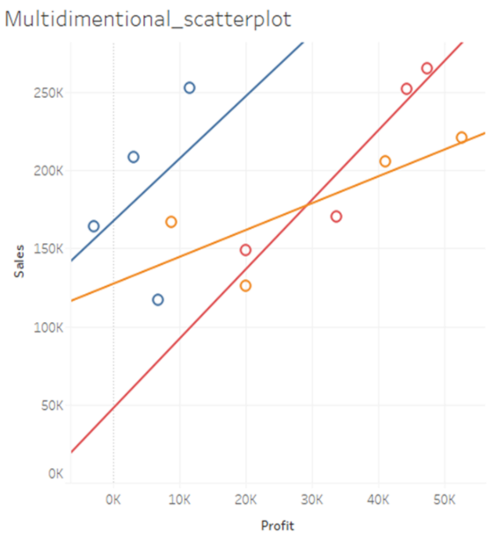
Multidimenstional Example:suersore dataset

Category

Region

Sales

Profit



Hierarchical Data Visualizations/Trees

Hierarchical Visualizations or Trees are collections of items with each item having a link to one parent item (except the root).

Items and the links between parent and child can have multiple attributes.

A tree map is a visualization that nests rectangles in hierarchies so you can compare different dimension combinations across one or two measures (one for size; one for color) and quickly interpret their respective contributions to the whole.

When used poorly, tree maps are not much more than an alternative pie chart.

When used well, they provide at least two big benefits:

Depending on the analysis, some portions of the tree map will be composed of large rectangles where additional context can be added as labels. This is beneficial when the visualization will not be interactive and you still want the written information represented.

In addition to the scatter plot, tree maps are one of the only visualization types that allow you to reasonably communicate and consume hundreds of marks on a single view. This makes it easier to spot patterns and relationships

Examples include:

dendrogram,

phylogenetic tree,

radial tree,

hyperbolic tree,

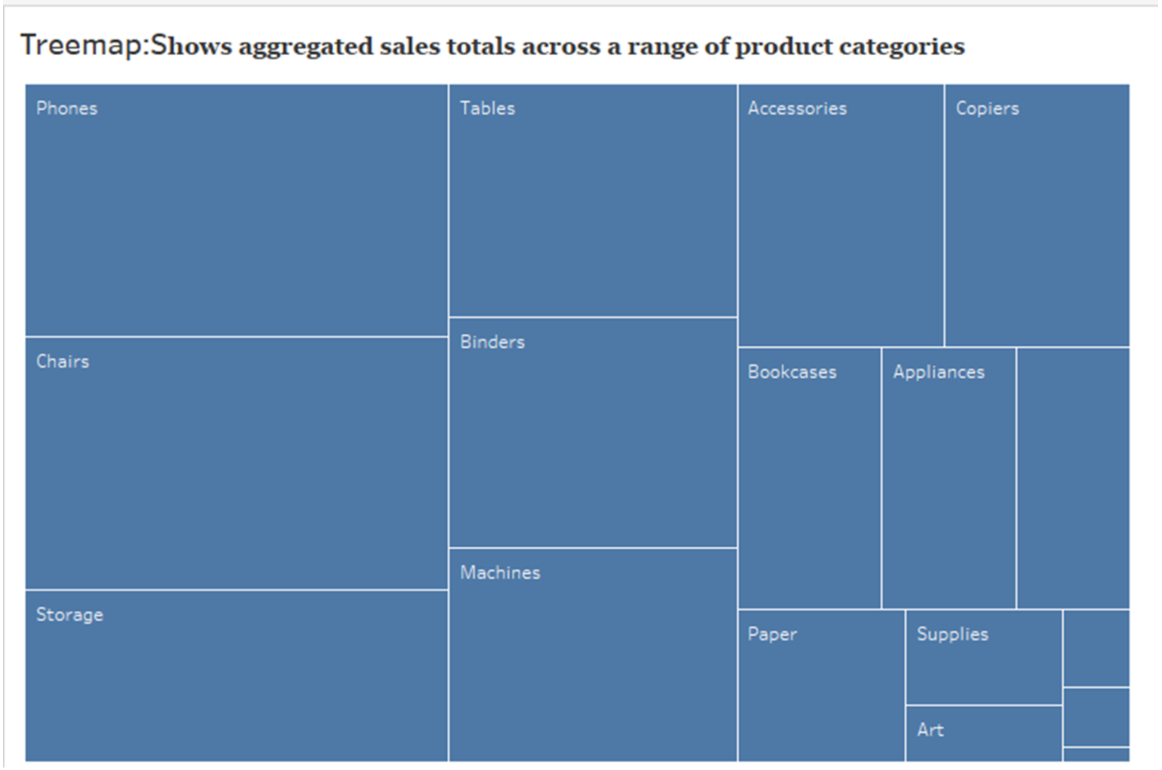
tree map,

cone tree,

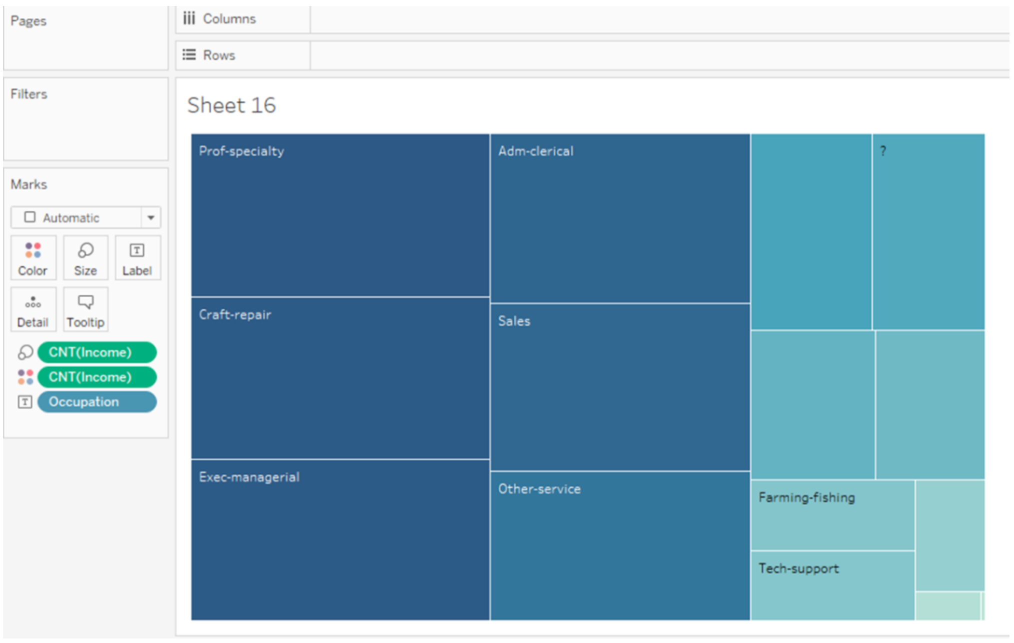
radial hierarchy,

decision tree/flow chart

Treemap example:superstore dataset



TreeMap Example : Income Generated based on Occupation(adult dataset)



Temporal Data visualization

Temporal data refers to time.

Temporal data is data that represents a state in time such as

Land use pattern of Hong-Cong in 1990

Total rainfall in US on Jul 1,2009

Temporal data is collected to analyze

weather patterns

other environmental variables

monitor traffic condition and so on

Temporal data is data in which values You see in dataset depends on time.

Example: Business sales, Natural Phenomenon, Behavior/Movement, Traffic/Mobility, Medical/Healthcare, Finance.

Types of Temporal data

Event Data :

Every object in dataset represent one single event.

Something happened at time T

Example:

Let say any message on social media at some time can be an event.

Message on facebook

Measurement Data

Time+measure

This is the value at time T

Example

From sensor temperature

From finance reveneu, stock value

Examples of temporal data visualization

Time series

Time line

Scatterplot

Line graphs

Time Series Visualization

Time series analysis is a statistical technique used to record and analyze data points over a period of time, such as daily, monthly, yearly, etc.

A time series chart is the graphical representation of the time series data across the interval period.

Provides the flexibility to reflect on historical data and analyze trends and seasonal components.

It also helps to compare

multiple dimensions over time

spot trends,

and identify seasonal patterns in the data.

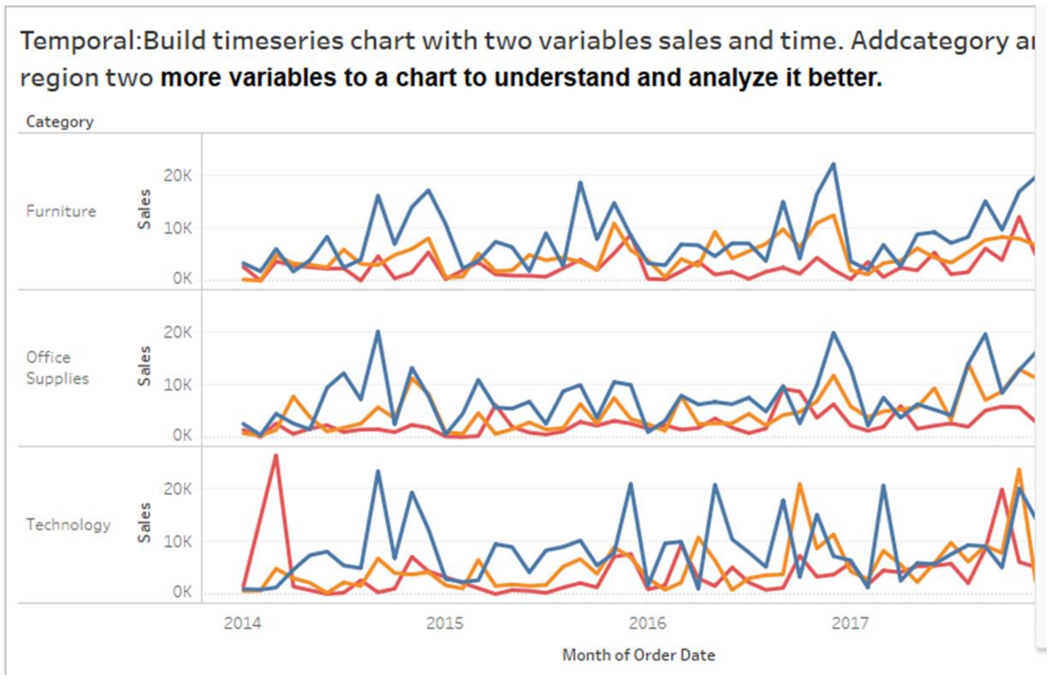
Examples include

stock market analysis,

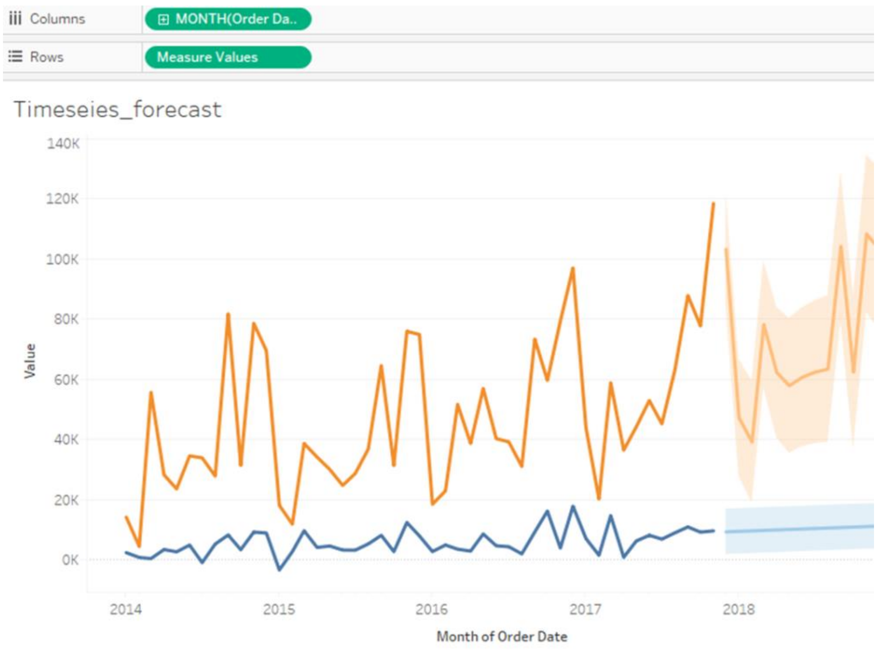
population trend analysis using a census,

sales and profit trends over time.

Timeseries Example:Superstore dataset



Timeseries forecast:superstore dataset



Network data visualization

Network visualizations display relationships between elements by linking nodes with common characteristics.

User can visualize clusters quickly and determine relationships.

Unfortunately, the development process in Tableau isn't as straightforward.

To get the data into a Tableau-friendly format for network visualization, we'll first do some data preparation then bring it into Gephi to produce spatial coordinates.

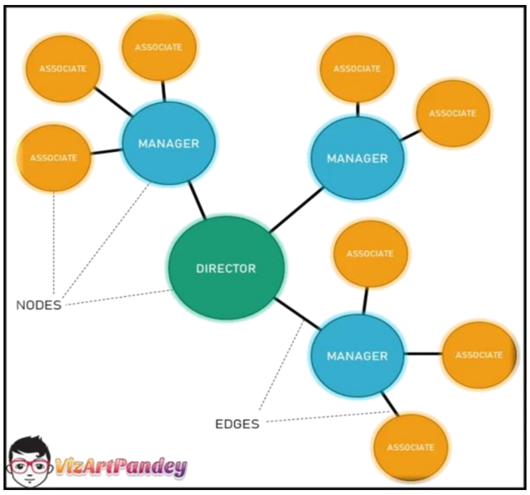
A network graph is a data visualization method that allows users to easily understand relationships in data.

Network graphs are composed of nodes and edges.

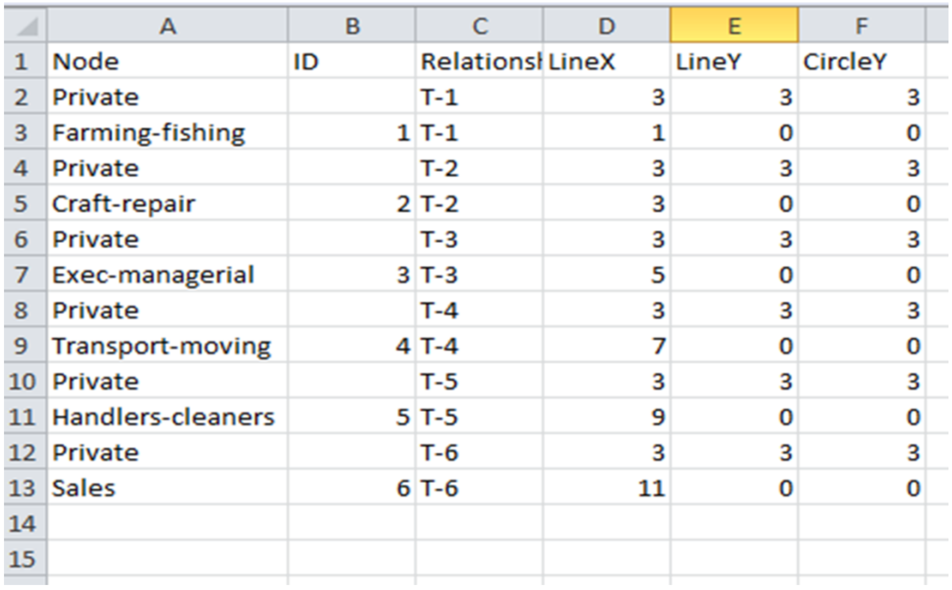
Nodes are singular data points which are connected to other nodes through edges.

Consider an organization with three employee types: directors, managers, and associates.

A network graph would show each employee type as a node, and the relationship between those individual employees as edges. es show the relationship between two or more nodes.



Dataset for network data visualization(adult dataset)



	A	B	C	D	E	F
1	Node	ID	Relationsl	LineX	LineY	CircleY
2	Private		T-1	3	3	3
3	Farming-fishing	1	T-1	1	0	0
4	Private		T-2	3	3	3
5	Craft-repair	2	T-2	3	0	0
6	Private		T-3	3	3	3
7	Exec-managerial	3	T-3	5	0	0
8	Private		T-4	3	3	3
9	Transport-moving	4	T-4	7	0	0
10	Private		T-5	3	3	3
11	Handlers-cleaners	5	T-5	9	0	0
12	Private		T-6	3	3	3
13	Sales	6	T-6	11	0	0
14						
15						

Conclusion :

Assignment 9 (Part C)

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

Code:

```
!pip install BeautifulSoup4
!pip install requests
from bs4 import BeautifulSoup as bs
import requests
link = 'https://meesho.com/classic-fancy-bedsheets/p/f6j5p?view_mode=all_r_n_r'
page = requests.get(link)
page
page.content
soup = bs(page.content,'html.parser')
print(soup.prettify())
names = soup.find_all('span',class_='Text__StyledText-sc-oo0kvp-0 cwTMA-d')
names
cust_name = []
for i in range(0,len(names)):
  cust_name.append(names[i].get_text())
```

```
publish_date = soup.find_all('span',class_='Text__StyledText-sc-oo0kvp-0 fMjoAc')
publish_date
cust_publish_date = []
for i in range(0,len(publish_date)):
  cust_publish_date.append(publish_date[i].get_text())
cust_publish_date
cust_publish_date.pop(0)
cust_publish_date
reviews = soup.find_all('span',class_='Text__StyledText-sc-oo0kvp-0 gKkBjb Comment__CommentText-sc-1ju5q0e-3 kFZtes Comment__CommentText-sc-1ju5q0e-3 kFZtes')
reviews
cust_review = []
for i in range(0,len(reviews)):
  cust_review.append(reviews[i].get_text())
cust_review
review_rate = soup.find_all('span',class_='Text__StyledText-sc-oo0kvp-0 gYxLUd')
review_rate
cust_review_rate = []
for i in range(0,len(review_rate)):
  cust_review_rate.append(review_rate[i].get_text())
cust_review_rate
cust_name
cust_publish_date
cust_review
cust_review_rate
```

Assignment 10 (Part C)

Develop a mini project in a group of 4-5 students using different predictive models techniques to solve any real life problem. (Refer link dataset- https://www.kaggle.com/tanmoyie/us-graduate-schools- admissionparameters) and students should submit the document related to it as part of journal.

Reference Books:

1. Big Data, Black Book, DT Editorial services, 2015 edition.
2. Data Analytics with Hadoop, Jenny Kim, Benjamin Bengfort, OReilly Media, Inc.
3. Python for Data Analysis by Wes McKinney published by O' Reilly media, ISBN : 978-1-449-31979-3.
4. Python Data Science Handbook by Jake VanderPlas https://tanthiamhuat.files.wordpress.com/2018/04/pythondatasciencehandbook.pdf
5. Alex Holmes, Hadoop in practice, Dreamtech press.
6. Online References for data set http://archive.ics.uci.edu/ml/ https://www.kaggle.com/tanmoyie/us-graduate-schools-admission-parameters https://www.kaggle.com