

**Marathwada Mitra Mandal's
College of Engineering, Pune**

Karvenagar, Pune-411 052
Accredited with 'A' Grade by NAAC



Department of Information Technology

Lab Manual

314457 – Data Science & Big Data Analytics Lab

Preface

Data science is an interdisciplinary academic field that uses statistics, scientific computing, scientific methods, processes, algorithms and systems to extract or extrapolate knowledge and insights from noisy, structured and unstructured data. Data science also integrates domain knowledge from the underlying application domain (e.g., natural sciences, information technology, medicine). Data science is multifaceted and can be described as a science, as a research paradigm, as a research method, as a discipline, as a workflow, and as a profession.

Data science is an interdisciplinary field focused on extracting knowledge from typically large data sets and applying the knowledge and insights from that data to solve problems in a wide range of application domains. The field encompasses preparing data for analysis, formulating data science problems, analyzing data, developing data-driven solutions, and presenting findings to inform high-level decisions in a broad range of application domains. As such, it incorporates skills from computer science, statistics, information science, mathematics, data visualization, information visualization, data sonification, data integration, graphic design, complex systems, communication and business. In 2015, the American Statistical Association identified database management, statistics and machine learning, and distributed and parallel systems as the three emerging foundational professional communities.

Big data primarily refers to data sets that are too large or complex to be dealt with by traditional data-processing application software. Data with many fields (rows) offer greater statistical power, while data with higher complexity (more attributes or columns) may lead to a higher false discovery rate. Big data analysis challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy, and data source. Big data was originally associated with three key concepts: volume, variety, and velocity. The analysis of big data presents challenges in sampling, and thus previously allowing for only observations and sampling.

The final outcome of DS&BDA study is information that will help you make the right kind of analysis with the help of available data and correct tools. Keeping this in mind, course outcomes and lab assignments are prepared. It gives the complete understanding and utility of Data Science and Big Data domain. Proper mapping is also done with Program Outcomes and Program Specific Outcomes. This clearly states the goals for the students in terms of their skill development.



‘येथे बहुतांचे हित ।’

Marathwada Mitra Mandal's
COLLEGE OF ENGINEERING
Karvenagar, Pune

Vision

To aspire for the Welfare of Society
through excellence in Science and Technology.



Mission

Our Mission is to

- **M**ould young talent for higher endeavours.
- **M**eeet the challenges of globalization.
- **C**ommit for social progress with values and ethics.
- **O**rient faculty and students for research and development.
- **E**mphasize excellence in all disciplines.



A. Course Outcome

Course Outcome	Statement
	<i>At the end of the course, a student will be able to (write/install/solve/apply)</i>
314457.1	Apply Big data primitives and fundamentals for application development.
314457.2	Explore different Big data processing techniques with use cases.
314457.3	Apply the Analytical concept of Big data using Python.
314457.4	Visualize the Big Data using Tableau.
314457.5	Design algorithms and techniques for Big data analytics.
314457.6	Design and develop Big data analytic applications for emerging trends.

B. CO-PO mapping

Course Outcome	Program outcomes											
	1	2	3	4	5	6	7	8	9	10	11	12
314457.1	3	3	3	3	3				3	1	1	3
314457.2	3	3	3	3	3				3	1	1	3
314457.3	3	3	3	3	3				3	1	1	3
314457.4	3	3	3	3	3				3	3	1	3
314457.5	3	3	3	3	3				3	1	1	3
314457.6	3	3	3	3	3				3	1	1	3
Average	3	3	3	3	3				3	2.66	1	3

CO-PSO mapping

Course Outcome	Program Specific Outcomes	
	1	2
314457.1	1	3
314457.2	1	3
314457.3	1	3
314457.4	1	3
314457.5	1	3
314457.6	1	3
Average	1	3

I N D E X

SN.	Experiment	Page	Date of conduction	Date of completion	CAS	Signature of Faculty
1	Single node/Multiple node Hadoop Installation.					
2	Design a distributed application using MapReduce(Using Java) which processes a log file of a system					
3	Write an application using HiveQL for flight information system with several database operations					
4	Perform the primary data manipulation operations using Python on the Facebook metrics data sets					
5	Perform other primary data manipulation operations using Python on the Air quality and Heart Diseases data sets					
6	Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs					
7	Perform the data visualization operations using Tableau on Adult and Iris datasets.					
8	Create a review scrapper for any ecommerce website to fetch real time comments, reviews, ratings, comment tags, customer name using Python					
9	Develop a mini project in a group using different predictive models techniques to solve any real life problem.					

CERTIFICATE

Certified that Mr./Ms. _____ of Class TE Electrical Engineering Roll No. _____ has completed the Tutorial work in the subject **DS & BDA LAB (314457)** during the academic year 2022-23 Sem II.

Signature of the Faculty

Signature of Head of the Department

Date:

Assignment No 1

Title: Hadoop Installation on a)Single Node b)Multiple Node

CO Achieved: CO6

Theory:

What is Hadoop?

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing). It is used for batch/offline processing. It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Moreover it can be scaled up just by adding nodes in the cluster.

Modules of Hadoop:

1. **HDFS:** Hadoop Distributed File System. Google published its paper GFS and on the basis of that HDFS was developed. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.
2. **Yarn:** Yet another Resource Negotiator is used for job scheduling and manage the cluster.
3. **Map Reduce:** This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.
4. **Hadoop Common:** These Java libraries are used to start Hadoop and are used by other Hadoop modules.

Advantages of Hadoop:

- **Fast:** In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval. Even the tools to process the data are often on the same servers, thus reducing the processing time. It is able to process terabytes of data in minutes and Peta bytes in hours.
- **Scalable:** Hadoop cluster can be extended by just adding nodes in the cluster.
- **Cost Effective:** Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.
- **Resilient to failure:** HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop

takes the other copy of data and use it. Normally, data are replicated thrice but the replication factor is configurable.¹

Steps to install single node Hadoop:

1. You must have JDK 1.7 or more installed on UBUNTU 14.04, To check just issue a command: `~$ java -version`

To install jdk just type command: `sudo apt-get install default-jdk`

2. Add a dedicated Hadoop user:

2.1: `sudo addgroup hadoop`

2.2: `sudo adduser --ingroup hadoop hduser`

2.3: `sudo adduser hduser sudo`

3. Install SSH:

ssh has two main components:

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

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

3.3 The ssh is pre-enabled on Linux, but in order to start sshd daemon, we need to install ssh first. Use this command to do that : `sudo apt-get install openssh-server`

To check if ssh is installed properly, issue following commands:

i. `which ssh`

ii. `which sshd`

3.4 Create and Setup SSH Certificates

i. `su hduser`

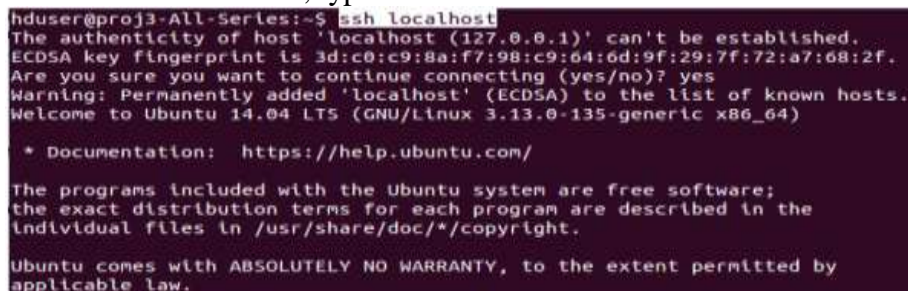
ii. `cd`

iii. `ssh-keygen -t rsa -P ""`

- If asked for a filename just leave it blank and press the enter key to continue

iv. `cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys`

v. To check if ssh works, type: `ssh localhost`



```
hduser@proj3-All-Series:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is 3d:c0:c9:8a:f7:98:c9:64:6d:9f:29:7f:72:a7:68:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-135-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

¹ <https://www.javatpoint.com/hadoop-installation>

vi. exit

4. Install Hadoop:

4.1 Download & extract:

- Go to Link: <http://www-eu.apache.org/dist/hadoop/common/> and download Hadoop 2.9 version `hadoop-2.9.0.tar.gz`

- go the location where the downloaded file is present and extract using below command

- `tar -xvf hadoop-2.9.0.tar.gz`

4.2 move the Hadoop installation to the `/usr/local/hadoop` directory using the following command: `sudo mv hadoop-2.9.0 /usr/local/hadoop`

4.3 Now, change the owner of hadoop folder using command:

- `sudo chown -R hduser /usr/local/`

5. Setup Configuration Files

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

i. `~/.bashrc`:

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

`update-alternatives --config java`

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

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

Next issue the following command: `source ~/.bashrc`

note that the `JAVA_HOME` should be set as the path just before the `'.../bin/`:

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

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

```
vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

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

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

The /usr/local/hadoop/etc/hadoop/core-site.xml file contains configuration properties that Hadoop uses when starting up.

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

```
~$ sudo gedit /usr/local/hadoop/etc/hadoop/core-site.xml

<property>

  <name>fs.default.name</name>

  <value>hdfs://localhost:9000</value>

</property>
```

iv. /usr/local/hadoop/etc/hadoop/mapred-site.xml

By default, the /usr/local/hadoop/etc/hadoop/ folder contains

/usr/local/hadoop/etc/hadoop/mapred-site.xml.template file which has to be renamed/copied with the name mapred-site.xml using command:

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

The mapred-site.xml file is used to specify which framework is being used for MapReduce.

We need to enter the following content in between the <configuration></configuration> tag:

```
<property>

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

  <value>yarn</value>

</property>
```

v. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The /usr/local/hadoop/etc/hadoop/hdfs-site.xml file needs to be configured for each host in the cluster that is being used. It is used to specify the directories which will be used as the namenode and the datanode on that host.

Before editing this file, we need to create two directories which will contain the namenode and the datanode for this Hadoop installation. This can be done using the following commands:

```
~$ sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
```

```
~$ sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
```

```
~$ sudo chown -R hduser:hadoop /usr/local/hadoop_tmp
```

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

```
~$ sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml

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

vi. Edit the yarn-site.xml file and paste the code given below in <configuration> tags.

```
sudo gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml

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

```
<name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>
```

6. Format the New Hadoop Filesystem

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

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

```
hduser@proj3-All-Series:~$ hdfs namenode -format
18/01/01 18:59:58 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = proj3-All-Series/127.0.1.1
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 2.9.0
STARTUP_MSG: classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/commons-collectio
ns-3.2.2.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beanutils-1.7.0.jar:/usr/local/hadoop/share/hadoop/c
ommon/lib/apachedfs-kerberos-codec-2.0.0-M15.jar:/usr/local/hadoop/share/hadoop/common/lib/json-smart-1.1.1.jar:/usr
/local/hadoop/share/hadoop/common/lib/jettison-1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/avro-1.7.7.jar:/us
r/local/hadoop/share/hadoop/common/lib/jets3t-0.9.0.jar:/usr/local/hadoop/share/hadoop/common/lib/guava-11.0.2.jar:/
usr/local/hadoop/share/hadoop/common/lib/jackson-core-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/lib/asm
-3.2.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beanutils-core-1.8.0.jar:/usr/local/hadoop/share/hadoop/
common/lib/jersey-server-1.9.jar:/usr/local/hadoop/share/hadoop/common/lib/java-xmlbuilder-0.4.jar:/usr/local/hadoc
p/share/hadoop/common/lib/curator-recipes-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/zookeeper-3.4.6.jar:/
usr/local/hadoop/share/hadoop/common/lib/activation-1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/log4j-1.2.17.
jar:/usr/local/hadoop/share/hadoop/common/lib/snappy-java-1.0.5.jar:/usr/local/hadoop/share/hadoop/common/lib/commo
ns-net-3.1.jar:/usr/local/hadoop/share/hadoop/common/lib/hamcrest-core-1.3.jar:/usr/local/hadoop/share/hadoop/commo
n/lib/jsr305-3.0.0.jar:/usr/local/hadoop/share/hadoop/common/lib/woodstox-core-5.0.3.jar:/usr/local/hadoop/share/ha
dooop/common/lib/jetty-util-6.1.26.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-sslengine-6.1.26.jar:/usr/loc
al/hadoop/share/hadoop/common/lib/jackson-xc-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/lib/curator-framework
-2.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-jaxrs-1.9.13.jar:/usr/local/hadoop/share/hadoop/common
/lib/jaxb-api-2.2.2.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-3.6.2.Final.jar:/usr/local/hadoop/share/had
```

```
hduser@proj3-All-Series:~$
18/01/01 18:59:59 INFO util.GSet: capacity = 2^18 = 262144 entries
18/01/01 18:59:59 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.window.num.buckets = 10
18/01/01 18:59:59 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.num.users = 10
18/01/01 18:59:59 INFO metrics.TopMetrics: NNTop conf: dfs.namenode.top.windows.minutes = 1,5,25
18/01/01 18:59:59 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
18/01/01 18:59:59 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total heap and retry cache entry expiry
time is 600000 millis
18/01/01 18:59:59 INFO util.GSet: Computing capacity for map NameNodeRetryCache
18/01/01 18:59:59 INFO util.GSet: VM type = 64-bit
18/01/01 18:59:59 INFO util.GSet: 0.0299999999329447746% max memory 889 MB = 273.1 KB
18/01/01 18:59:59 INFO util.GSet: capacity = 2^15 = 32768 entries
18/01/01 18:59:59 INFO namenode.FSImage: Allocated new BlockPoolId: BP-120182733-127.0.1.1-1514813399571
18/01/01 18:59:59 INFO common.Storage: Storage directory /usr/local/hadoop_tnp/hdfs/namenode has been successfully
formatted.
18/01/01 18:59:59 INFO namenode.FSImageFormatProtobuf: Saving image file /usr/local/hadoop_tnp/hdfs/namenode/curren
t/fsimage.ckpt.00000000000000000000 using no compression
18/01/01 18:59:59 INFO namenode.FSImageFormatProtobuf: Image file /usr/local/hadoop_tnp/hdfs/namenode/current/fsima
ge.ckpt.00000000000000000000 of size 323 bytes saved in 0 seconds.
18/01/01 18:59:59 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
18/01/01 18:59:59 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at proj3-All-Series/127.0.1.1
hduser@proj3-All-Series:~$
```

Note that `hadoop namenode -format` command should be executed once before we start using Hadoop. If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

7. Starting Hadoop

Now it's time to start the newly installed single node cluster.

We can use `start-all.sh` or (`start-dfs.sh` and `start-yarn.sh`)

```
k@laptop:~$ cd /usr/local/hadoop/sbin

k@laptop:/usr/local/hadoop/sbin$ ls
distribute-exclude.sh  start-all.cmd      stop-balancer.sh
hadoop-daemon.sh       start-all.sh       stop-dfs.cmd
hadoop-daemons.sh     start-balancer.sh   stop-dfs.sh
hdfs-config.cmd        start-dfs.cmd       stop-secure-dns.sh
hdfs-config.sh         start-dfs.sh        stop-yarn.cmd
httpfs.sh              start-secure-dns.sh stop-yarn.sh
kms.sh                 start-yarn.cmd      yarn-daemon.sh
mr-jobhistory-daemon.sh start-yarn.sh       yarn-daemons.sh
refresh-namenodes.sh   stop-all.cmd
slaves.sh              stop-all.sh

k@laptop:/usr/local/hadoop/sbin$ sudo su hduser

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

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

```
hduser@laptop:/usr/local/hadoop/sbin$ jps
9026 NodeManager
7348 NameNode
```

```
9766 Jps
8887 ResourceManager
7507 DataNode
```

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

8. Stopping Hadoop

```
$ pwd
/usr/local/hadoop/sbin

$ ls
distribute-exclude.sh  httpfs.sh  start-all.sh
start-yarn.cmd  stop-dfs.cmd  yarn-daemon.sh
hadoop-daemon.sh  mr-jobhistory-daemon.sh  start-balancer.sh
start-yarn.sh  stop-dfs.sh  yarn-daemons.sh
hadoop-daemons.sh  refresh-namenodes.sh  start-dfs.cmd
stop-all.cmd  stop-secure-dns.sh
hdfs-config.cmd  slaves.sh  start-dfs.sh
stop-all.sh  stop-yarn.cmd
hdfs-config.sh  start-all.cmd  start-secure-
dns.sh  stop-balancer.sh  stop-yarn.sh
```

We run stop-all.sh or (stop-dfs.sh and stop-yarn.sh) to stop all the daemons running on our machine:

```
hduser@laptop:/usr/local/hadoop/sbin$ pwd
/usr/local/hadoop/sbin
hduser@laptop:/usr/local/hadoop/sbin$ ls
distribute-exclude.sh  httpfs.sh  start-all.cmd
start-secure-dns.sh  stop-balancer.sh  stop-yarn.sh
hadoop-daemon.sh  kms.sh  start-all.sh
start-yarn.cmd  stop-dfs.cmd  yarn-daemon.sh
hadoop-daemons.sh  mr-jobhistory-daemon.sh  start-balancer.sh
start-yarn.sh  stop-dfs.sh  yarn-daemons.sh
```

```

hdfs-config.cmd          refresh-namenodes.sh      start-dfs.cmd
stop-all.cmd            stop-secure-dns.sh

hdfs-config.sh           slaves.sh                start-dfs.sh
stop-all.sh             stop-yarn.cmd

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

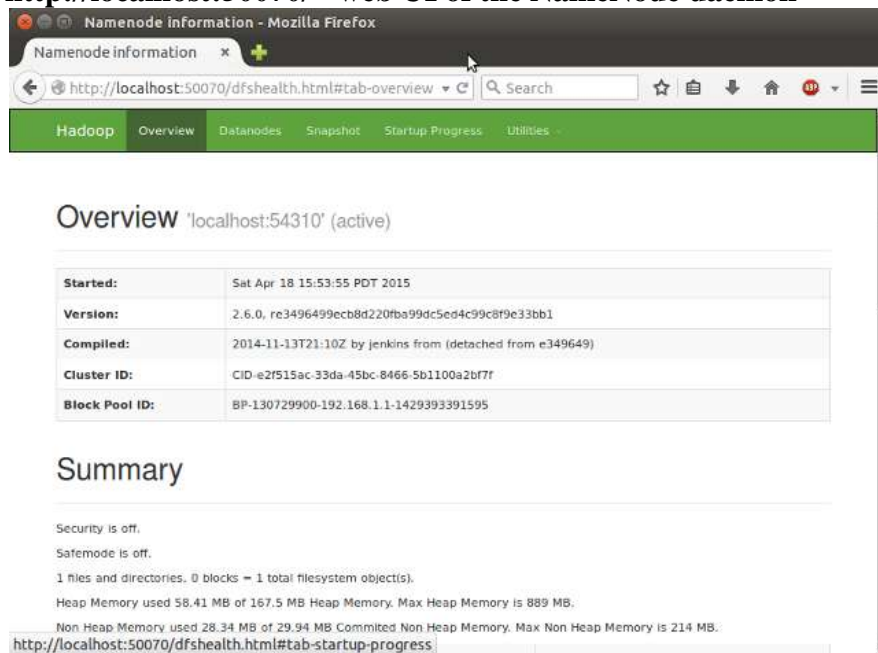
```

9. Hadoop Web Interfaces

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

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

<http://localhost:50070/> - web UI of the NameNode daemon



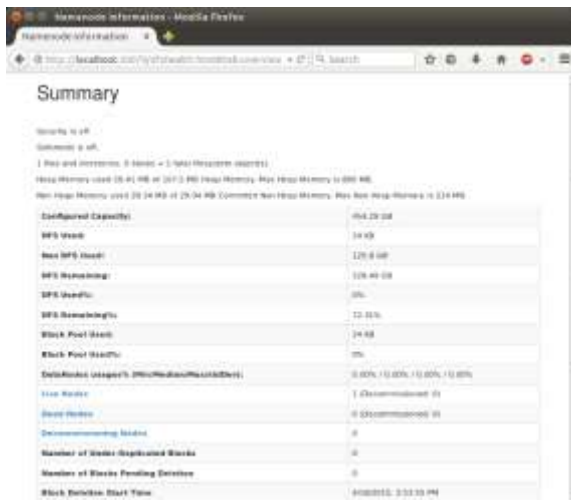
Overview 'localhost:54310' (active)

Started:	Sat Apr 18 15:53:55 PDT 2015
Version:	2.6.0, rc3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)
Cluster ID:	CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f
Block Pool ID:	BP-130729900-192.168.1.1-1429393391595

Summary

Security is off.
 Safemode is off.
 1 files and directories, 0 blocks = 1 total filesystem object(s).
 Heap Memory used 58.41 MB of 167.5 MB Heap Memory. Max Heap Memory is 889 MB.
 Non Heap Memory used 28.34 MB of 29.94 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

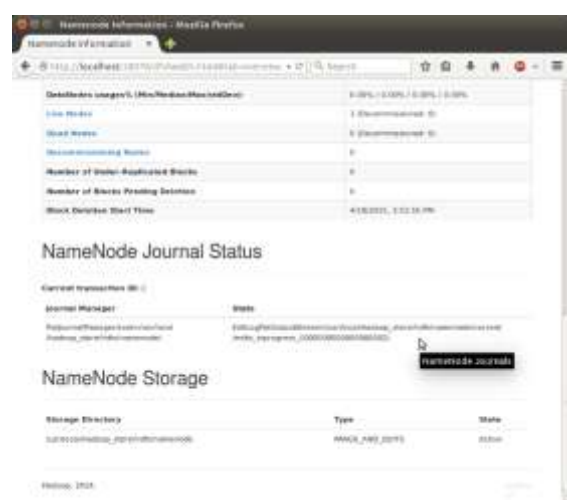
<http://localhost:50070/dfshealth.html#tab-startup-progress>



Summary

Reports to JFS
 Reports to JFS
 1 Rep and Secondary, 0 Nodes = 1 Total Replicas (replicas)
 Heap Memory used (5.41 MB) at (2013 MB) Heap Memory: Max Heap Memory is 800 MB
 Max Heap Memory used (20.34 MB) at (20.34 MB) Committed Max Heap Memory: Max Max Heap Memory is 124 MB

Configuration Property	Value
DFS Used	14 KB
Max DFS Used	126.8 MB
DFS Remaining	126.40 MB
DFS Used%	0%
DFS Remaining%	12.15%
Block Pool Used	24 KB
Block Pool Remaining	0%
Datanode usage% (Min/Max/MaxUsed)	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	2015-04-18 15:33:39 PM



NameNode Journal Status

Current transaction ID: 0

Journal Manager	State
PrimaryJournalManager (local)	Active
SecondaryJournalManager (remote)	Active

NameNode Storage

Storage Directory	Type	State
hdfs://localhost:8020/	HDFS	Active

SecondaryNameNode



SecondaryNameNode

Version:	2.6.0, e3496499ecb8d220fba99dc5ed4c99c8f9e33bb1
Compiled:	2014-11-13T21:10Z by jenkins from (detached from e349649)

SecondaryNameNode Status

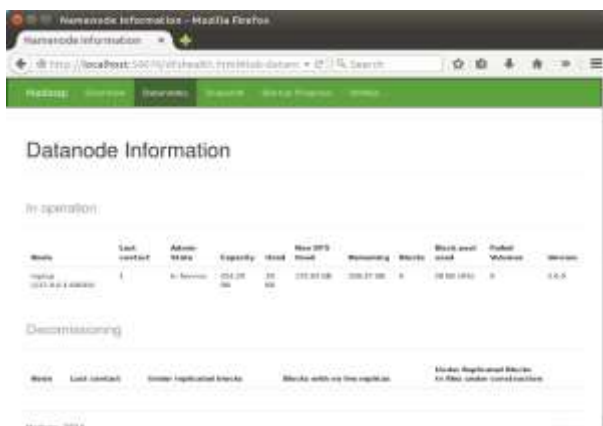
Name Node Address : localhost/127.0.0.1:54310
 Start Time : Sat Apr 18 16:43:38 PDT 2015
 Last Checkpoint : 79 seconds ago
 Checkpoint Period : 3600 seconds
 Checkpoint Transactions : 1000000
 Checkpoint Dirs : [file:///app/hadoop/tmp/dfs/namesecondary]
 Checkpoint Edits Dirs : [file:///app/hadoop/tmp/dfs/namesecondary]

Logs

Hadoop, 2015.

(Note) U have to restart Hadoop to get this Secondary Namenode.

DataNode



Datanode Information

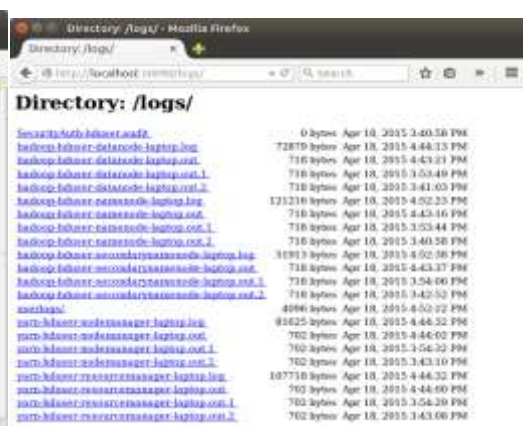
In operation:

Block	Last contact	Admin-Status	Capacity	Used	Max DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
hdfs://localhost:8020/	1	In Service	128.00 MB	20.34 MB	107.66 MB	107.66 MB	0	107.66 MB	0	0.0.0

Decommissioning:

Block	Last contact	Group: Replicated blocks	Blocks with no live replicas	Under-Replicated Blocks: An-Raw under-replicated replicas

Hadoop, 2015.



Directory: /logs/

File	Size	Time
hadoop-hdfs-datanode-journal.log	0 bytes	Apr 18, 2015 3:40:58 PM
hadoop-hdfs-datanode-journal.log	72879 bytes	Apr 18, 2015 4:44:13 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 4:43:31 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 3:53:40 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 3:41:03 PM
hadoop-hdfs-datanode-journal.log	121210 bytes	Apr 18, 2015 4:02:25 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 4:43:16 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 3:53:44 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 3:40:58 PM
hadoop-hdfs-datanode-journal.log	31913 bytes	Apr 18, 2015 4:02:36 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 4:43:37 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 3:54:06 PM
hadoop-hdfs-datanode-journal.log	716 bytes	Apr 18, 2015 3:42:52 PM
hadoop-hdfs-datanode-journal.log	4096 bytes	Apr 18, 2015 4:02:22 PM
hadoop-hdfs-datanode-journal.log	81623 bytes	Apr 18, 2015 4:44:52 PM
hadoop-hdfs-datanode-journal.log	702 bytes	Apr 18, 2015 4:44:02 PM
hadoop-hdfs-datanode-journal.log	702 bytes	Apr 18, 2015 3:54:32 PM
hadoop-hdfs-datanode-journal.log	702 bytes	Apr 18, 2015 3:43:10 PM
hadoop-hdfs-datanode-journal.log	107710 bytes	Apr 18, 2015 4:44:32 PM
hadoop-hdfs-datanode-journal.log	702 bytes	Apr 18, 2015 4:44:00 PM
hadoop-hdfs-datanode-journal.log	702 bytes	Apr 18, 2015 3:54:10 PM
hadoop-hdfs-datanode-journal.log	702 bytes	Apr 18, 2015 3:43:06 PM

Steps to install Multi Node Hadoop Cluster:

For simplicity, we use 1 node as master and 1 as slave.

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.

1. Install Java (as done previously) on all the slaves and master.

2. Disable IPv6 (if applicable) using command below:

```
sudo sed -i 's/net.ipv6.conf.all.disable_ipv6 = 1/net.ipv6.conf.all.disable_ipv6 = 0/' \
/etc/sysctl.d/bindv6only.conf && sudo invoke-rc.d procps restart
```

3. Setting up Hadoop User:

Hadoop talks to other nodes in the cluster using no-password ssh. 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.

```
# 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
```

Run the below two commands on each slave node:

```
#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
```

4. Download and Install Hadoop binaries on Master and Slave nodes

Do this step on master and every slave node. Extract and copy the hadoop folder to path /usr/local/hadoop

5. Setup Hadoop Environment on Master and Slave Nodes

5.1 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=/usr/local/hadoop
# Set JAVA_HOME
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
# Add Hadoop bin and sbin directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

5.2 Update JAVA_HOME in /usr/local/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-openjdk-amd64
```

6. 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

<property>
  <name>hadoop.tmp.dir</name>
  <value>/usr/local/tmp</value>
  <description>Temporary Directory.</description>
</property>

<property>
  <name>fs.defaultFS</name>
  <value>hdfs://IP_address_of_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
```

```
<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 we will be 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 we manually created this directory 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 we manually created this directory 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>/usr/local/hadoop_tmp/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>/usr/local/hadoop_tmp /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>IP_of_Master:8030</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value> IP_of_Master:8032</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address</name>
  <value> IP_of_Master:8088</value>
</property>
<property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value> IP_of_Master:8031</value>
</property>
<property>
  <name>yarn.resourcemanager.admin.address</name>
  <value> IP_of_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.

```

/home/hadoopuser/hadoop/etc/hadoop/slaves
Master
slave-1
slave-2

```

7. Format the Namenode

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

```
hdfs namenode -format
```

8. Start the Distributed Format System

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

```
$ sudo /usr/local/hadoop/sbin/start-dfs.sh
```

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.

9. Start the Yarn MapReduce Job tracker

Run the following command to start the Yarn mapreduce framework.

```
$ sudo /usr/local/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 to troubleshoot

Conclusion: Single node and multi-node installation of hadoop is completed successfully. With hadoop version 2.9.0.

FAQ's:

1. What is SSH? Explain the use of SSH in hadoop.
2. What are the basic differences between relational database and HDFS?
3. What will you do when NameNode is down?

Assignment No 2

Title: 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.

CO Achieved: CO6

Theory:

Map-Reduce programming model:

Mapreduce is a framework for processing big data in two phases Map & Reduce. Both the phases have key-value pairs as input and output. Map phase implements Mapper function, in which user-provided code will be executed on each key-value pair (k1, v1) read from the input files. The output of the mapper function would be zero or more key-value pairs (k2, v2) which are called intermediate pairs. Here the key is what the data will be grouped on and the value is the information related to the analysis in the reducer.

Reduce phase takes mapper output (grouped key-value data) (k2, v2) and runs reduce function on each key-value group. reduce function iterates over the list of values associated with a key and produces outputs like aggregations, statistics etc.. Once the reduce function is done, it sends zero or more key-value pairs (k3, v3) to the final the output file. By default, Mapreduce input and output file formats are text file formats.

Map-Reduce Programming model in Java:

In order to express the above functionality in code, we need three things: A map () function, reduce () function and some driver code to run the job. an abstract map () function is present in Mapper class and reduce () function in Reducer class. These Mapper and Reducer classes are provided by Hadoop Java API. And any specific mapper/reducer implementations should be subclass these classes and override the abstract functions map () and reduce ().

Below is the code snippet for sample mapper & reducer classes:

```
public class ExampleMapper extends Mapper<K1, V1, K2, V2>
{
    void map(K1 key, V1 value, Context context) throws IOException, InterruptedException
    {..}
}
```

```

public class ExampleReducer extends Reducer<K2, V2, K3, V3>
{
    void reduce(K1 key, Iterable<V2> values, Context context) throws IOException,
    InterruptedException
    {..}
}

```

The driver is the main part of Mapreduce job and it communicates with Hadoop framework and specifies the configuration elements needed to run a mapreduce job. It is the place where programmer specifies which mapper/reducer classes a mapreduce job should run and also input/output file paths along with their formats. There is no default parent Driver class provided by Hadoop; the driver logic usually exists in the main method of the class.

Below is the code snippet of example driver:

```

public class ExampleDriver
{
    ...
    public static void main(String[] args) throws Exception
    {
        //Create a Configuration Object to set job level configurations.
        Configuration conf = new Configuration() ;
        //Set the job name
        Job job = new Job(conf, "ExampleJob") ;
        //Set the driver class in job jar file
        job.setJarByClass(ExampleDriver.class) ;
        //Set Mapper class that should be used by job
        job.setMapperClass(ExampleMapper.class) ;
        //Set Reducer class that should be used by job
        job.setReducerClass(ExampleReducer.class) ;
        // Set the data types for the final output key and values from reducer
        job.setOutputKeyClass(Text.class) ;
        job.setOutputValueClass(IntWritable.class) ;
        // Set input and output file paths to be processed by job
        FileInputFormat.addInputPath(job, new Path(args[0])) ;
        FileOutputFormat.setOutputPath(job, new Path(args[1]))
        // Execute the job and wait for it to complete
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

On high level, the user of mapreduce framework needs to specify the following things:

- The job's input location(s) in the distributed file system.
- The job's output location in the distributed file system.
- The input format.
- The output format.
- The class containing the map function.
- The class containing the reduce function but it is optional.
- The JAR file containing the mapper and reducer classes and driver classes.

Implementation Steps:

1. Create folder Data on Desktop and copy all given files
 - i. access_log_short.txt – Data set to analyze
 - ii. SalesCountryDriver.java : Manages input and output from Mapper and Reducer
 - iii. SalesMapper.java : Reads input file
 - iv. SalesCountryReducer.java : Reads the input from mapper file)
2. Open file named access_log_short.txt using libreoffice/wps and use separator as “-” (Uncheck remaining options), save the file as text CSV format
3. su – hduser (Change the user)
4. Sudo mkdir analyzelogs (Create folder analyzelogs)
and copy the contents of Data folder into analyzelogs folder using command
5. sudo cp /home/ll01/Desktop/Data/* ~/analyzelogs
6. Give permissions using
sudo chmod -R 777 analyzelogs /
7. Change the owner to hduser using following
sudo chown -R hduser analyzelogs /
8. cd analyzelogs
9. sudo chmod +r *.*
10. export CLASSPATH="\$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-clientcore-2.9.0.jar:\$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-common2.9.0.jar:\$HADOOP_HOME/share/hadoop/common/hadoop-common2.9.0.jar:~/analyzelogs/SalesCountry/*:\$HADOOP_HOME/lib/*"
(above command is available in file: hadoop-mapreduce-example-file.txt)
11. sudo gedit Manifest.txt
12. Add following line and press enter and save the file

Main-Class: SalesCountry.SalesCountryDriver

13. `javac -d . SalesMapper.java SalesCountryReducer.java SalesCountryDriver.java`

14. `jar -cfm analyzelogs.jar Manifest.txt SalesCountry/*.class`

15. `start-dfs.sh`

16. `Start-yarn.sh`

17. `jps`

18. `cd analyzelogs/`

19. `sudo mkdir ~/input`

20. `sudo cp access_log_short.csv ~/input/`

Conclusion:

FAQ's:

1. What does 'jps' command do?
2. What is "MapReduce"? What is the syntax to run a "MapReduce" program?
3. What are the main configuration parameters in a "MapReduce" program?

Assignment No 3

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

- 1) Creating, Dropping, and altering Database tables
- 2) Creating an external Hive table
- 3) Load table with data, insert new values and field in the table, Join tables with Hive
- 4) Create index on Flight information Table
- 5) Find the average departure delay per day in 2008.

CO Achieved: CO6

Theory:

What is 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.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is not

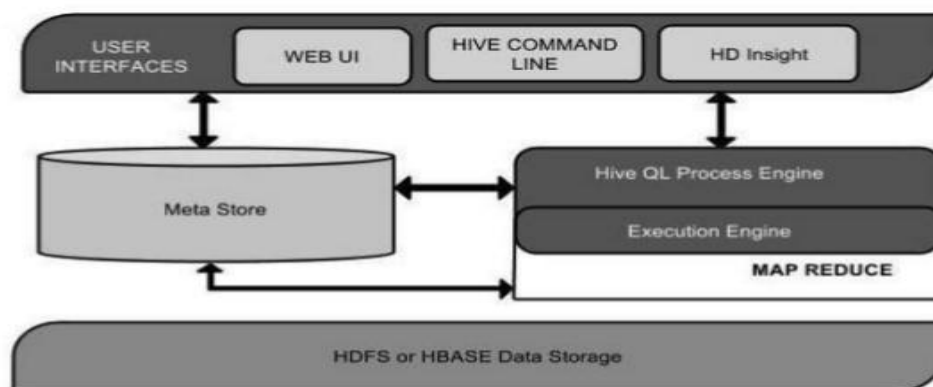
- A relational database
- A design for OnLine Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Features of Hive

- 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 of Hive:

The following component diagram depicts the architecture of Hive:



What is Hive Query Language (HiveQL)?

Hive Query Language (HiveQL) is a query language in Apache Hive for processing and analyzing structured data. It separates users from the complexity of Map Reduce programming. It reuses common concepts from relational databases, such as tables, rows, columns, and schema, to ease learning. Hive provides a CLI for Hive query writing using Hive Query Language (HiveQL).

HiveQL Built-in Operators

Hive provides Built-in operators for Data operations to be implemented on the tables present inside Hive warehouse.

These operators are used for mathematical operations on operands, and it will return specific value as per the logic applied.

Below are the main types of Built-in Operators in HiveQL:

- Relational Operators
- Arithmetic Operators
- Logical Operators
- Operators on Complex types
- Complex type Constructors

Create Database Statement:

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.

The following query is used to verify a databases list:

```
hive> SHOW DATABASES;
```

Create Table Statement

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]
```

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;
```

Load Data Statement

Generally, after creating a table in SQL, we can insert data using the Insert statement. But in Hive, we can insert data using the LOAD DATA statement.

While inserting data into Hive, it is better to use LOAD DATA to store bulk records. There are two ways to load data: one is from local file system and second is from Hadoop file system.

Syntax

The syntax for load data is as follows:

```
LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)]
```

- LOCAL is identifier to specify the local path. It is optional.
- OVERWRITE is optional to overwrite the data in the table.
- PARTITION is optional.

Example

We will insert the following data into the table. It is a text file named **sample.txt** in **/home/user** directory.

1201	Gopal	45000	Technical manager
1202	Manisha	45000	Proof reader
1203	Masthanvali	40000	Technical writer
1204	Kiran	40000	Hr Admin
1205	Kranthi	30000	Op Admin

The following query loads the given text into the table.

```
hive> LOAD DATA LOCAL INPATH '/home/user/sample.txt'
OVERWRITE INTO TABLE employee;
```

Select Statement:

SELECT statement is used to retrieve the data from a table. WHERE clause works similar to a condition. It filters the data using the condition and gives you a finite result. The built-in operators and functions generate an expression, which fulfils the condition.

Syntax

Given below is the syntax of the SELECT query:

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

ORDER BY clause in a SELECT statement:

Syntax

Given below is the syntax of the ORDER BY clause:

```
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];
```

Conclusion: We have implemented flight information table and performed basic operations using HIVE QL.

FAQ's:

1. What are the different types of tables available in Hive?
2. Can a table be renamed in Hive?
3. Why do we need Hive?
4. What is a Hive variable? What for we use it?

Assignment No 4

Title: 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

CO Achieved: CO6

Theory:

Loading Data in Python:

pandas is a powerful data analysis package. It makes data exploration and manipulation easy. It has several functions to read data from various sources.

If you are using Anaconda, pandas must be already installed. You need to load the package by using the following command -

import pandas as pd

If pandas package is not installed, you can install it by running the following code in Ipython Console. If you are using Spyder, you can submit the following code in Ipython console within Spyder.

!pip install pandas

If you are using Anaconda, you can try the following line of code to install pandas -

!conda install pandas

Import CSV files:

import pandas as pd

mydata= pd.read_csv("path/to/file1.csv")

A to T: M-27

If no header (title) in raw data file

mydata1 = pd.read_csv("path/to/file1.csv", header = None)

You need to include header = None option to tell Python there is no column name (header) in data.

Add Column Names

We can include column names by using names= option.

mydata2 = pd.read_csv("path/to/file1.csv", header = None, names = ['ID', 'first_name', 'salary'])

The variable names can also be added separately by using the following command.

```
mydata1.columns = ['ID', 'first_name', 'salary']
```

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.

1. Selecting Only Columns

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

```
mydata['salary']
```

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

You can also select multiple columns using indexing operator.

```
mydata[['salary', 'ID']]
```

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

```
Subset1 = mydata[['salary', 'ID']]
```

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 salary greater than 50000 you can use the following line of code.

```
salary_50000 = mydata[mydata['salary']>50000]
```

Subset a Dataframe using Python .loc():

.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.

1. Selecting Rows with loc()

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

```
mydata.loc[1]
```

To select multiple rows use :

```
mydata.loc[[1,3,4]]
```

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

```
mydata.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 :

```
mydata.loc[1:7,['salary', 'ID']]
```

This line of code selects rows from 1 to 7 and columns corresponding to the labels 'salary' and 'ID'.

Merge Data in Python:

Pandas provides various facilities for easily combining together Series or DataFrame with various kinds of set logic for the indexes and relational algebra functionality in the case of join / merge-type operations.

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.

```
pd.concat(
    objs,
    axis=0,
    join="outer",
    ignore_index=False,
    keys=None,
    levels=None,
    names=None,
    verify_integrity=False,
    copy=True,
)
```

`objs` : a sequence or mapping of Series or DataFrame objects.

`axis` : {0, 1, ...}, default 0. The axis to concatenate along.

`join` : {'inner', 'outer'}, default 'outer'.

`keys` : sequence, default None.

`levels` : list of sequences, default None.


```

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)

```

```

In [6]: result = pd.concat(frames, keys=["x", "y", "z"])

```

df1					Result					
	A	B	C	D			A	B	C	D
0	A0	B0	C0	D0	x	0	A0	B0	C0	D0
1	A1	B1	C1	D1		1	A1	B1	C1	D1
2	A2	B2	C2	D2		2	A2	B2	C2	D2
3	A3	B3	C3	D3		3	A3	B3	C3	D3
df2					y	4	A4	B4	C4	D4
4	A4	B4	C4	D4		5	A5	B5	C5	D5
5	A5	B5	C5	D5		6	A6	B6	C6	D6
6	A6	B6	C6	D6		7	A7	B7	C7	D7
7	A7	B7	C7	D7	z	8	A8	B8	C8	D8
df3						9	A9	B9	C9	D9
8	A8	B8	C8	D8		10	A10	B10	C10	D10
9	A9	B9	C9	D9		11	A11	B11	C11	D11
10	A10	B10	C10	D10						
11	A11	B11	C11	D11						

As you can see (if you've read the rest of the documentation), the resulting object's

index has a hierarchical index. This means that we can now select out each chunk by key:

```
In [7]: result.loc["y"]
Out[7]:
```

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

Appending rows to a DataFrame:

If you have a series that you want to append as a single row to a DataFrame, you can convert the row into a DataFrame and use concat

```
In [31]: s2 = pd.Series(["X0", "X1", "X2", "X3"], index=["A", "B", "C", "D"])
In [32]: result = pd.concat([df1, s2.to_frame().T], ignore_index=True)
```

df1					Result				
	A	B	C	D		A	B	C	D
0	A0	B0	C0	D0	0	A0	B0	C0	D0
1	A1	B1	C1	D1	1	A1	B1	C1	D1
2	A2	B2	C2	D2	2	A2	B2	C2	D2
3	A3	B3	C3	D3	3	A3	B3	C3	D3
s2					4	X0	X1	X2	X3
	A			X0					
	B			X1					
	C			X2					
	D			X3					

Database-style DataFrame or named Series joining/merging:

Pandas has full-featured, high performance in-memory join operations idiomatically very similar to relational databases like SQL. These methods perform significantly better (in some cases well over an order of magnitude better) than other open source implementations (like `base::merge.data.frame` in R). The reason for this is careful algorithmic design and the internal layout of the data in DataFrame.

pandas provides a single function, `merge()`, as the entry point for all standard database join operations between DataFrame or named Series objects:

```

pd.merge(
    left,
    right,
    how="inner",
    on=None,
    left_on=None,
    right_on=None,
    left_index=False,
    right_index=False,
    sort=True,
    suffixes=("_x", "_y"),
    copy=True,
    indicator=False,
    validate=None,
)

```

- left: A DataFrame or named Series object.
- right: Another DataFrame or named Series object.
- on: Column or index level names to join on. Must be found in both the left and right DataFrame and/or Series objects.
- left_on: Columns or index levels from the left DataFrame or Series to use as keys.
- right_on: Columns or index levels from the right DataFrame or Series to use as keys.
- left_index: If True, use the index (row labels) from the left DataFrame or Series as its join key(s).
- right_index: Same usage as left_index for the right DataFrame or Series
- how: One of 'left', 'right', 'outer', 'inner', 'cross'. Defaults to inner.
- sort: Sort the result DataFrame by the join keys in lexicographical order
- suffixes: A tuple of string suffixes to apply to overlapping columns. Defaults to ('_x', '_y').

The return type will be the same as left. If left is a DataFrame or named Series and right is a subclass of DataFrame, the return type will still be DataFrame.

merge is a function in the pandas namespace, and it is also available as a DataFrame instance method merge(), with the calling DataFrame being implicitly considered the left object in the join.

There are several cases to consider which are very important to understand:

- one-to-one joins: for example when joining two DataFrame objects on their indexes (which must contain unique values).
- many-to-one joins: for example when joining an index (unique) to one or more columns in a different DataFrame.
- many-to-many joins: joining columns on columns.

It is worth spending some time understanding the result of the many-to-many join case. In SQL / standard relational algebra, if a key combination appears more than once in both tables, the resulting table will have the Cartesian product of the associated data. Here is a very basic example with one unique key combination:

```
In [33]: left = pd.DataFrame(
.....:     {
.....:         "key": ["K0", "K1", "K2", "K3"],
.....:         "A": ["A0", "A1", "A2", "A3"],
.....:         "B": ["B0", "B1", "B2", "B3"],
.....:     }
.....: )

In [34]: right = pd.DataFrame(
.....:     {
.....:         "key": ["K0", "K1", "K2", "K3"],
.....:         "C": ["C0", "C1", "C2", "C3"],
.....:         "D": ["D0", "D1", "D2", "D3"],
.....:     }
.....: )

In [35]: result = pd.merge(left, right, on="key")
```

left				right				Result					
	key	A	B		key	C	D		key	A	B	C	D
0	K0	A0	B0	0	K0	C0	D0	0	K0	A0	B0	C0	D0
1	K1	A1	B1	1	K1	C1	D1	1	K1	A1	B1	C1	D1
2	K2	A2	B2	2	K2	C2	D2	2	K2	A2	B2	C2	D2
3	K3	A3	B3	3	K3	C3	D3	3	K3	A3	B3	C3	D3

Here is a more complicated example with multiple join keys. Only the keys appearing in left and right are present (the intersection), since how='inner' by default.

```
In [36]: left = pd.DataFrame(
.....:     {
.....:         "key1": ["K0", "K0", "K1", "K2"],
.....:         "key2": ["K0", "K1", "K0", "K1"],
.....:         "A": ["A0", "A1", "A2", "A3"],
.....:         "B": ["B0", "B1", "B2", "B3"],
.....:     }
.....: )

In [37]: right = pd.DataFrame(
.....:     {
.....:         "key1": ["K0", "K1", "K1", "K2"],
.....:         "key2": ["K0", "K0", "K0", "K0"],
.....:         "C": ["C0", "C1", "C2", "C3"],
.....:         "D": ["D0", "D1", "D2", "D3"],
.....:     }
.....: )

In [38]: result = pd.merge(left, right, on=["key1", "key2"])
```

left					right					Result						
	key1	key2	A	B		key1	key2	C	D		key1	key2	A	B	C	D
0	K0	K0	A0	B0	0	K0	K0	C0	D0	0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	1	K1	K0	C1	D1	1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	2	K1	K0	C2	D2	2	K1	K0	A2	B2	C2	D2
3	K2	K1	A3	B3	3	K2	K0	C3	D3							

The how argument to merge specifies how to determine which keys are to be included in the resulting table. If a key combination does not appear in either the left or right tables, the values in the joined table will be NA.

Merge method	SQL Join Name	Description
left	LEFT OUTER JOIN	Use keys from left frame only
right	RIGHT OUTER JOIN	Use keys from right frame only
outer	FULL OUTER JOIN	Use union of keys from both frames
inner	INNER JOIN	Use intersection of keys from both frames
cross	CROSS JOIN	Create the cartesian product of rows of both frames

Sort Data:

- 1. 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

2. 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. An index isn't considered a column, and you typically have only a single row index. The row index can be thought of as the row numbers, which start from zero.

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.

```
mydata.sort_values("salary")
```

This sorts your DataFrame using the column values from salary, showing the employees with the lowest salary first. By default, `.sort_values()` sorts your data in ascending order. Although you didn't specify a name for the argument you passed to `.sort_values()`, you actually used the `by` parameter.

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.

By passing `False` to `ascending`, you reverse the sort order. Now your DataFrame is sorted in descending order by the salary.

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:

```
mydata.sort_values(  
...     by="salary",  
...     ascending=False,  
...     kind="mergesort"  
... )
```

Using `kind`, you set the sorting algorithm to mergesort. The previous output used the default quicksort algorithm. Looking at the highlighted indices, you can see the rows are in a different order. This is because quicksort is not a stable sorting algorithm, but mergesort is.

When you're sorting multiple records that have the same key, a stable sorting algorithm 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

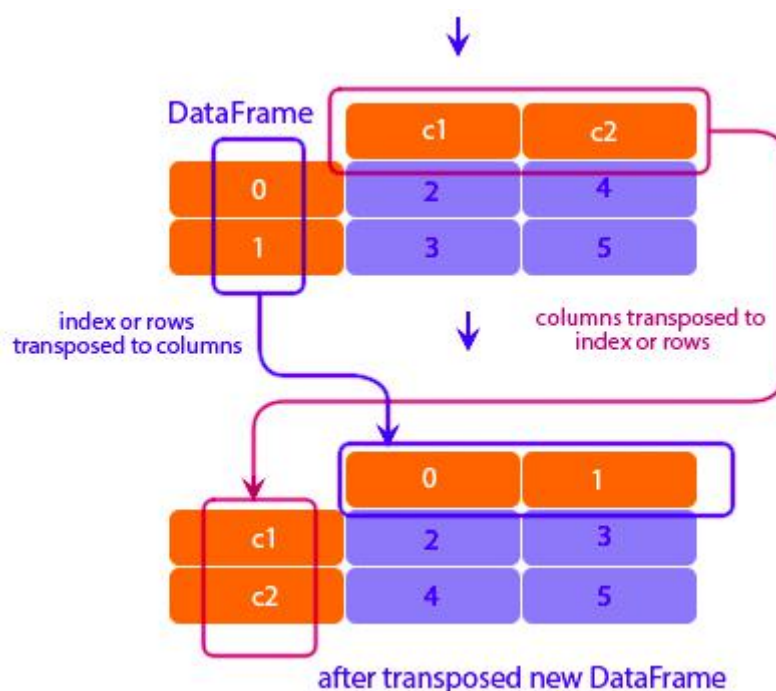
In data analysis, it's common to want to sort your data based on the values of multiple columns. Imagine you have a dataset with people's first and last names. It would make sense to sort by last name and then first name, so that people with the same last name are arranged alphabetically according to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

```
mydata.sort_values(by=["city08", "highway08"])
```

Transpose Data:

The `transpose()` function is used to transpose index and columns. Reflect the DataFrame over its main diagonal by writing rows as columns and vice-versa.

```
mydata_transposed = mydata.T      # or mydata.transpose()
```



Shape and reshape Data:

Shape

The `shape` attribute shows the number of items in each dimension. Checking a DataFrame's shape returns a tuple with two integers. The first is the number of rows and the second is the number of columns.

```
mydata.shape
```

```
(3, 2)
```

We have three rows and two columns. The `size` attribute shows us how many cells we have.

mydata.size

6

Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

```
df.pivot(index='foo',  
          columns='bar',  
          values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

Conclusion: Completed the basic operations of data sub-setting, Merging, Sorting, Transposing and Shape and reshape on the Facebook metrics dataset.

FAQ's:

1. Explain difference between .loc and .iloc functions of pandas.
2. Explain the concept of Boolean indexing with example
3. Compare and contrast inner and outer joins of merge function with SQL operations.

Assignment No. 5

Title: 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

Theory :

Data Cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

Empty Cells

Empty cells can potentially give you a wrong result when you analyze data.

Remove Rows

One way to deal with empty cells is to remove rows that contain empty cells.

This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

new_df = df.dropna()

By default, the dropna() method returns a new DataFrame, and will not change the original.

If you want to change the original DataFrame, use the inplace = True argument:

df.dropna(inplace = True)

Replace Empty Values

Another way of dealing with empty cells is to insert a new value instead.

This way you do not have to delete entire rows just because of some empty cells.

The fillna() method allows us to replace empty cells with a value:

Replace NULL values with the number 130: ***df.fillna(130, inplace = True)***

Replace Only For Specified Columns

The example above replaces all empty cells in the whole Data Frame.

To only replace empty values for one column, specify the column name for the DataFrame

Replace NULL values in the "Calories" columns with the number 130:
df["Calories"].fillna(130, inplace = True)

Replace Using Mean, Median, or Mode

A common way to replace empty cells, is to calculate the mean, median or mode value of the column.

Pandas uses the mean() median() and mode() methods to calculate the respective values for a specified column:

Calculate the MEAN, and replace any empty values with it:

```
x = df["Calories"].mean()  
df["Calories"].fillna(x, inplace = True)
```

Calculate the MEDIAN, and replace any empty values with it:

```
x = df["Calories"].median()  
df["Calories"].fillna(x, inplace = True)
```

Calculate the MODE, and replace any empty values with it:

```
x = df["Calories"].mode()[0]  
df["Calories"].fillna(x, inplace = True)
```

Data of Wrong Format

Cells with data of wrong format can make it difficult, or even impossible, to analyze data.

To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format.

Let's try to convert all cells in the 'Date' column into dates.

Pandas has a to_datetime() method for this: *df['Date'] = pd.to_datetime(df['Date'])*

Discovering Duplicates

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

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

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

```
print(df.duplicated())
```

Removing Duplicates

To remove duplicates, use the drop_duplicates() method.

```
df.drop_duplicates(inplace = True)
```

Integrate Data from a Database:

Beside CSV and other similar files you will often connect data to a database here we need beside Pandas also the sqlite3 module

```
import pandas as pd  
import sqlite3
```

After the successful import of the data, we now can easily query tables via SQL String:

```
# Read via SQLite databases  
con = sqlite3.connect("your.database.link")  
#Read table via Select Statement  
player = pd.read_sql_query("SELECT * from Table", con)  
#close the connection  
con.close()
```

Data transformation

Visualization is an important tool for insight generation, but it is rare that you get the data in exactly the right form you need. You will often need to create some new variables or summaries, rename variables, or reorder observations for the data to be easier to manage

Below are the five methods that allow you to solve the vast majority of your data manipulation challenges:

- Pick observations by their values (query()).
- Reorder the rows (sort_values()).
- Pick variables by their names (filter()).
- Create new variables with functions of existing variables (assign()).
- Collapse many values down to a single summary (groupby()).

1. The groupby() changes the scope of each function from operating on the entire dataset to operating on it group-by-group. These functions provide the verbs for a language of data manipulation.

All verbs work similarly:

- The first argument is a pandas DataFrame.
- The subsequent methods describe what to do with the data frame.
- The result is a new data frame.

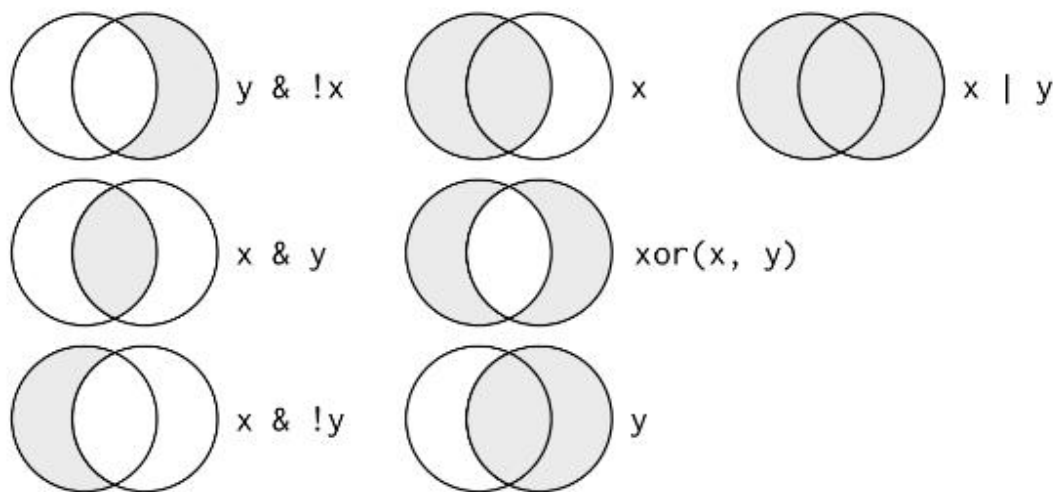
Together these properties make it easy to chain together multiple simple steps to achieve a complex result. Let's dive in and see how these verbs work.

2. `.query()` allows you to subset observations based on their values. The first argument specifies the rows to be selected. This argument can be label names or a boolean series. The second argument specifies the columns to be selected. The boolean filter on the rows is our focus. For example, we can select all flights on January 1st with:

`flights.query('month == 1 & day == 1')`

To use filtering effectively, you have to know how to select the observations that you want using the comparison operators. Python provides the standard suite: `>`, `>=`, `<`, `<=`, `!=` (not equal), and `==` (equal).

Multiple arguments to `query()` are combined with “and”: every expression must be true in order for a row to be included in the output. For other types of combinations, you’ll need to use Boolean operators yourself: `&` is “and”, `|` is “or”, and `!` is “not”. Figure shows the complete set of Boolean operations.



It’s not uncommon to get datasets with hundreds or even thousands of variables. In this case, the first challenge is often narrowing in on the variables you’re actually interested in. `.filter()` allows you to rapidly zoom in on a useful subset using operations based on the names of the variables.

Additionally, `.loc[]` is often used to select columns by many user of pandas.

There are a number of helper regular expressions you can use within `filter()`:

`flights.filter(regex = '^sch')`: matches column names that begin with “sch”.

`flights.filter(regex = "time$")`: matches names that end with “time”.

`flights.filter(regex = "_dep_")`: matches names that contain “dep”.

`flights.filter(regex = '(.\\|I)')`: selects variables that match a regular expression. This one matches any variables that contain repeated characters

Use `rename()` to rename a column or multiple columns.

```
flights.rename(columns = {'year': 'YEAR', 'month': 'MONTH'})
```

Besides selecting sets of existing columns, it's often useful to add new columns that are functions of existing columns. That's the job of `.assign()`.

`.assign()` always adds new columns at the end of your dataset

```
(flights_sml.assign(  
    gain = lambda x: x.dep_delay - x.arr_delay,  
    speed = lambda x: x.distance / x.air_time * 60  
).head())
```

The last key verb is `.agg()`. It collapses a data frame to a single row:

```
flights.agg({'dep_delay': np.mean})
```

Pandas aggregate functions ignores the `np.nan` values like `na.rm = TRUE` in R.

`.agg()` is not terribly useful unless we pair it with `.groupby()`. This changes the unit of analysis from the complete dataset to individual groups. Then, when you use the pandas functions on a grouped data frame they'll be automatically applied "by group".

```
by_day = flights.groupby(['year', 'month', 'day'])  
by_day.agg(delay = ('dep_delay', np.mean)).reset_index()
```

Together `.groupby()` and `.agg()` provide one of the tools that you'll use most commonly when working with pandas: grouped summaries

3. Grouping is most useful in conjunction with `.agg()`, but you can also do convenient operations with `.transform()`. This is a difference in pandas as compared to dplyr. Once you create a `.groupby()` object you cannot use `.assign()` and the best equivalent is `.transform()`.

Pandas `DataFrame.transform()` function call `func` on `self` producing a `DataFrame` with transformed values and that has the same axis length as `self`.

Syntax: `DataFrame.transform(func, axis=0, *args, **kwargs)`

Parameter :

func : Function to use for transforming the data

axis : {0 or 'index', 1 or 'columns'}, default 0

***args :** Positional arguments to pass to `func`.

****kwargs :** Keyword arguments to pass to `func`.

Returns : DataFrame

Example:

```
# add 10 to each element of the dataframe  
result = df.transform(func = lambda x : x + 10)
```

Data model building:

1. visualize your data and check if there are correlations between the different characteristics that we obtained. It will be necessary to make a selection of characteristics since the ones you choose will directly impact the execution times and the results. You can also reduce dimensions by applying PCA if necessary.

2. You must also separate the data into two groups: one for training and the other for model evaluation which can be divided approximately in a ratio of 80/20 but it can vary depending on the case and the volume of data we have.

At this stage, you can also pre-process your data by normalizing, eliminating duplicates, and making error corrections.

3. There are several models that you can choose according to the objective that you might have: you will use algorithms of classification, prediction, linear regression, clustering, i.e. k-means or K-Nearest Neighbor, Deep Learning, i.e Neural Networks, Bayesian, etc.

There are various models to be used depending on the data you are going to process such as images, sound, text, and numerical values.

Model	Applications
Logistic Regression	Price prediction
Fully connected networks	Classification
Convolutional Neural Networks	Image processing
Recurrent Neural Networks	Voice recognition
Random Forest	Fraud Detection
Reinforcement Learning	Learning by trial and error
Generative Models	Image creation
K-means	Segmentation
k-Nearest Neighbors	Recommendation systems
Bayesian Classifiers	Spam and noise filtering

4. You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate. Remember to initialize the weights of your model randomly -the weights are the values that multiply or affect the relationships between the inputs and outputs- which will be automatically adjusted by the selected algorithm the more you train them.

5. You will have to check the machine created against your evaluation data set that contains inputs that the model does not know and verify the precision of your already trained model. If the accuracy is less than or equal to 50%, that model will not be useful since it would be like tossing a coin to make decisions. If you reach 90% or more, you can have good confidence in the results that the model gives you.

#From this we observe that the minimum correlation between output and other features in

#fbs,trtbps and chol

x = data.drop("output", axis=1)

y = data["output"]

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)

*****Logistic Regression*****

logReg = LogisticRegression(random_state=0, solver='liblinear')

logReg.fit(x_train, y_train)

#Check accuracy of Logistic Regression

y_pred_logReg = logReg.predict(x_test)

#Model Accuracy

*print("Accuracy of logistic regression classifier ::
",metrics.accuracy_score(y_test,y_pred_logReg))*

*****K Neighbours Classifier*****

knc = KNeighborsClassifier()

knc.fit(x_train,y_train)

y_pred_knc = knc.predict(x_test)

*print("Accuracy of K-Neighbours classifier :: ",
metrics.accuracy_score(y_test,y_pred_knc))*

Conclusion: Implemented the operations of data cleaning, integration, transformation and model building on the Air Quality and Heart Disease datasets.

FAQ's:

1. List the best practices for cleaning data
2. How are null values stored in pandas data frames?
3. How to choose the right machine learning algorithm for your dataset?

Assignment No. 7

Title: Visualize the data using Python libraries matplotlib, seaborn by plotting the graphs for assignment no. 4 and 5 above

Theory:

It may sometimes seem easier to go through a set of data points and build insights from it but usually this process may not yield good results. There could be a lot of things left undiscovered as a result of this process. Additionally, most of the data sets used in real life are too big to do any analysis manually. This is essentially where data visualization steps in.

Data visualization is an easier way of presenting the data, however complex it is, to analyze trends and relationships amongst variables with the help of pictorial representation.

The following are the advantages of Data Visualization

- Easier representation of complex data
- Highlights good and bad performing areas
- Explores relationship between data points
- Identifies data patterns even for larger data points

While building visualization, it is always a good practice to keep some below mentioned points in mind

- Ensure appropriate usage of shapes, colors, and size while building visualization
- Plots/graphs using a co-ordinate system are more pronounced
- Knowledge of suitable plot with respect to the data types brings more clarity to the information
- Usage of labels, titles, legends and pointers passes seamless information to the wider audience

Python Libraries

There are a lot of python libraries which could be used to build visualization like matplotlib, vispy, bokeh, seaborn, pygal, folium, plotly, cufflinks, and networkx. Of the many, matplotlib and seaborn seems to be very widely used for basic to intermediate level of visualizations.

Matplotlib

It is an amazing visualization library in Python for 2D plots of arrays. It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. Let's try to understand some of the benefits and features of matplotlib

- It's fast, efficient as it is based on numpy and also easier to build
- Has undergone a lot of improvements from the open source community since

inception and hence a better library having advanced features as well

- Well maintained visualization output with high quality graphics draws a lot of users to it
- Basic as well as advanced charts could be very easily built
- From the users/developers point of view, since it has a large community support, resolving issues and debugging becomes much easier

Seaborn

Conceptualized and built originally at the Stanford University, this library sits on top of matplotlib. In a sense, it has some flavors of matplotlib while from the visualization point, it is much better than matplotlib and has added features as well. Below are its advantages

- Built-in themes aid better visualization
- Statistical functions aiding better data insights
- Better aesthetics and built-in plots
- Helpful documentation with effective examples

Nature of Visualization

Depending on the number of variables used for plotting the visualization and the type of variables, there could be different types of charts which we could use to understand the relationship. Based on the count of variables, we could have

- Univariate plot(involves only one variable)
- Bivariate plot(more than one variable is required)

A Univariate plot could be for a continuous variable to understand the spread and distribution of the variable while for a discrete variable it could tell us the count.

Similarly, a Bivariate plot for continuous variable could display essential statistic like correlation, for a continuous versus discrete variable could lead us to very important conclusions like understanding data distribution across different levels of a categorical variable. A bivariate plot between two discrete variables could also be developed.

Box plot

A boxplot, also known as a box and whisker plot, the box and the whisker are clearly displayed in the below image. It is a very good visual representation when it comes to measuring the data distribution. Clearly plots the median values, outliers and the quartiles. Understanding data distribution is another important factor which leads to better model building. If data has outliers, box plot is a recommended way to identify them and take necessary actions.

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.

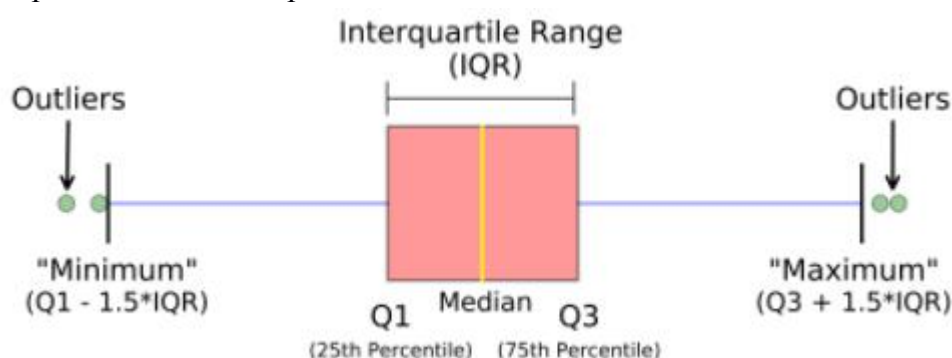
Returns: It returns the Axes object with the plot drawn onto it.

The box and whiskers chart shows how data is spread out. Five pieces of information are generally included in the chart

- The minimum is shown at the far left of the chart, at the end of the left 'whisker'
- First quartile, Q1, is the far left of the box (left whisker)
- The median is shown as a line in the center of the box
- Third quartile, Q3, shown at the far right of the box (right whisker)
- The maximum is at the far right of the box

As could be seen in the below representations and charts, a box plot could be plotted for one or more than one variable providing very good insights to our data.

Representation of box plot:



Scatter Plot

Scatter plots or scatter graphs is a bivariate plot having greater resemblance to line graphs in the way they are built. A line graph uses a line on an X-Y axis to plot a continuous function, while a scatter plot relies on dots to represent individual pieces of data. These plots are very useful to see if two variables are correlated. Scatter plot could be 2 dimensional or 3 dimensional.

Syntax: `seaborn.scatterplot(x=None, y=None, hue=None, style=None, size=None, data=None, palette=None, hue_order=None, hue_norm=None, sizes=None, size_order=None, size_norm=None, markers=True, style_order=None, x_bins=None, y_bins=None, units=None, estimator=None, ci=95, n_boot=1000, alpha='auto', x_jitter=None, y_jitter=None, legend='brief', ax=None, **kwargs)`

Parameters:

x, y: Input data variables that should be numeric.

data: Dataframe where each column is a variable and each row is an observation.

size: Grouping variable that will produce points with different sizes.

style: Grouping variable that will produce points with different markers.

palette: Grouping variable that will produce points with different markers.

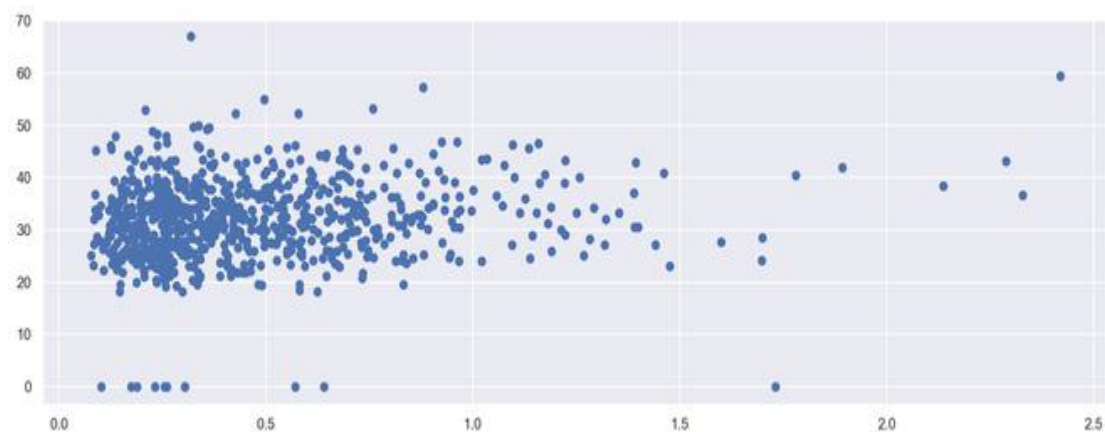
markers: Object determining how to draw the markers for different levels.

alpha: Proportional opacity of the points.

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

Advantages of a scatter plot

- Displays correlation between variables
- Suitable for large data sets
- Easier to find data clusters
- Better representation of each data point

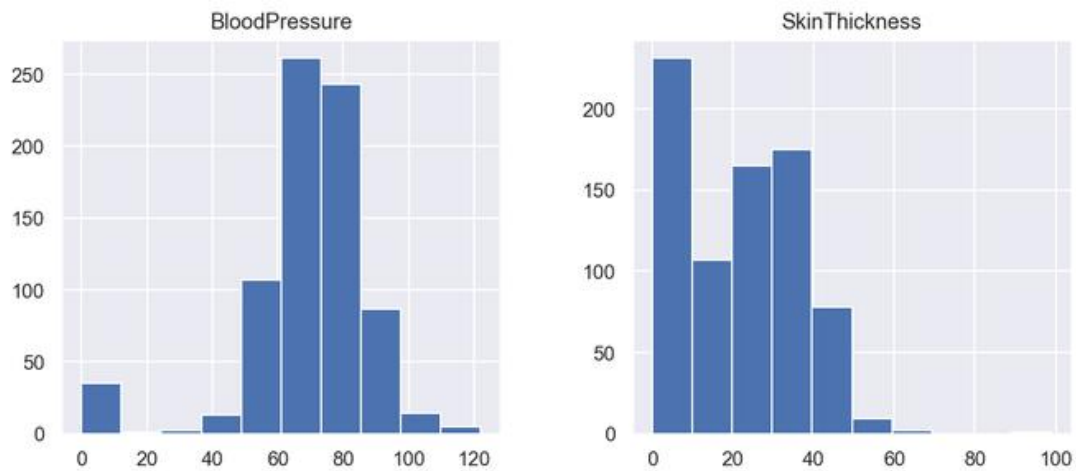


Histogram

Histograms display counts of data and are hence similar to a bar chart. A histogram plot can also tell us how close a data distribution is to a normal curve. While working out statistical method, it is very important that we have a data which is normally or close to a normal distribution. However, histograms are univariate in nature and bar charts bivariate.

A bar graph charts actual counts against categories e.g. height of the bar indicates the number of items in that category whereas a histogram displays the same categorical variables in bins.

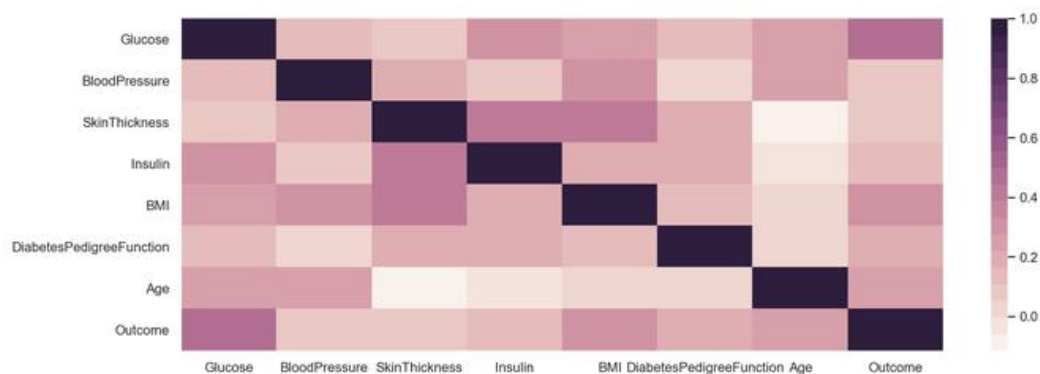
Bins are integral part while building a histogram they control the data points which are within a range. As a widely accepted choice we usually limit bin to a size of 5-20, however this is totally governed by the data points which is present.



Correlation plot

Correlation plot is a multi-variate analysis which comes very handy to have a look at relationship with data points. Scatter plots helps to understand the affect of one variable over the other. Correlation could be defined as the affect which one variable has over the other.

Correlation could be calculated between two variables or it could be one versus many correlations as well which we could see the below plot. Correlation could be positive, negative or neutral and the mathematical range of correlations is from -1 to 1. Understanding the correlation could have a very significant effect on the model building stage and also understanding the model outputs.



Heat Maps

Heat map is a multi-variate data representation. The color intensity in a heat map displays becomes an important factor to understand the affect of data points. Heat maps are easier to understand and easier to explain as well. When it comes to data analysis using visualization, its very important that the desired message gets conveyed with the help of plots.

Syntax:

seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, robust=False, annot=None, fmt='.2g', annot_kws=None, linewidths=0, linecolor='white', cbar=True, cbar_kws=None, cbar_ax=None, square=False, xticklabels='auto', yticklabels='auto', mask=None, 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.

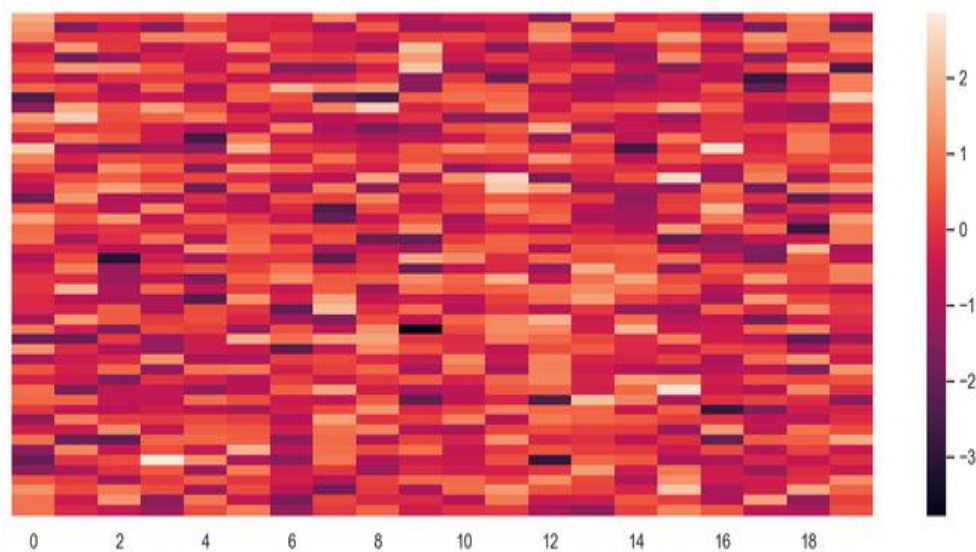
color : (optional) This parameter take matplotlib color, Color for all of the elements, or seed for a gradient palette.

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.

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.



Pie Chart

Pie chart is a univariate analysis and are typically used to show percentage or

proportional data. The percentage distribution of each class in a variable is provided next to the corresponding slice of the pie. The python libraries which could be used to build a pie chart is matplotlib and seaborn.

Syntax: `matplotlib.pyplot.pie(data, explode=None, labels=None, colors=None, autopct=None, shadow=False)`

Parameters:

data represents the array of data values to be plotted, the fractional area of each slice is represented by $\text{data}/\text{sum}(\text{data})$. If $\text{sum}(\text{data}) < 1$, then the data values returns the fractional area directly, thus resulting pie will have empty wedge of size $1 - \text{sum}(\text{data})$.

labels is a list of sequence of strings which sets the label of each wedge.

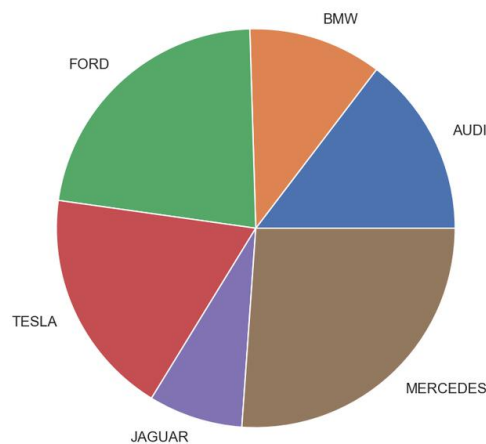
color attribute is used to provide color to the wedges.

autopct is a string used to label the wedge with their numerical value.

shadow is used to create shadow of wedge.

Below are the advantages of a pie chart

- Easier visual summarization of large data points
- Effect and size of different classes can be easily understood
- Percentage points are used to represent the classes in the data points



Conclusion: Implemented various visualization techniques on the airquality and heart disease data using Python libraries matplotlib, seaborn by plotting the graphs.

FAQs:

1. Name a few libraries in Python used for Data Analysis and Scientific computations?
2. Which library would you prefer for plotting in Python language: Seaborn or Matplotlib and WHY?

Assignment No 8

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

- 1) 1D (Linear) Data visualization
- 2) 2D (Planar) Data Visualization
- 3) 3D (Volumetric) Data Visualization
- 4) Temporal Data Visualization
- 5) Multidimensional Data Visualization
- 6) Tree/ Hierarchical Data visualization
- 7) Network Data visualization

CO Achieved: CO6

Theory:

Introduction

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

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

1D/Linear

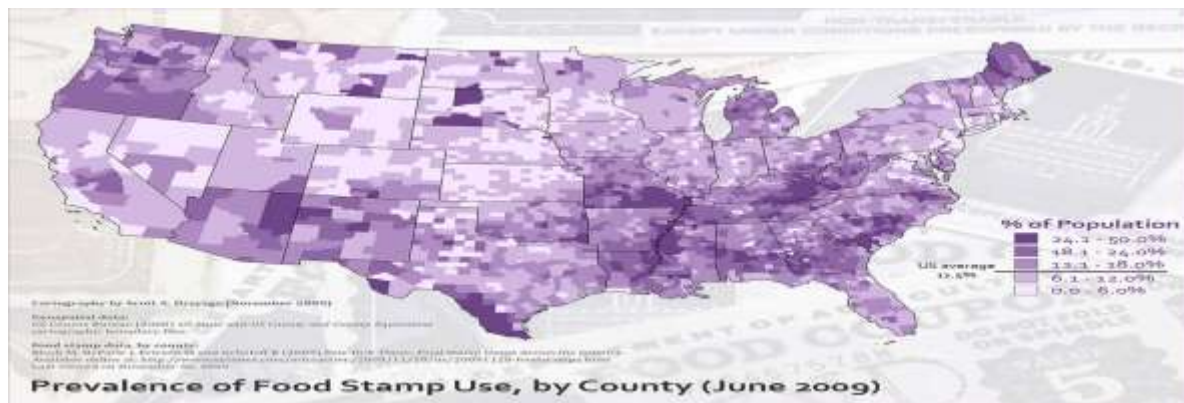
Examples:

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

2D/Planar (especially geospatial)

Examples (geospatial):

- **choropleth**



3D/Volumetric

3D/Volumetric

Broadly, examples of scientific visualization:

- 3D computer models

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

- surface and volume rendering

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

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

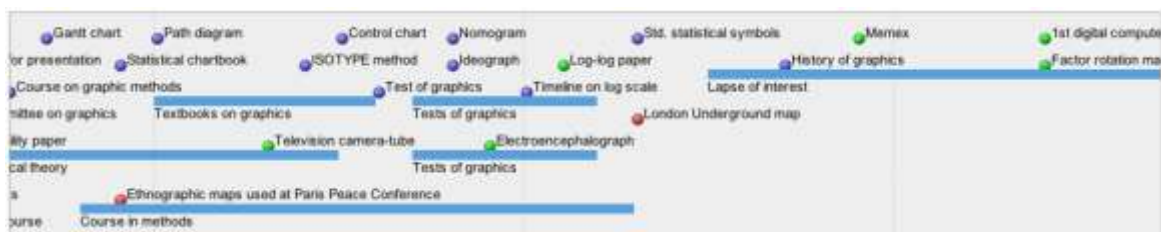
- computer simulations

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

Temporal

Examples:

- timeline

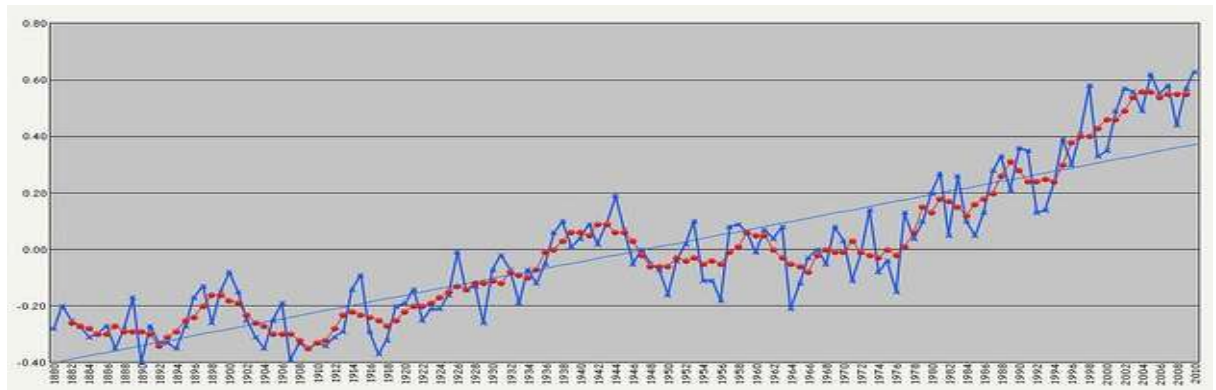


Tools: [SIMILE Timeline](#), [TimeFlow](#), [Timeline JS](#), [Excel](#)

Image:

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

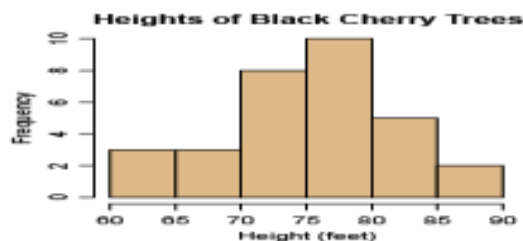
- **time series**



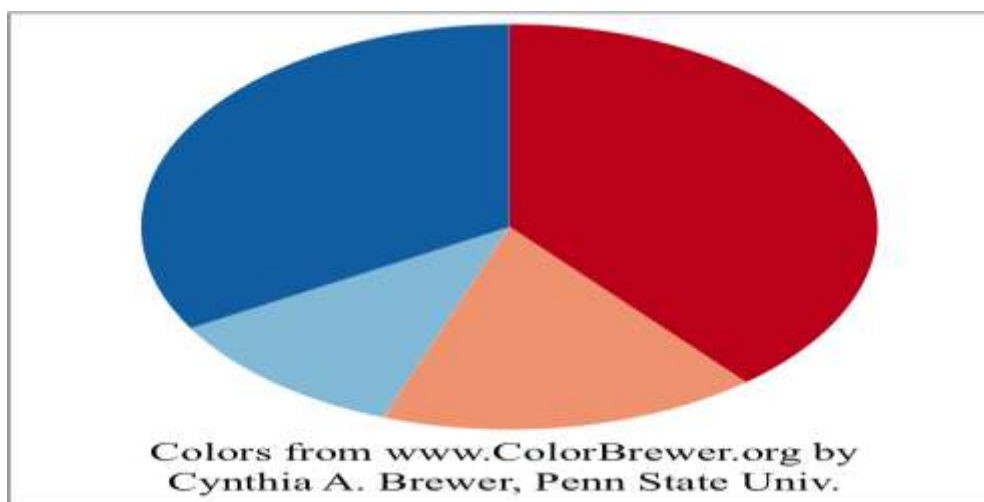
nD/Multidimensional

Examples (category proportions, counts):

- **histogram**



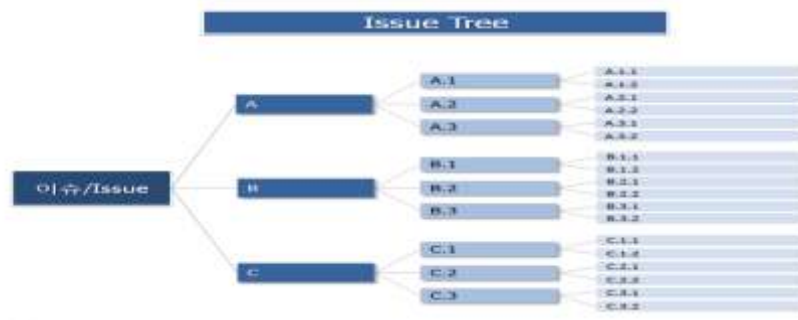
- **pie chart**



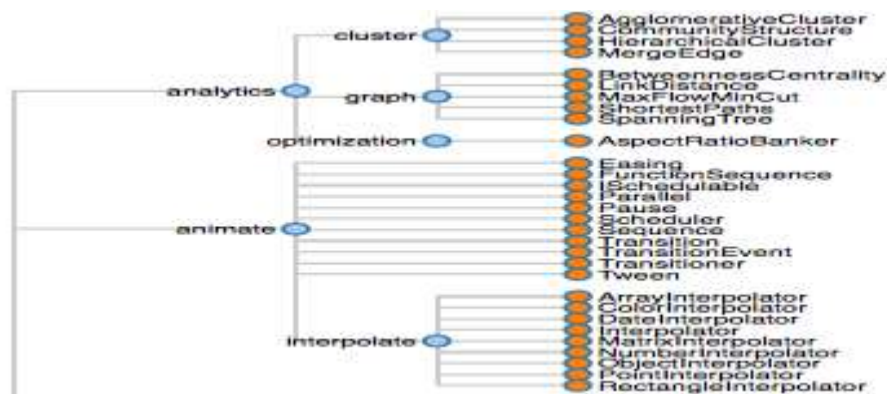
Tree/Hierarchical

Examples:

- general tree visualization



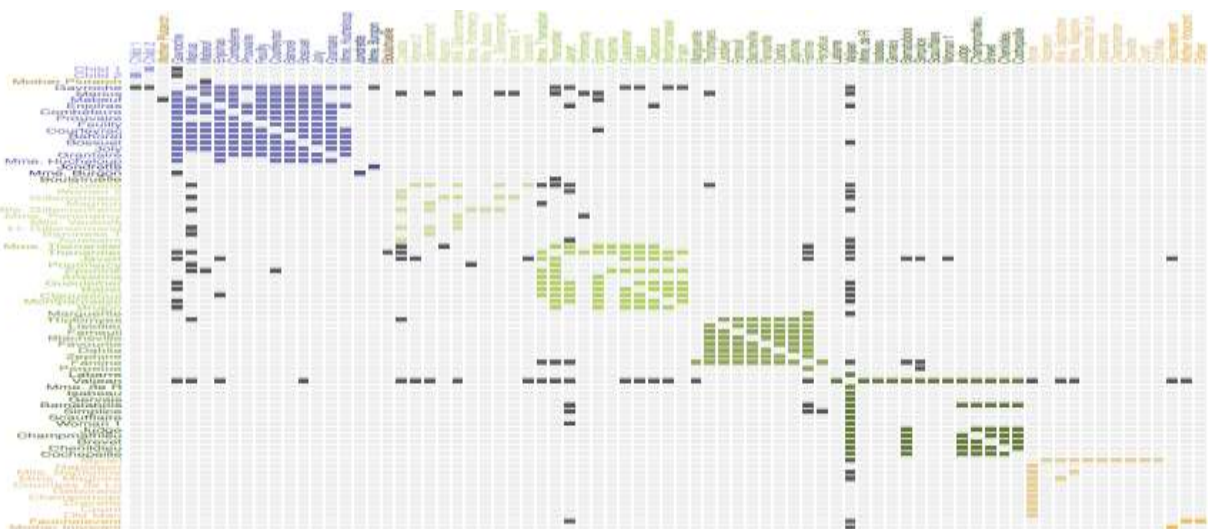
- dendrogram



Network

Examples:

- matrix



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

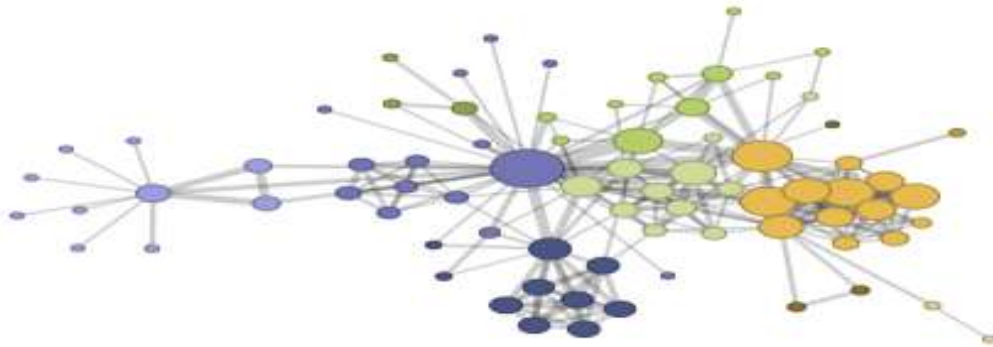


Tableau:

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

Tableau Features:

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

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

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

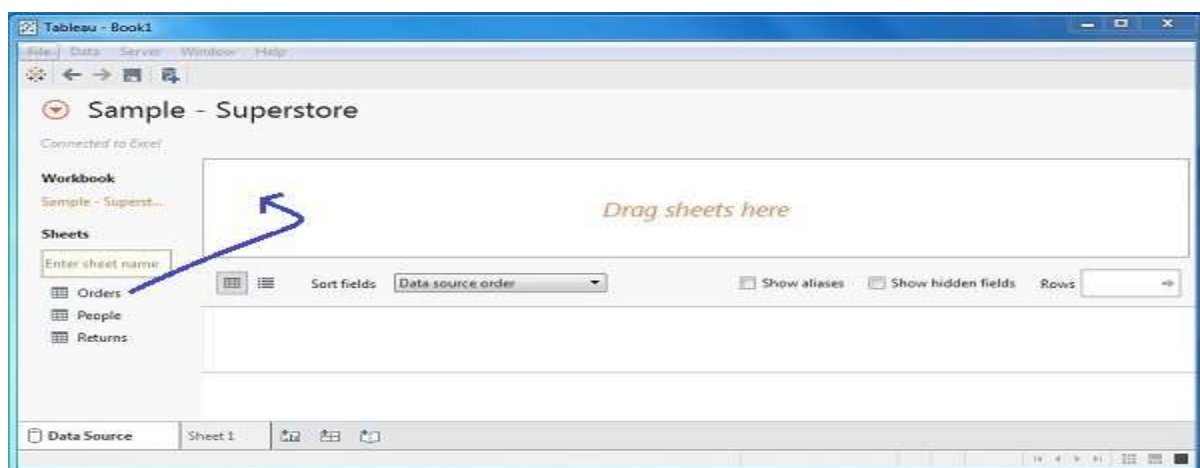
These three steps are –

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

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

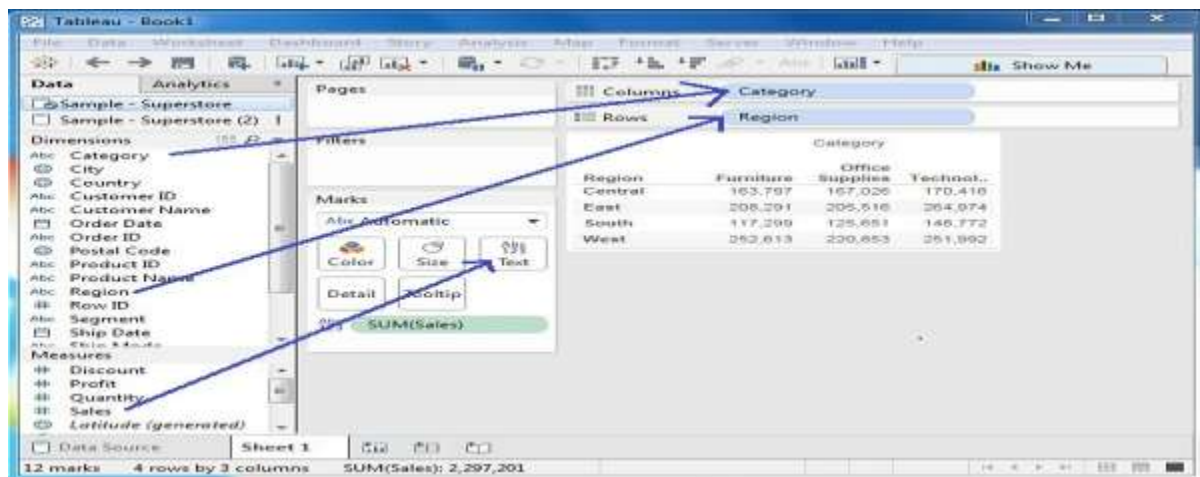
Connect to a Data Source

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



Choose the Dimensions and Measures

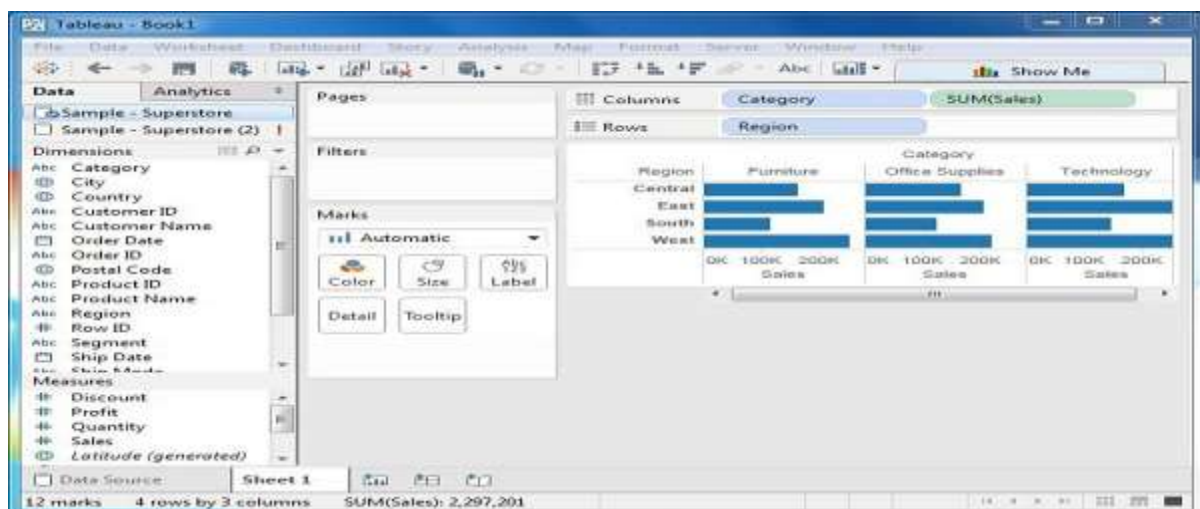
Next, choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help visualize the performance of the dimensional data with respect to the data which are measures. Choose **Category** and **Region**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



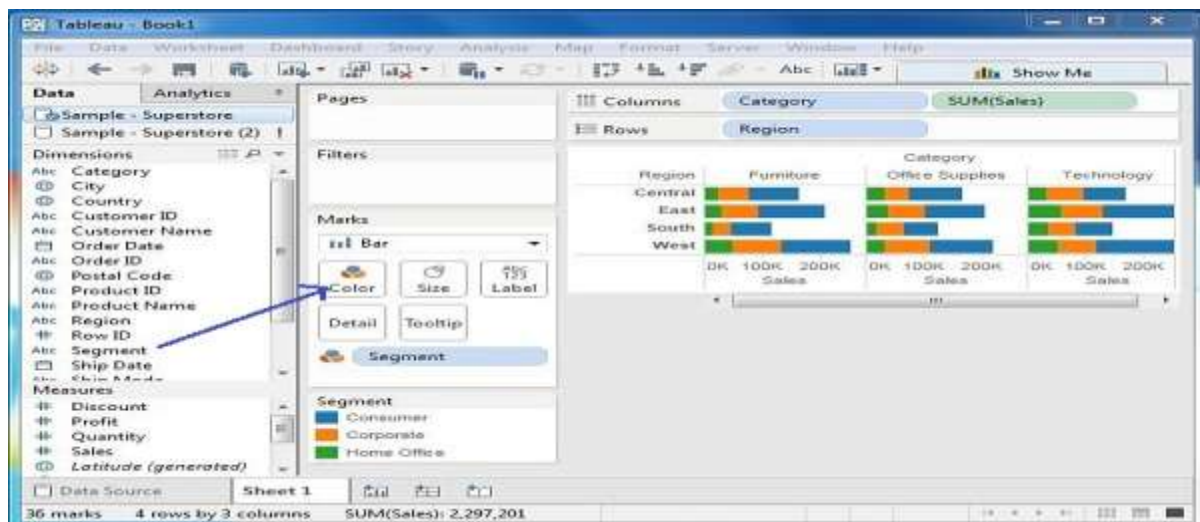
Apply Visualization Technique

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

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



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



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

FAQ's:

1. What are the differences between Tableau desktop and Tableau Server?
2. What are fact table and Dimension table in Tableau?
3. How many maximum tables can you join in Tableau?
4. How to add Custom Color to Tableau?