**BIG DATA ANALYTICS LAB MANUAL**

**III-B Tech – I Semester [Branch: CSE-DS]**

**JNTUH-SYLLABUS**

**III Year B.Tech.CSE(DS). II – Sem L T P C**

**Course Code: 0 0 2 1**

**BIG DATA ANALYTICTS LAB MANUAL**

**Course Objectives**

1. The purpose of this course is to provide the students with the knowledge of Big data Analytics principles and techniques.
2. This course is also designed to give an exposure of the frontiers of Big data Analytics

**Course Outcomes**

1. Use Excel as an Analytical tool and visualization tool.
2. Ability to program using HADOOP and Map reduce.
3. Ability to perform data analytics using ML in R.
4. Use cassandra to perform social media analytics.

**List of Experiments**

1. Implement a simple map-reduce job that builds an inverted index on the set of input documents (Hadoop)
2. Process big data in HBase
3. Store and retrieve data in Pig
4. Perform Social media analysis using cassandra
5. Buyer event analytics using Cassandra on suitable product sales data.
6. Using Power Pivot (Excel) Perform the following on any dataset
7. Big Data Analytics
8. Big Data Charting
9. Use R-Project to carry out statistical analysis of big data
10. Use R-Project for data visualization of social media data

**TEXT BOOKS:**

1. Big Data Analytics, Seema Acharya, Subhashini Chellappan, Wiley 2015.
2. Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today’s Business, Michael Minelli, Michehe Chambers, 1st Edition, Ambiga Dhiraj, Wiely CIO Series, 2013.
3. Hadoop: The Definitive Guide, Tom White, 3rd Edition, O‟Reilly Media, 2012.
4. Big Data Analytics: Disruptive Technologies for Changing the Game, Arvind Sathi, 1st Edition,

IBM Corporation, 2012.

**REFERENCES:**

1. Big Data and Business Analytics, Jay Liebowitz, Auerbach Publications, CRC press (2013).
2. Using R to Unlock the Value of Big Data: Big Data Analytics with Oracle R Enterprise and Oracle R Connector for Hadoop, Tom Plunkett, Mark Hornick, McGraw-Hill/Osborne Media (2013), Oracle press.
3. Professional Hadoop Solutions, Boris lublinsky, Kevin t. Smith, Alexey Yakubovich, Wiley, ISBN: 9788126551071, 2015.
4. Understanding Big data, Chris Eaton, Dirk deroos et al., McGraw Hill, 2012.
5. Intelligent Data Analysis, Michael Berthold, David J. Hand, Springer, 2007.
6. Taming the Big Data Tidal Wave: Finding Opportunities in Huge Data Streams with Advanced Analytics, Bill Franks, 1st Edition, Wiley and SAS Business Series, 2012.

**INDEX**

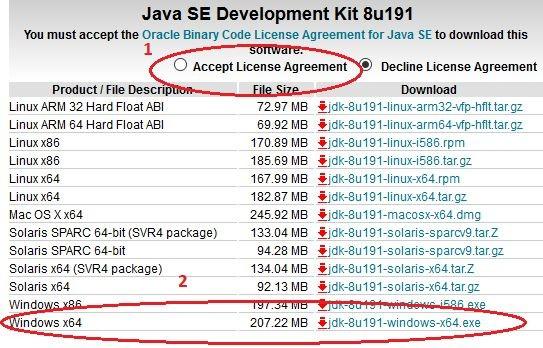
|  |  |  |
| --- | --- | --- |
| **S.No** | **Program Name** | **Page No.** |
|  | | |
| 1. | Write the Hadoop Installation Procedure on windows systems |  |
| 2. | Write Hadoop’s Basic Shell Commands used for its Framework. |  |
| 3. | Implement a simple map-reduce job that builds an inverted index on the set of input  documents (Hadoop) |  |
| 4. | Process big data in HBase |  |
| 5. | Store and retrieve data in Pig |  |
| 6. | Perform Social media analysis using Cassandra |  |
| 7. | Buyer event analytics using Cassandra on suitable product sales data. |  |
| 8. | Using Power Pivot (Excel) Perform the following on any dataset   1. Big Data Analytics 2. Big Data Charting |  |
| 9. | Use R-Project to carry out statistical analysis of big data |  |
| 10. | Use R-Project for data visualization of social media data |  |

Experiment-01

Implement Hadoop step-by-step

## Preparations

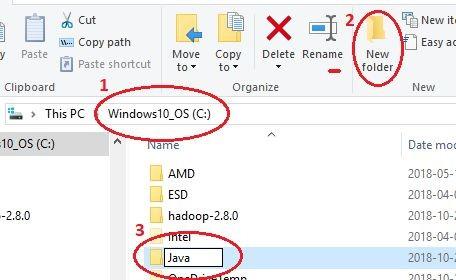
1. Make sure that you are using Windows 10 and are logged in as admin.
2. Download Java jdk1.8.0 from <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
3. Accept Licence Agreement [**1**] and download the exe-file [**2**]



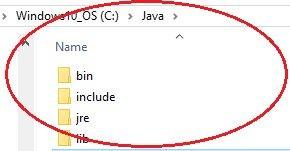
1. Download Hadoop 2.8.0 from <http://archive.apache.org/dist/hadoop/core//hadoop-2.8.0/hadoop-2.8.0.tar.gz>
2. Download Notepad++ from

[https://notepad-plus-plus.org](https://notepad-plus-plus.org/) (current version for Windows)

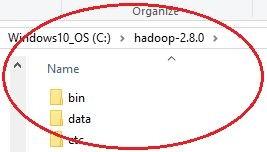
F.Navigate to C:\ [**1**], make a New folder [**2**] and name it Java [**3**]



G. Run the Java installation file **jdk-8u191-windows-x64**. Install direct in the folder **C:\Java**, or move the items from the folder **jdk1.8.0** to the folder **C:\Java**. It should look like this:



H. Install Hadoop 2.8.0 right under C:\ like this:



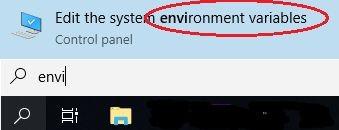


If Windows Defender Firewall is activated on your PC, then you must at least open the two **ports 8088** and **50070.** If the firewall is deactivated you can skip this step. Else, go to <https://www.windowscentral.com/how-open-port-windows-firewall>

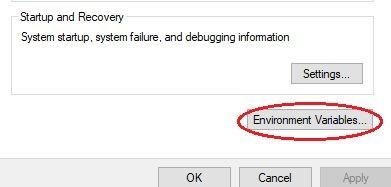
follow the instructions and opens the two ports.

## 1.1 Setup Environment variables

1. Use the search-function to find the environment variables.

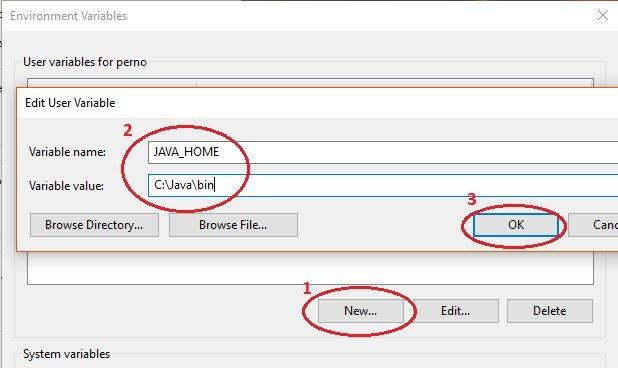


In System properties, click the button Environment Variables...

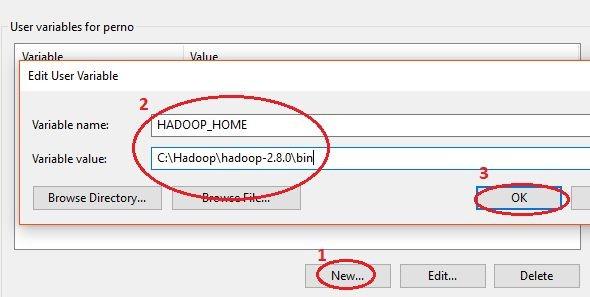


A new window will open with two tables and buttons. The upper table is for User variables and the lower for System variables.

1. Make a New User variable [**1**]. Name it JAVA\_HOME and set it to the Java bin-folder [**2**]. Click OK [**3**].

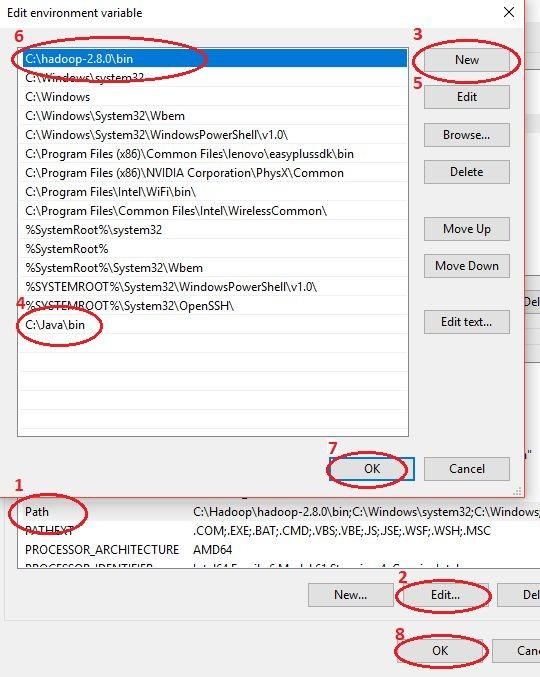


1. Make another New User variable [**1**]. Name it HADOOP\_HOME and set it to the hadoop-2.8.0 bin-folder [**2**]. Click OK [**3**].

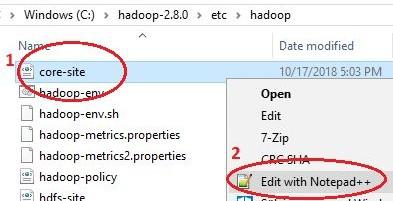


1. Now add Java and Hadoop to System variables path: Go to path [**1**] and click edit [**2**]. The editor window opens. Chose New [**3**] and add the address C:\Java\bin [**4**]. Chose New again

[**5**] and add the address C:\hadoop-2.8.0\bin [**6**]. Click OK [**7**] in the editor window and OK

[**8**] to change the System variables.

## 1.1Configuration

1. Go to the file **C:\Hadoop\Hadoop-2.8.0\etc\hadoop\core-site.xml** [**1**]. Right-click on the file and edit with Notepad++ [**2**].
2. In the end of the file you have two configuration tags.

<configuration>

</configuration>

Paste the following code between the two tags and save (spacing doesn’t matter):

<property>

<name>fs.defaultFS</name>

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

</property>

It should look like this in Notepad++:



1. Rename **C:\Hadoop-2.8.0\etc\hadoop\mapred-site.xml.template** to **mapred-site.xml** and edit this file with Notepad++. Paste the following code between the configuration tags and save:

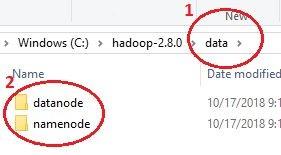
<property>

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

<value>yarn</value>

</property>

1. Under C:\Hadoop-2.8.0 create a folder named data [**1**] with two subfolders, “datanode” and “namenode” [**2**].



1. Edit the file **C:\Hadoop-2.8.0\etc\hadoop/hdfs-site.xml** with Notepad++. Paste the following code between the configuration tags and save:

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

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

<value>C:\hadoop-2.8.0\data\namenode</value>

</property>

<property>

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

<value>C:\hadoop-2.8.0\data\datanode</value>

</property>

1. Edit the file **C:\Hadoop-2.8.0\etc\hadoop\yarn-site.xml** with Notepad++. Paste the following code between the configuration tags and save:

<property>

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

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

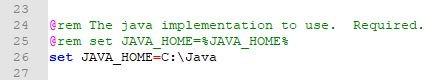
<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

1. Edit the file **C:\Hadoop-2.8.0\etc/hadoop\hadoop-env.cmd** with Notepad++. Write **@rem** in front of “set JAVA\_HOME=%JAVA\_HOME%”.

Write **set JAVA\_HOME=C:\Java** at the next row.

It should look like this is Notepad++:



Don’t forget to save.

**Bravo**, configuration done!

## Replace the bin folder

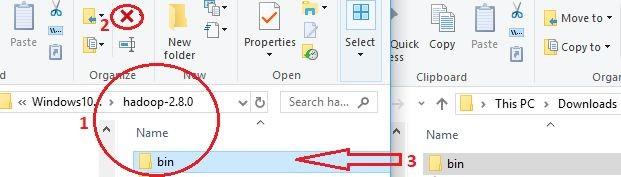
Before we can start testing we must exchange a folder in Hadoop.

1. Download **Hadoop Configuration.zip**

[https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-10/blob](https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-10/blob/master/Hadoop%20Configuration.zip)

[/master/Hadoop%20Configuration.zip](https://github.com/MuhammadBilalYar/HADOOP-INSTALLATION-ON-WINDOW-10/blob/master/Hadoop%20Configuration.zip) Unzip the bin file.

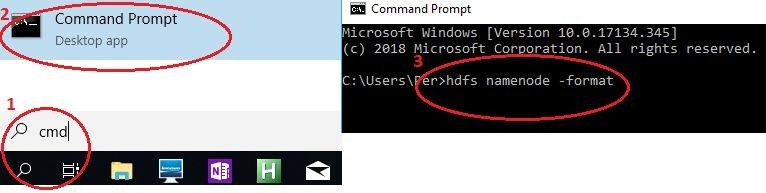
1. Delete the bin file **C:\Hadoop\Hadoop-2.8.0\bin** [**1, 2**] and replace it with the new bin-folder from Hadoop Configuration.zip. [**3**].



## 1.1Testing

1. Search for cmd [**1**] and open the Command Prompt [**2**]. Write

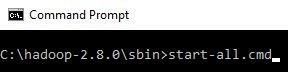
**hdfs namenode –format** [**3**] and push enter.



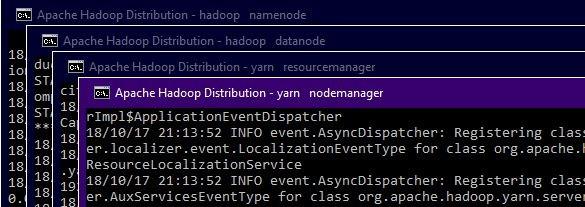
If this first test works the Command Prompt will run a lot of information. It is a good sign!

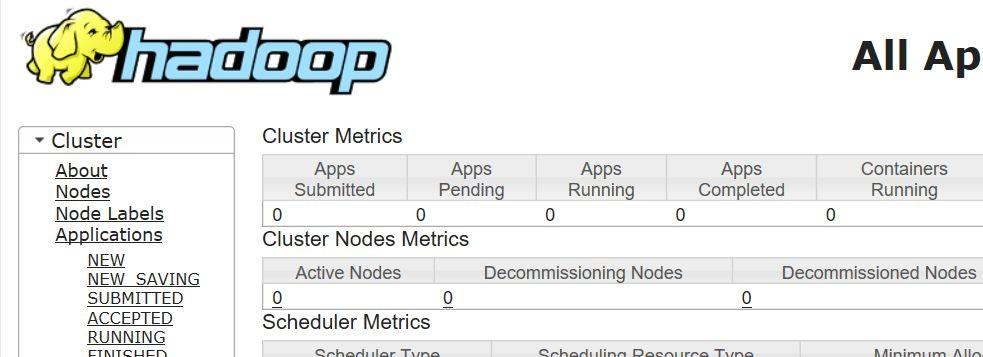
1. Now you must change directory in the Command Prompt. Write **cd C:\hadoop-2.8.0\sbin**

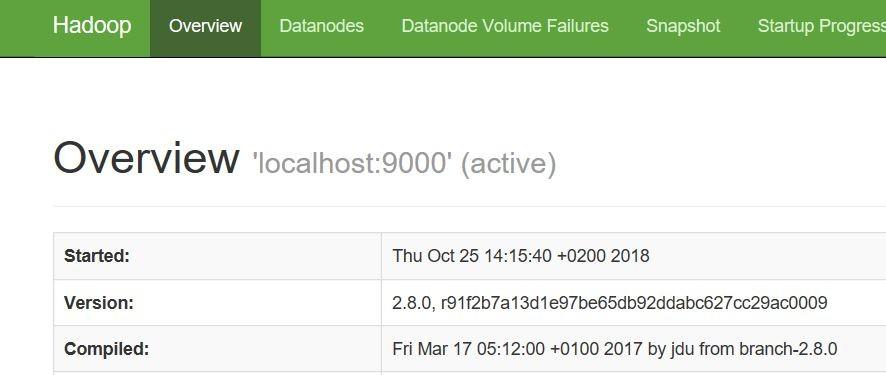
And push enter. In the sbin folder, write **start-all.cmd** and push enter.



If the configuration is right, four apps will start running and it will look something like this:



1. Now open a browser and write in the address field **localhost:8088** and push enter. Can you see the little hadoop elephant? Then you have made a really good work!
2. Last test - try to write **localhost:50070** instead.



If you can see the overview you have implemented Hadoop on your PC.

#### Congratulations, you did it!!!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*To close the running programs, run “stop-all.cmd” in tho command prompt*

Experiment-02

Hadoop Shell Commands

# DFShell

The HDFS shell is invoked by bin/hadoop dfs <args>. All the HDFS shell commands take path URIs as arguments. The URI format is *scheme://autority/path*. For HDFS the scheme is *hdfs*, and for the local filesystem the scheme is *file*. The scheme and authority are optional. If not specified, the default scheme specified in the configuration is used. An HDFS file or directory such as */parent/child* can be specified as *hdfs://namenode:namenodeport/parent/child* or simply as */parent/child* (given that your configuration is set to point to *namenode:namenodeport*). Most of the commands in HDFS shell behave like corresponding Unix commands. Differences are described with each of the commands. Error information is sent to *stderr* and the output is sent to *stdout*.

# cat

Usage: hadoop dfs -cat URI [URI …]

Copies source paths to *stdout*. Example:

* hadoop dfs -cat hdfs://host1:port1/file1 hdfs://host2:port2/file2
* hadoop dfs -cat file:///file3 /user/hadoop/file4

Exit Code:

Returns 0 on success and -1 on error.

# chgrp

Usage: hadoop dfs -chgrp [-R] GROUP URI [URI …]

Change group association of files. With -R, make the change recursively through the directory structure. The user must be the owner of files, or else a super-user. Additional information is in the Permissions User Guide.

# chmod

Usage: hadoop dfs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI …]

Change the permissions of files. With -R, make the change recursively through the directory structure. The user must be the owner of the file, or else a super-user. Additional information

is in the Permissions User Guide.

# chown

Usage: hadoop dfs -chown [-R] [OWNER][:[GROUP]] URI [URI ]

Change the owner of files. With -R, make the change recursively through the directory structure. The user must be a super-user. Additional information is in the Permissions User Guide.

# copyFromLocal

Usage: hadoop dfs -copyFromLocal <localsrc> URI

Similar to **put** command, except that the source is restricted to a local file reference.

# copyToLocal

Usage: hadoop dfs -copyToLocal [-ignorecrc] [-crc] URI

<localdst>

Similar to **get** command, except that the destination is restricted to a local file reference.

# cp

Usage: hadoop dfs -cp URI [URI …] <dest>

Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.

Example:

* hadoop dfs -cp /user/hadoop/file1 /user/hadoop/file2
* hadoop dfs -cp /user/hadoop/file1 /user/hadoop/file2

/user/hadoop/dir

Exit Code:

Returns 0 on success and -1 on error.

# du

Usage: hadoop dfs -du URI [URI …]

Displays aggregate length of files contained in the directory or the length of a file in case its

just a file.

Example:

hadoop dfs -du /user/hadoop/dir1 /user/hadoop/file1 hdfs://host:port/user/hadoop/dir1

Exit Code:

Returns 0 on success and -1 on error.

# dus

Usage: hadoop dfs -dus <args>

Displays a summary of file lengths.

# expunge

Usage: hadoop dfs -expunge

Empty the Trash. Refer to HDFS Design for more information on Trash feature.

# get

Usage: hadoop dfs -get [-ignorecrc] [-crc] <src> <localdst>

Copy files to the local file system. Files that fail the CRC check may be copied with the

-ignorecrc option. Files and CRCs may be copied using the -crc option. Example:

* hadoop dfs -get /user/hadoop/file localfile
* hadoop dfs -get hdfs://host:port/user/hadoop/file localfile

Exit Code:

Returns 0 on success and -1 on error.

# getmerge

Usage: hadoop dfs -getmerge <src> <localdst> [addnl]

Takes a source directory and a destination file as input and concatenates files in src into the destination local file. Optionally addnl can be set to enable adding a newline character at the end of each file.

# ls

Usage: hadoop dfs -ls <args>

For a file returns stat on the file with the following format:

filename <number of replicas> filesize modification\_date modification\_time permissions userid groupid

For a directory it returns list of its direct children as in unix. A directory is listed as:

dirname <dir> modification\_time modification\_time permissions userid groupid

Example:

hadoop dfs -ls /user/hadoop/file1 /user/hadoop/file2 hdfs://host:port/user/hadoop/dir1 /nonexistentfile Exit Code:

Returns 0 on success and -1 on error.

# lsr

Usage: hadoop dfs -lsr <args>

Recursive version of ls. Similar to Unix ls -R.

# mkdir

Usage: hadoop dfs -mkdir <paths>

Takes path uri's as argument and creates directories. The behavior is much like unix mkdir -p creating parent directories along the path.

Example:

* hadoop dfs -mkdir /user/hadoop/dir1 /user/hadoop/dir2
* hadoop dfs -mkdir hdfs://host1:port1/user/hadoop/dir hdfs://host2:port2/user/hadoop/dir

Exit Code:

Returns 0 on success and -1 on error.

# movefromLocal

Usage: dfs -moveFromLocal <src> <dst>

Displays a "not implemented" message.

# mv

Usage: hadoop dfs -mv URI [URI …] <dest>

Moves files from source to destination. This command allows multiple sources as well in which case the destination needs to be a directory. Moving files across filesystems is not permitted.

Example:

* hadoop dfs -mv /user/hadoop/file1 /user/hadoop/file2
* hadoop dfs -mv hdfs://host:port/file1 hdfs://host:port/file2 hdfs://host:port/file3 hdfs://host:port/dir1

Exit Code:

Returns 0 on success and -1 on error.

# put

Usage: hadoop dfs -put <localsrc> ... <dst>

Copy single src, or multiple srcs from local file system to the destination filesystem. Also reads input from stdin and writes to destination filesystem.

* hadoop dfs -put localfile /user/hadoop/hadoopfile
* hadoop dfs -put localfile1 localfile2 /user/hadoop/hadoopdir
* hadoop dfs -put localfile hdfs://host:port/hadoop/hadoopfile
* hadoop dfs -put - hdfs://host:port/hadoop/hadoopfile

Reads the input from stdin.

Exit Code:

Returns 0 on success and -1 on error.

# rm

Usage: hadoop dfs -rm URI [URI …]

Delete files specified as args. Only deletes non empty directory and files. Refer to rmr for recursive deletes.

Example:

* hadoop dfs -rm hdfs://host:port/file /user/hadoop/emptydir

Exit Code:

Returns 0 on success and -1 on error.

# rmr

Usage: hadoop dfs -rmr URI [URI …]

Recursive version of delete. Example:

* hadoop dfs -rmr /user/hadoop/dir
* hadoop dfs -rmr hdfs://host:port/user/hadoop/dir

Exit Code:

Returns 0 on success and -1 on error.

# setrep

Usage: hadoop dfs -setrep [-R] <path>

Changes the replication factor of a file. -R option is for recursively increasing the replication factor of files within a directory.

Example:

* hadoop dfs -setrep -w 3 -R /user/hadoop/dir1

Exit Code:

Returns 0 on success and -1 on error.

# stat

Usage: hadoop dfs -stat URI [URI …]

Returns the stat information on the path. Example:

* hadoop dfs -stat path

Exit Code:

Returns 0 on success and -1 on error.

***24.tail***

Usage: hadoop dfs -tail [-f] URI

Displays last kilobyte of the file to stdout. -f option can be used as in Unix. Example:

* hadoop dfs -tail pathname

Exit Code:

Returns 0 on success and -1 on error.

# test

Usage: hadoop dfs -test -[ezd] URI

Options:

-e check to see if the file exists. Return 0 if true.

-z check to see if the file is zero length. Return 0 if true

-d check return 1 if the path is directory else return 0.

Example:

* hadoop dfs -test -e filename

1. **text**

Usage: hadoop dfs -text <src>

Takes a source file and outputs the file in text format. The allowed formats are zip and TextRecordInputStream.

1. **touchz**

Usage: hadoop dfs -touchz URI [URI …]

Create a file of zero length. Example:

* hadoop -touchz pathname

Exit Code:

Returns 0 on success and -1 on error.

**Experiment 3 : Implement a simple map-reduce job that builds an inverted index on the set of input documents (Hadoop)**

Aim: To implement an Inverted index on Hadoop. Resources:Hadoop,Java,Eclipse

Theory; Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others. In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data.

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

##### Hadoop Architecture

At its core, Hadoop has two major layers namely −

* + Processing/Computation layer (MapReduce), and
  + Storage layer (Hadoop Distributed File System).

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each o昀昀ering local computation and storage.



##### MapReduce

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.

Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.

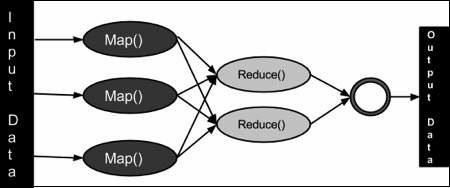
As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers.

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

##### The Algorithm

* + Generally the MapReduce paradigm is based on sending the computer to where the data resides!
  + MapReduce program executes in three stages, namely map stage, shu昀툀e stage, and reduce stage.
    - **Map stage** − The map or mapper’s job is to process the input data. Generally the input data is in the form of 昀椀le or directory and is stored in the Hadoop 昀椀le system (HDFS). The input 昀椀le is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
  + During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
  + The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
  + Most of the computing takes place on nodes with data on local disks that reduces the network tra昀케c.
  + After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



##### Inputs and Outputs (Java Perspective)

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

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

→ <k2, v2> → reduce → <k3, v3>(Output).

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

**Steps to run WordCount Program on Hadoop:**

1. Make sure Hadoop and Java are installed properly

**hadoop version**

**javac - version**

1. Create a directory on the Desktop named Lab and inside it create two folders; one called “Input” and the other called “tutorial\_classes”.

[You can do this step using GUI normally or through terminal commands]

**cd Desktop**

**mkdir Lab mkdir Lab/Input**

**mkdir Lab/tutorial\_classes**

1. Add the file attached with this document “WordCount.java” in the directory Lab
2. Add the file attached with this document “input.txt” in the directory Lab/Input.
3. Type the following command to export the hadoop classpath into bash.

**export HADOOP\_CLASSPATH=$(hadoop classpath)**

Make sure it is now exported.

**echo $HADOOP\_CLASSPATH**

1. It is time to create these directories on HDFS rather than locally. Type the following commands.

**hadoop fs -mkdir /WordCountTutorial hadoop fs -mkdir**

**/WordCountTutorial/Input**

**hadoop fs -put Lab/Input/input.txt /WordCountTutorial/Input**

1. Go to localhost:9870 from the browser, Open “Utilities → Browse File System” and you should see the directories and 昀椀les we placed in the 昀椀le system.
2. Then, back to local machine where we will compile the WordCount.java file. Assuming we are currently in the Desktop directory.

**cd Lab**

**javac -classpath $HADOOP\_CLASSPATH -d tutorial\_classes WordCount.java**

Put the output files in one jar file (There is a dot at the end)

**jar -cvf WordCount.jar -C tutorial\_classes .**

1. Now, we run the jar file on Hadoop.

**hadoop jar WordCount.jar WordCount /WordCountTutorial/Input**

**/WordCountTutorial/Output**

1. Output the result:

**hadoop dfs -cat /WordCountTutorial/Output/\***



**Program:**

**First Create Indexmapper.java class**

Packagemr03.inverted\_index;

import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.lib.input.FileSplit;

import java.io.IOException;

import java.util.StringTokenizer;

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

private final Text wordAtFileNameKey = new Text(); private final Text ONE\_STRING = new Text("1");

@Override

protected void map(LongWritable key, Text value,

Context context) throws IOException, InterruptedException {

FileSplit split = (FileSplit) context.getInputSplit();

StringTokenizer tokenizer = new StringTokenizer(value.toString());

while (tokenizer.hasMoreTokens()) { String fileName =

split.getPath().getName().split("\\.")[0];

//remove special char using

// tokenizer.nextToken().replaceAll("[^a- zA-

Z]", "").toLowerCase()

//check for empty words

wordAtFileNameKey.set(tokenizer.nextToken () +

"@" + fileName);

context.write(wordAtFileNameKey, ONE\_STRING);

}

}

}

**IndexReducer.java**

package mr03.inverted\_index;

import

org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class IndexReducer extends Reducer<Text, Text, Text, Text>

{

private final Text allFilesConcatValue = new Text();

@Override

protected void reduce(Text key, Iterable<Text> values,

Context context) throws java.io.IOException ,InterruptedException {

StringBuilder filelist = new StringBuilder("");

for(Text value:values) { filelist.append(value.toString()).append(";");

}

allFilesConcatValue.set(filelist.toString()); context.write(key, allFilesConcatValue);

};

}

**IndexDriver.java**

package mr03.inverted\_index;

import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.mapreduce.Job; import

org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.Text;

public class IndexDriver {

public static void main(String[] args) throws Exception

{

if (args.length != 2) {

System.err.println("Usage IndexDriver

<input\_dir>

<output\_dir>");

System.exit(2);

}

Configuration conf = new Configuration(); String input = args[0];

String output = args[1];

FileSystem fs = FileSystem.get(conf);

boolean exists = fs.exists(new Path(output));

if(exists) {

fs.delete(new Path(output), true);

}

Job job = Job.getInstance(conf); job.setJarByClass(IndexDriver.class);

job.setMapperClass(IndexMapper.class); job.setCombinerClass(IndexCombiner.class); job.setReducerClass(IndexReducer.class);

job.setOutputKeyClass(Text.class); job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(input));

FileOutputFormat.setOutputPath(job, new Path(output)); System.exit(job.waitForCompletion(true)?

0:1);

}

}

**IndexCombiner.java**

package mr03.inverted\_in de x;

import java.io.IOException; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class IndexCombiner extends Reducer<Text, Text, Text, Text>

{

private final Text fileAtWordFreqValue = new Text();

@Override

protected void reduce(Text key, java.lang.Iterable<Text> values,

,InterruptedException { int sum = 0;

Context context) throws IOException

for(Text value:values)

{

sum += Integer.parseInt(value.toString());

}

int splitIndex = key.toString().indexOf("@");

fileAtWordFreqValue.set(key.toString().substring(splitIndex+1)

+":"

+sum);

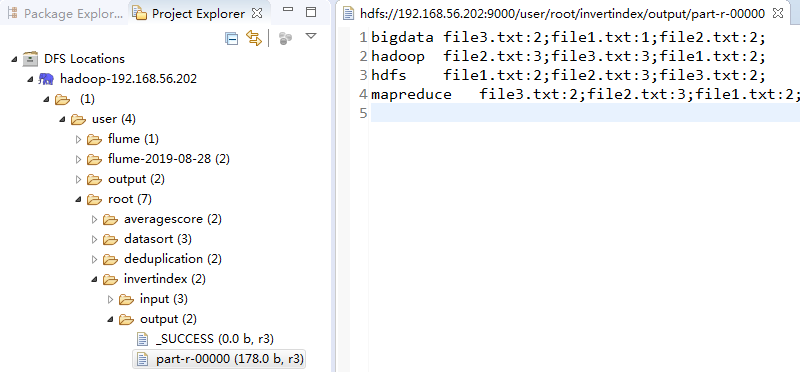
key.set(key.toString().substring(0,splitInde x));

context.write(key, fileAtWordFreqValue);

}

}

**Output:**



Experiment 4. Process big data in HBase

Aim:To create a table and process the big data in Hbase Resources:Hadoop,oracle virtual box,Hbase

Theory:

Hbase is an open source and sorted map data built on Hadoop. It is column oriented and horizontally scalable.

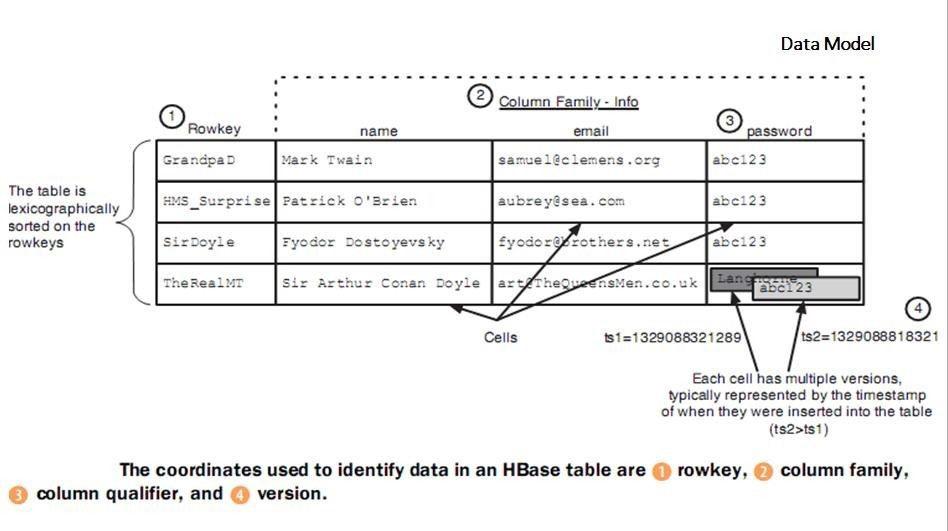
It is based on Google's Big Table.It has set of tables which keep data in key value format. Hbase is well suited for sparse data sets which are very common in big data use cases. Hbase provides APIs enabling development in practically any programming language. It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.

* + RDBMS get exponentially slow as the data becomes large
  + Expects data to be highly structured, i.e. ability to fit in a well-defined schema
  + Any change in schema might require a downtime
  + For sparse datasets, too much of overhead of maintaining NULL values

**Features of Hbase**

* + Horizontally scalable: You can add any number of columns anytime.
  + Automatic Failover: Automatic failover is a resource that allows a system administrator to automatically switch data handling to a standby system in the event of system compromise
  + Integrations with Map/Reduce framework: Al the commands and java codes internally implement Map/ Reduce to do the task and it is built over Hadoop Distributed File System.
  + sparse, distributed, persistent, multidimensional sorted map, which is indexed by rowkey, column key,and timestamp.
  + Often referred as a key value store or column family-oriented database, or storing versioned maps of maps.
  + fundamentally, it's a platform for storing and retrieving data with random access.
  + It doesn't care about datatypes(storing an integer in one row and a string in another for the same column).
  + It doesn't enforce relationships within your data.
  + It is designed to run on a cluster of computers, built using commodity hardware.

Cloudera VM is recommended as it has Hbase pre installed on it. Starting Hbase: Type Hbase shell in terminal to start the hbase.

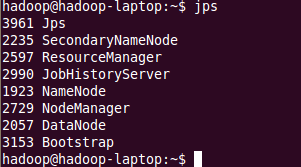


Cloudera VM is recommended as it has Hbase pre installed on it.

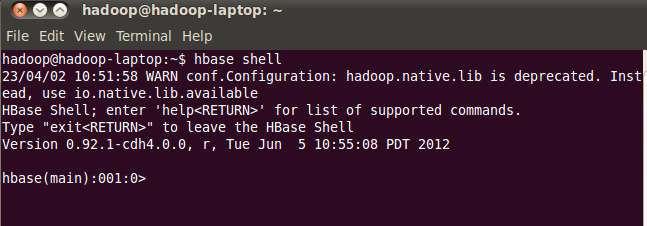
Hbase commands

Step 1:First go to terminal and type **StartCDH.sh**

Step 2:Next type **jps** command in the terminal

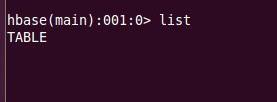


Step 3:Type **hbase shell**



Step 4:hbase(main):001:0> **list**

List will gives you list of tables in Hbase



Step 5:hbase(main):001:0>**version**



Version will gives you the version of hbase

**Create Table Syntax**

CREATE 'name\_space:table\_name', 'column\_family’

**hbase(main):011:0> create 'newtbl','knowledge' hbase(main):011:0>describe 'newtbl' hbase(main):011:0>status**

**1 servers, 0 dead, 15.0000 average load**

HBase – Using PUT to Insert data to Table

To insert data into the HBase table use PUT command, this would be similar to insert statement on RDBMS but the syntax is completely different. In this article I will describe how to insert data into the HBase table with examples using the PUT command from the HBase shell.

**HBase PUT command syntax**

Below is the syntax of the PUT command which is used to insert data (rows and columns) into a HBase table.

**HBase PUT command syntax**

Below is the syntax of the PUT command which is used to insert data (rows and columns) into a HBase table.

**put '<name\_space:table\_name>', '<row\_key>' '<cf:column\_name>', '<value>'**

hbase(main):015:0> **put 'newtbl','r1','knowledge:sports','cricket'**

0 row(s) in 0.0150 seconds

hbase(main):016:0> **put 'newtbl','r1','knowledge:science','chemistry'**

0 row(s) in 0.0040 seconds

hbase(main):017:0> **put 'newtbl','r1','knowledge:science','physics'**

0 row(s) in 0.0030 seconds

hbase(main):018:0> **put 'newtbl','r2','knowledge:economics','macroeconomics'**

0 row(s) in 0.0030 seconds

hbase(main):019:0> **put 'newtbl','r2','knowledge:music','songs'**

0 row(s) in 0.0170 seconds hbase(main):020:0> **scan 'newtbl'**

ROW COLUMN+CELL

r1 column=knowledge:science, timestamp=1678807827189, value

=physics

r1 column=knowledge:sports, timestamp=1678807791753, value= cricket

r2 column=knowledge:economics, timestamp=1678807854590, val

ue=macroeconomics

r2 column=knowledge:music, timestamp=1678807877340, value=s ongs

2 row(s) in 0.0250 seconds To retrieve only the row1 data

hbase(main):023:0> **get 'newtbl', 'r1'**

**output**

COLUMN CELL

knowledge:science timestamp=1678807827189, value=physics knowledge:sports timestamp=1678807791753, value=cricket 2 row(s) in 0.0150 seconds.



Veri昀椀cation

After disabling the table, you can still sense its existence through **list** and **exists** commands. You cannot scan it. It will give you the following error.



is\_disabled

This command is used to find whether a table is disabled. Its syntax is as follows.





disable\_all

This command is used to disable all the tables matching the given regex. The syntax for **disable\_all** command is given below.



Suppose there are 5 tables in HBase, namely raja, rajani, rajendra, rajesh, and raju. The following code will disable all the tables starting with **raj.**



##### Enabling a Table using HBase Shell

Syntax to enable a table:



Example

Given below is an example to enable a table.



Veri昀椀cation

After enabling the table, scan it. If you can see the schema, your table is successfully enabled.



##### is\_enabled

This command is used to find whether a table is enabled. Its syntax is as follows:



The following code verifies whether the table named **emp** is enabled. If it is enabled, it will return true and if not, it will return false.



##### describe

This command returns the description of the table. Its syntax is as follows:





**Experiment: 5 Store and retrieve data in Pig**

###### Aim:To perform storing and retrieval of big data using Apache pig Resources:Apache pig

Theory:

Pig is a platform that works with large data sets for the purpose of analysis. The Pig dialect is called Pig Latin, and the Pig Latin commands get compiled into MapReduce jobs that can be run on a suitable platform, like Hadoop.

**Apache Pig** is a platform for analyzing large data sets that consists of a high- level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin, which has the following key properties:

* + **Ease of programming.** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
  + **Optimization opportunities.** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
  + **Extensibility.** Users can create their own functions to do special-purpose processing.
  + Pig Latin – Relational Operations
  + The following table describes the relational operators of Pig Latin.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **Loading and Storing** | |
| LOAD | To Load the data from the 昀椀le system (local/HDFS) into a relation. |
| STORE | To save a relation to the 昀椀le system (local/HDFS). |

|  |  |
| --- | --- |
| **Filtering** | |
| FILTER | To remove unwanted rows from a relation. |
| DISTINCT | To remove duplicate rows from a relation. |
| FOREACH, GENERATE | To generate data transformations based on columns of data. |
| STREAM | To transform a relation using an external program. |
| **Grouping and Joining** | |
| JOIN | To join two or more relations. |
| COGROUP | To group the data in two or more relations. |
| GROUP | To group the data in a single relation. |
| CROSS | To create the cross product of two or more relations. |
| **Sorting** | |
| ORDER | To arrange a relation in a sorted order based on one or more 昀椀elds (ascending or descending). |
| LIMIT | To get a limited number of tuples from a relation. |
| **Combining and Splitting** | |
| UNION | To combine two or more relations into a single relation. |
| SPLIT | To split a single relation into two or more relations. |
| **Diagnostic Operators** | |
| DUMP | To print the contents of a relation on the console. |
| DESCRIBE | To describe the schema of a relation. |
| EXPLAIN | To view the logical, physical, or MapReduce execution plans to compute a relation. |
| ILLUSTRATE | To view the step-by-step execution of a series of statements. |

For the given Student dataset and Employee dataset,perform Rela琀椀onal opera琀椀ons like Loading, Storing, Diagnos琀椀c Opera琀椀ons (Dump, Describe, Illustrate & Explain) in Hadoop Pig framework using Cloudera

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student ID** | **First Name** | **Age** | **City** | **CGPA** |
| 001 | Jagruthi | 21 | Hyderabad | 9.1 |
| 002 | Praneeth | 22 | Chennai | 8.6 |
| 003 | Sujith | 22 | Mumbai | 7.8 |
| 004 | Sreeja | 21 | Bengaluru | 9.2 |
| 005 | Mahesh | 24 | Hyderabad | 8.8 |
| 006 | Rohit | 22 | Chennai | 7.8 |
| 007 | Sindhu | 23 | Mumbai | 8.3 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee ID** | **Name** | **Age** | **City** |
| 001 | Angelina | 22 | LosAngeles |
| 002 | Jackie | 23 | Beijing |
| 003 | Deepika | 22 | Mumbai |
| 004 | Pawan | 24 | Hyderabad |
| 005 | Rajani | 21 | Chennai |
| 006 | Amitabh | 22 | Mumbai |

Step-1: **Create a Directory**in HDFS with the name **pigdir** in the required path using **mkdir**:

$ hdfs dfs -mkdir /bdalab/pigdir

Step-2: The input 昀椀le of Pig contains each tuple/record in individual lines with the en琀椀琀椀es separated by a delimiter ( “,”).



Step-3: **Move the 昀椀le** from the local 昀椀le system to HDFS using **put (Or) copyFromLocal**

command and verify using -cat command

To get the path of the 昀椀le student\_data.txt type the below command readlink -f student\_data.txt

$ hdfs dfs -put /home/hadoop/Desktop/student\_data.txt /bdalab/pigdir/

$ hdfs dfs -cat /bdalab/pigdir/student\_data

$ hdfs dfs -put /home/hadoop/Desktop/employee\_data /bdalab/pigdir/

Step-4: **Apply Rela琀椀onal Operator – LOAD to load the data** from the 昀椀le student\_data.txt into Pig by execu琀椀ng the following Pig La琀椀n statement in the **Grunt shell**. Rela琀椀onal Operators are **NOT case sensi琀椀ve.**

**$ pig** => will direct to **grunt> shell**

grunt> student = LOAD ' /bdalab/pigdir/student\_data.txt' USING PigStorage(',')as ( id:int, name:chararray, age:int, city:chararray, cgpa:double );

grunt>employee = LOAD ' /bdalab/pigdir/employee\_data.txt’ USING PigStorage(',')as ( id:int, name:chararray, age:int, city:chararray);

Step-5: **Apply Rela琀椀onal Operator – STORE** to **Store the rela琀椀on** in the HDFS directory “/pig\_output/” as shown below.

grunt> STORE student INTO ' /bdalab/pigdir/pig\_output/ ' USING PigStorage (','); grunt> STORE employee INTO ' /bdalab/pigdir/pig\_output/ ' USING PigStorage (',');

Step-6: **Verify the stored data** as shown below

$ hdfs dfs -ls /bdalab/pigdir/pig\_output/

$ hdfs dfs -cat /bdalab/pigdir/pig\_output/**part-m-00000**

Step-7: **Apply Rela琀椀onal Operator – Diagnos琀椀c Operator – DUMP toPrint the contents of the rela琀椀on**.

grunt> Dump student grunt> Dump employee

Step-8: **Apply Rela琀椀onal Operator – Diagnos琀椀c Operator – DESCRIBE toView the schema of a rela琀椀on**.

grunt> Describe student grunt> Describe employee

Step-9: **Apply Rela琀椀onal Operator – Diagnos琀椀c Operator – EXPLAIN toDisplay the logical, physical, and MapReduce execu琀椀onplans** of a rela琀椀on using**Explain** operator

grunt> Explain student grunt>Explain employee

Step-9: **Apply Rela琀椀onal Operator – Diagnos琀椀c Operator – ILLUSTRATE to give the step- by-step execu琀椀on of a sequence of statements**

grunt>Illustrate student grunt>Illustrate employee

**Experiment 6. Perform Social media analysis using Cassandra Aim: To perform the social media data analysis using Cassandra Resources: Cassandra**

**Procedure:**

* + Apache Cassandra is an open-source distributed database management system designed to handle large amounts of data across many commodity servers.
  + Cassandra provides high availability with no single point of failure.
  + Cassandra o昀昀ers robust support for clusters spanning multiple data centers, with asynchronous master-less replication allowing low latency operations for all clients.

Cassandra is a distributed database for low latency, high throughput services that handle real time workloads comprising of hundreds of updates per second and tens of thousands of reads per second.

When looking to replace a key-value store with something more capable on the real-time replication and data distribution, research on Dynamo, the CAP theorem and eventual consistency model shows Cassandra 昀椀ts this model quite well. As one learns more about data modeling capabilities, we gradually move towards decomposing data.

If one is coming from a relational database background with strong ACID semantics, then one must take the time to understand the eventual consistency model.

Understand Cassandra’s architecture very well and what it does under the hood. With Cassandra 2.0 you get lightweight transaction and triggers, but they are not the same as the traditional database transactions one might be familiar with. For example, there are no foreign key constraints available – it has to be handled by one’s own application. Understanding one’s use cases and data access patterns clearly before modeling data with Cassandra and to read all the available documentation is a must.

#### Capture

This command captures the output of a command and adds it to a file. For example, take a look at the following code that captures the output to a file named **Outputfile**.



When we type any command in the terminal, the output will be captured by the file given. Given below is the command used and the snapshot of the output file.



You can turn capturing off using the following command.



#### Consistency

This command shows the current consistency level, or sets a new consistency level.

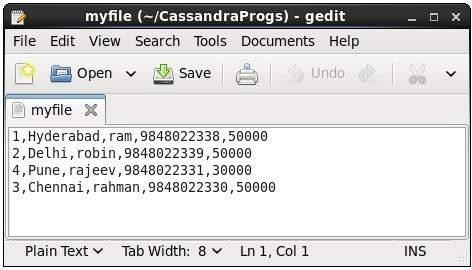


#### Copy

This command copies data to and from Cassandra to a file. Given below is an example to copy the table named **emp** to the file **myfile**.



If you open and verify the file given, you can find the copied data as shown below.



#### Describe

This command describes the current cluster of Cassandra and its objects. The variants of this command are explained below.

**Describe cluster** − This command provides information about the cluster.



**Describe Keyspaces** − This command lists all the keyspaces in a cluster. Given below is the usage of this command.



**Describe tables** − This command lists all the tables in a keyspace. Given below is the usage of this command.



**Describe table** − This command provides the description of a table. Given below is the usage of this command.





#### Describe Type

This command is used to describe a user-defined data type. Given below is the usage of this command.



#### Describe Types

This command lists all the user-defined data types. Given below is the usage of this command. Assume there are two user-defined data types: **card** and **card\_details**.



#### Expand

This command is used to expand the output. Before using this command, you have to turn the expand command on. Given below is the usage of this command.



**Note** − You can turn the expand option off using the following command.



#### Exit

This command is used to terminate the cql shell.

#### Show

This command displays the details of current cqlsh session such as Cassandra version, host, or data type assumptions. Given below is the usage of this command.



#### Source

Using this command, you can execute the commands in a file. Suppose our input file is as follows −



Then you can execute the file containing the commands as shown below.



**Experiment 7. Buyer event analytics using Cassandra on suitable product sales data.**

**Aim: To perform the buyer event analysis using Cassandra on sales data Resources Required: Apache Hadoop, Apache Cassandra**

**Theory:**

Users can access Cassandra through its nodes using Cassandra Query Language (CQL). CQL treats the database **(Keyspace)** as a container of tables. Programmers use **cqlsh:** a prompt to work with CQL or separate application language drivers.

Clients approach any of the nodes for their read-write operations. That node (coordinator) plays a proxy between the client and the nodes holding the data.

Write Operations

Every write activity of nodes is captured by the **commit logs** written in the nodes. Later the data will be captured and stored in the **mem-table.** Whenever the mem-table is full, data will be written into the **SStable** data file. All writes are automatically partitioned and replicated throughout the cluster. Cassandra periodically consolidates the SSTables, discarding unnecessary data.

Read Operations

During read operations, Cassandra gets values from the mem-table and checks the bloom filter to find the appropriate SSTable that holds the required data.

Apache is an open-source platform. This web server delivers web-related content using the internet. It has gained huge popularity over the last few years, as the most used web server software. Cassandra is a database management system that is open-source. It has the capacity to handle a large amount of data across servers. It was first developed by Facebook for the inbox search feature and was released as an open-source project back in 2008.

The following year, Cassandra became a part of Apache incubation, and combined with Apache, it has reached new heights. To put it in simple terms, Apache Cassandra is a powerful open-source distributed database system that can work efficiently to handle a massive amount of data across multiple servers.

Considering all the features of Apache Cassandra, be it Cassandra fault-tolerance, Cassandra data migration, Cassandra enterprise support, Cassandra cluster optimization and tuning, many organizations have opted for this product. Starting from big players in the market to startups, Cassandra has changed the way of database management. Let’s consider Netflix, the largest online streaming platform. Netflix has successfully provided updated data to its users day after day. Apache Cassandra has undeniably a huge role to play in this feat.

**DATA-MODELLING**

The way data is modeled is a major difference between Cassandra & MySQL. .

Let us consider a platform where users can post. Now, you have commented on a post of another user. In these two databases, the information will be stored differently. In Cassandra, you can store the data in a single table. The comments for each user is stored in the form of a list(as one single row).

In MySQL, you have to make two tables with one-to-many relationships between them. As MySQL does not permit unstructured data such as a List or a Map, one-to-many relationships are required among these tables.

**READ PERFORMANCE**

The query to retrieve the comments made by a user(for example ‘5’) in MySQL, will look like this.

SELECT \* from Users u, Comments c WHERE u.user\_id=c.user\_id and user\_id=5; When you utilize indexing in MySQL, it saves the data like a binary tree.

In Cassandra, it is surprisingly simple: SELECT \* from Users WHERE user\_id=3;

You only have to store a single row in Cassandra for a specific user\_id. It will require just one lookup.

**WRITE PERFORMANCE**

A search needs to be done with every INSERT/UPDATE/DELETE in MySQL. If you have to update a record with an existing primary key,

1. **It will first search for the row, and**
2. **Then update it**

Cassandra leverages an append-only model. Insert & update have no fundamental difference. If you want to insert a row that comes with the same primary key as an existing row, the row will be replaced. Or, if you update a row with a non-existent primary key, Cassandra will create the row. Cassandra is very fast and stores large swathes of data on commodity hardware without compromising the read efficiency in any way.

**TRANSACTIONS**

MySQL facilitates ACID transactions like any other Relational Database Management System

* + Atomicity
  + Consistency
  + Isolation
  + Durability

On the other hand, Cassandra has certain limitations to provide ACID transactions. Cassandra can achieve consistency if data duplication is not allowed. But, that will kill Cassandra’s availability. So, the systems that require ACID transactions must avoid NoSQL databases.

**Procedure:**

A sample query to insert a record into an Apache Cassandra table is as follows:



The same query in MongoDB will have an implementation as follows:



cqlsh>

SELECT TTL(name) FROM learn\_cassandra.todo\_by\_user\_email WHERE user\_email='john@email.com';

ttl(name)



43

(1 rows) cqlsh>

SELECT \* FROM learn\_cassandra.todo\_by\_user\_email WHERE user\_email='john@email.com';

user\_email | creation\_date | todo\_uuid | name

+ + +

(0 rows)

Let’s insert a new record: cqlsh>

INSERT INTO learn\_cassandra.todo\_by\_user\_email (user\_email, creation\_date, name) VALUES(' ('john@email.com', '2021-03-14 16:07:19.622+0000', 'Insert query');

cqlsh>

UPDATE learn\_cassandra.todo\_by\_user\_email SET name = 'Update query'

WHERE user\_email = 'john@email.com' AND creation\_date = '2021-03-14 16:10:19.622+0000';

2 new rows appear in our table: cqlsh>

SELECT \* FROM learn\_cassandra.todo\_by\_user\_email WHERE user\_email='john@email.com';

user\_email | creation\_date | name

+ +

[john@email.com](mailto:john@email.com) | 2021-03-14 16:10:19.622000+0000 | Update query [john@email.com](mailto:john@email.com) | 2021-03-14 16:07:19.622000+0000 | Insert query

(2 rows)

Let’s only update if an entry already exists, by using IF EXISTS: cqlsh>

UPDATE learn\_cassandra.todo\_by\_user\_email SET name = 'Update query with LWT'

WHERE user\_email = 'john@email.com' AND creation\_date = '2021-03-14 16:07:19.622+0000' IF EXISTS;

[applied]

True cqlsh>

INSERT INTO learn\_cassandra.todo\_by\_user\_email (user\_email,creation\_date,name) VALUES('john@email.com', toTimestamp(now()), 'Yet another entry') IF NOT EXISTS;

[applied] True

Experiment:8 - (a) using a power pivot(Excel) perform the following on any data set

Aim: To perform the big data analytics using power pivot in Excel

Resources: Microsoft Excel

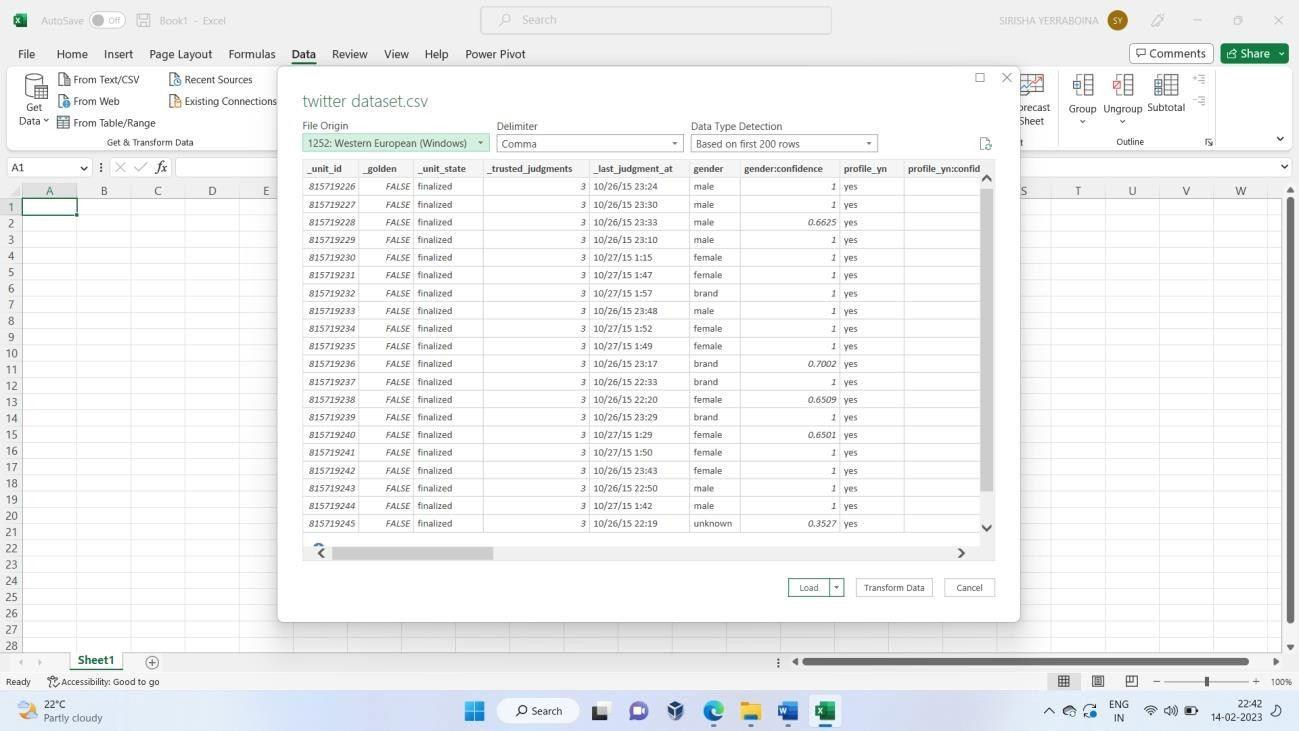
Theory: Power Pivot is an Excel add-in you can use to perform powerful data analysis and create sophisticated data models. With Power Pivot, you can mash up large volumes of data from various sources, perform information analysis rapidly, and share insights easily.

In both Excel and in Power Pivot, you can create a Data Model, a collection of tables with relationships. The data model you see in a workbook in Excel is the same data model you see in the Power Pivot window. Any data you import into Excel is available in Power Pivot, and vice versa.

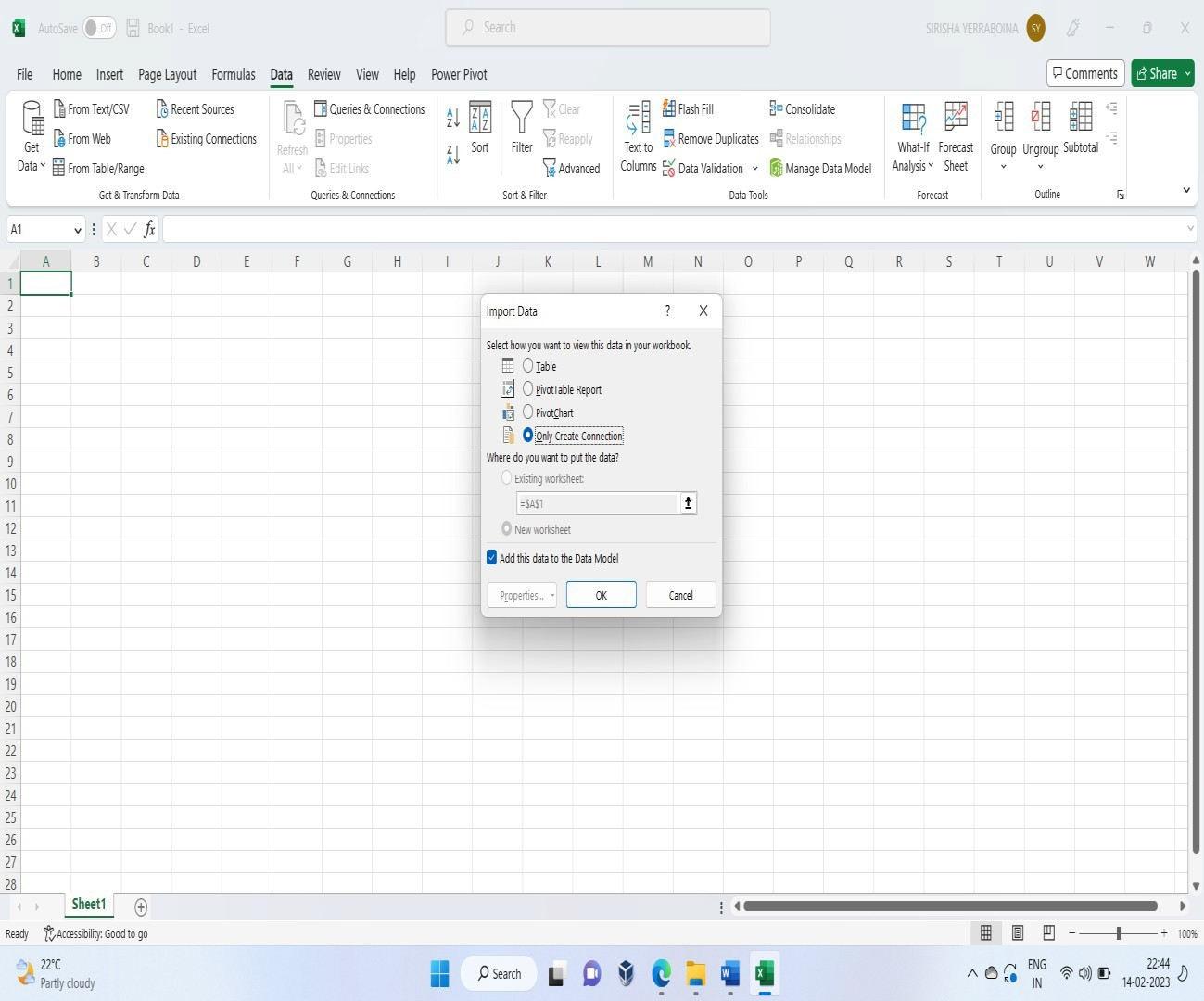
Procedure:

Open the Microsoft Excel and go to data menu and click get data

Import the Twitter data set and click load to button Now from the excel data will starts importing

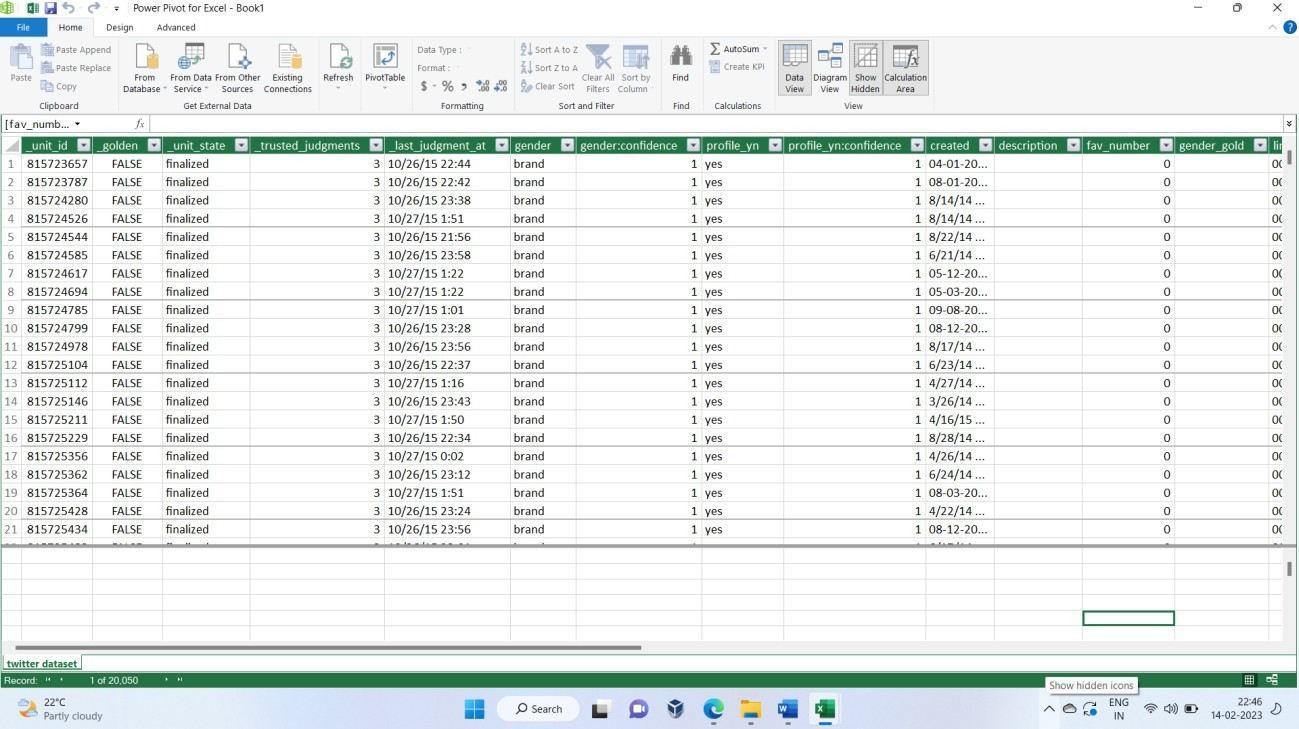


Next click create connection and click the check box add to the data model

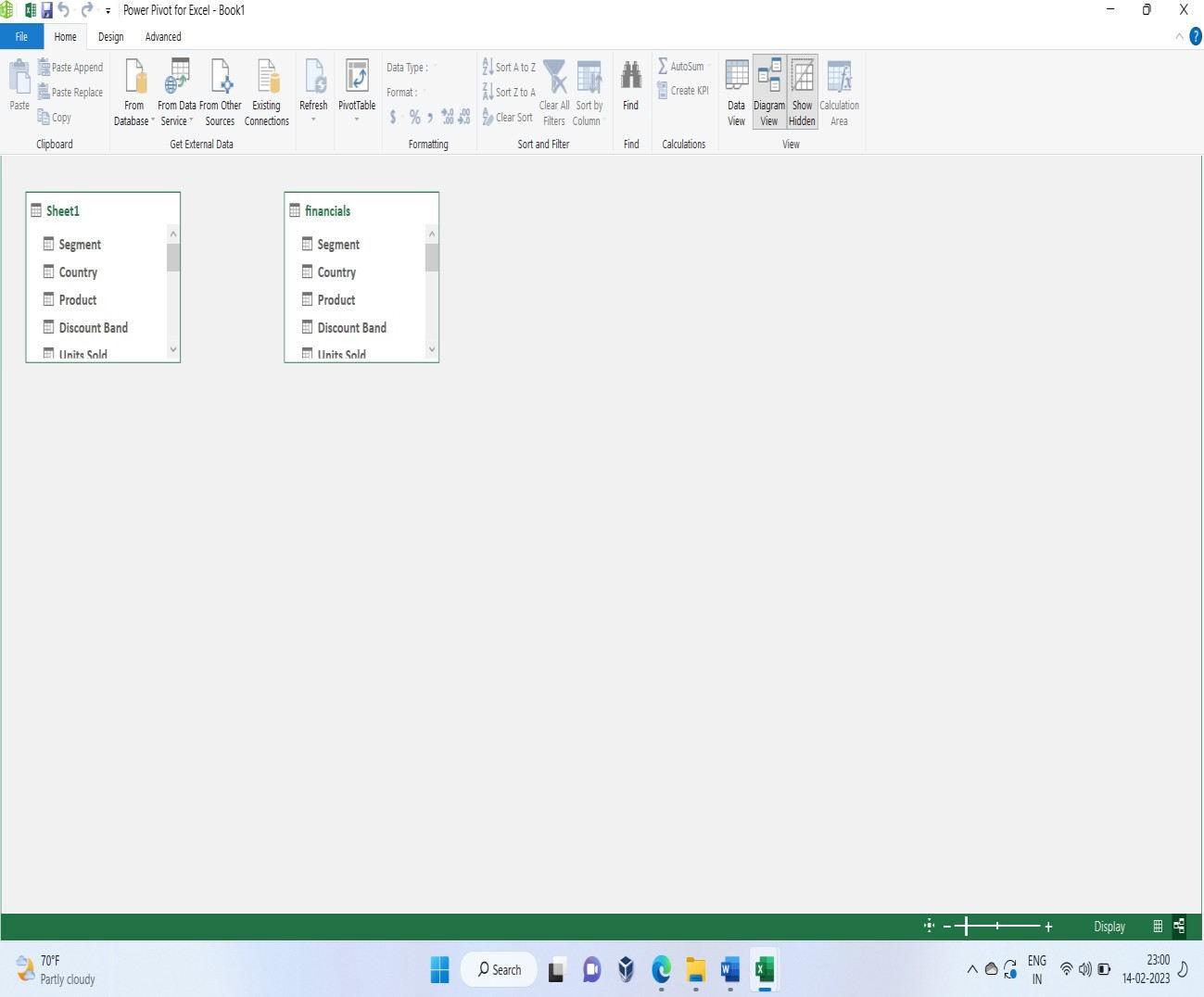


##### Next click manage data model and see that all the twitter data is loaded as model and close the power pivot window.

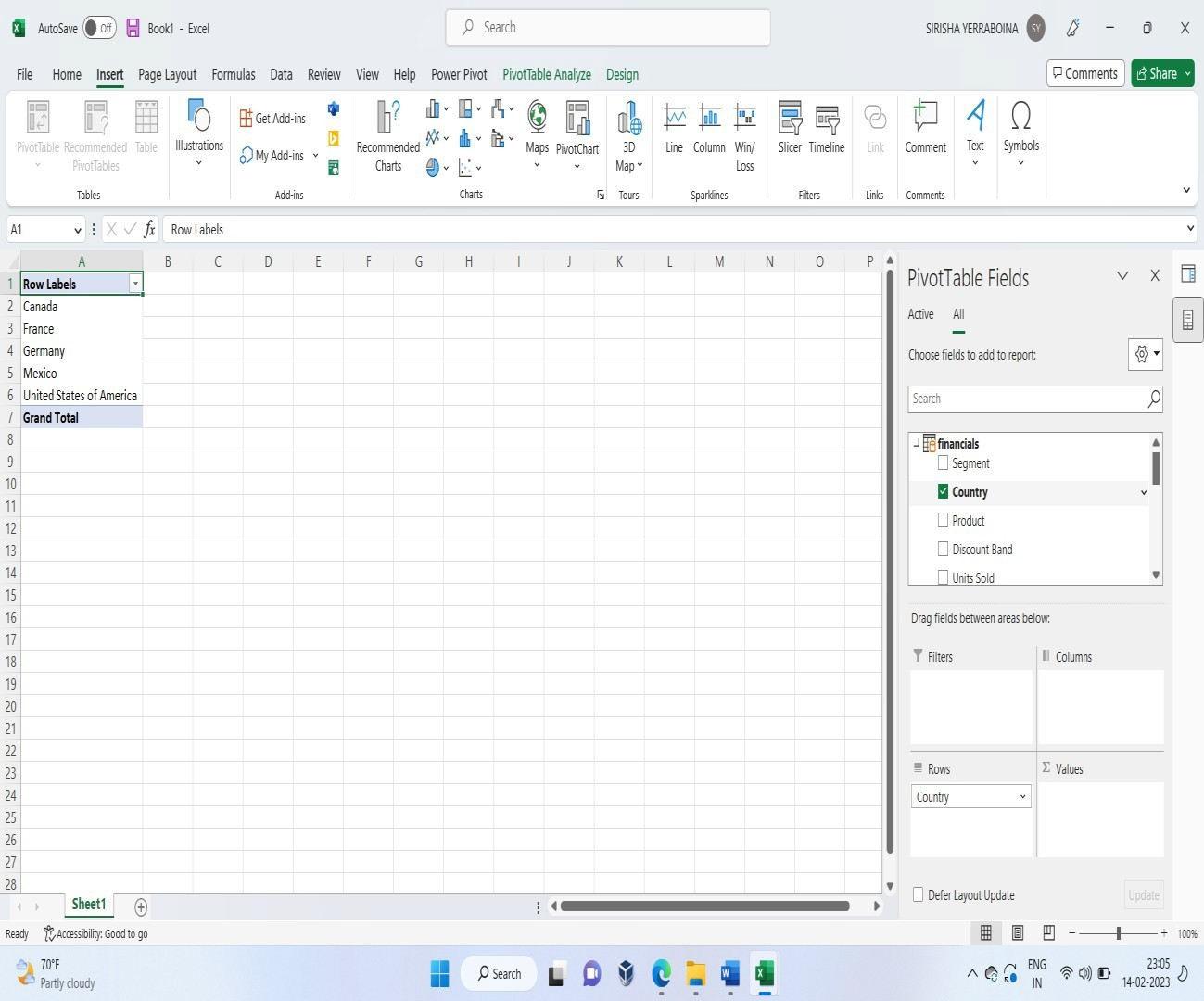
Save the excel 昀椀le as sample.xls



Click the diagram view and give the relationships between the tables



##### Go to the Insert menu and click pivot table



Select the columns and u can perform drill down and rollup operations using pivot table

We can load 10 millions rows of data also from multiple resources.

Experiment 8:Using Power Pivot perform the following on any data set B)Big data Charting

Aim :To create variety of charts using Excel for the given data Resources:Microsoft Excel

Theory:

When your data sets are big, you can use Excel Power Pivot that can handle hundreds of millions of rows of data. The data can be in external data sources and Excel Power Pivot builds a Data Model that works on a memory optimization mode. You can perform the calculations, analyze the data and arrive at a report to draw conclusions and decisions. The report can be either as a Power PivotTable or Power PivotChart or a combination of both.

You can utilize Power Pivot as an ad hoc reporting and analytics solution. Thus, it would be possible for a person with hands-on experience with Excel to perform the high-end data analysis and decision making in a matter of few minutes and are a great asset to be included in the dashboards.

### Uses of Power Pivot

You can use Power Pivot for the following −

* To perform powerful data analysis and create sophisticated Data Models.
* To mash-up large volumes of data from several di昀昀erent sources quickly.
* To perform information analysis and share the insights interactively.
* To create Key Performance Indicators (KPIs).
* To create Power PivotTables.
* To create Power PivotCharts.

### Di昀昀erences between PivotTable and Power PivotTable

Power PivotTable resembles PivotTable in its layout, with the following di昀昀erences −

* PivotTable is based on Excel tables, whereas Power PivotTable is based on data tables that are part of Data Model.
* PivotTable is based on a single Excel table or data range, whereas

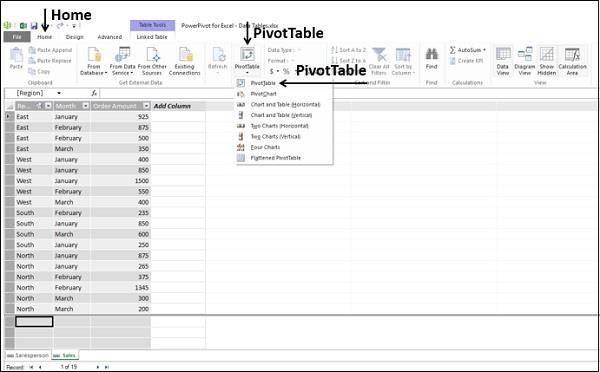
Power PivotTable can be based on multiple data tables, provided they are added to Data Model.

* PivotTable is created from Excel window, whereas Power PivotTable is created from PowerPivot window.

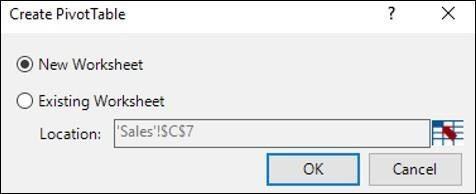
Creating a Power PivotTable

Suppose you have two data tables – Salesperson and Sales in the Data Model. To create a Power PivotTable from these two data tables, proceed as follows −

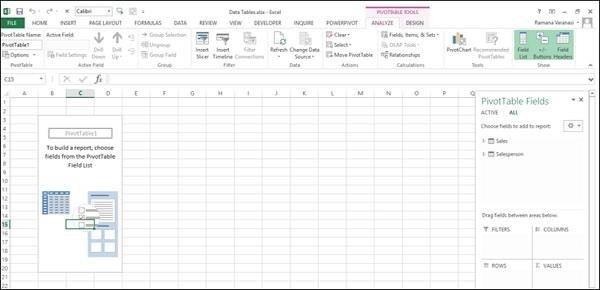
* Click on the Home tab on the Ribbon in PowerPivot window.
* Click on PivotTable on the Ribbon.
* Click on PivotTable in the dropdown list.



Create PivotTable dialog box appears. Click on New Worksheet.



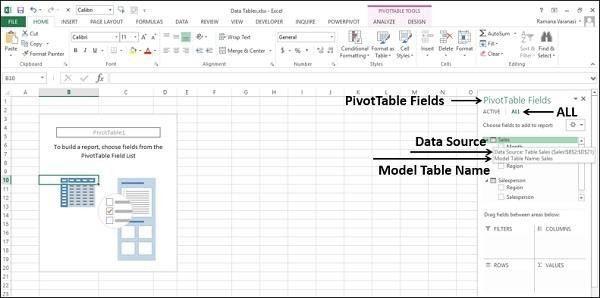
Click the OK button. New worksheet gets created in Excel window and an empty Power PivotTable appears.



As you can observe, the layout of the Power PivotTable is similar to that of PivotTable.

The PivotTable Fields List appears on the right side of the worksheet. Here, you will 昀椀nd some di昀昀erences from PivotTable. The Power PivotTable Fields list has two tabs − ACTIVE and ALL, that appear below the title and above the 昀椀elds list. ALL tab is highlighted. The ALL tab displays all the data tables in the Data Model and ACTIVE tab displays all the data tables that are chosen for the Power PivotTable at hand.

* Click the table names in the PivotTable Fields list under ALL. The corresponding 昀椀elds with check boxes will appear.
* Each table name will have the symbol on the left side.
* If you place the cursor on this symbol, the Data Source and the Model Table Name of that data table will be displayed.



* Drag Salesperson from Salesperson table to ROWS area.
* Click on the ACTIVE tab.

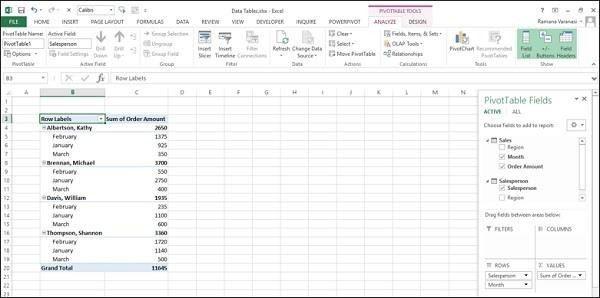


**75 |** P a g e

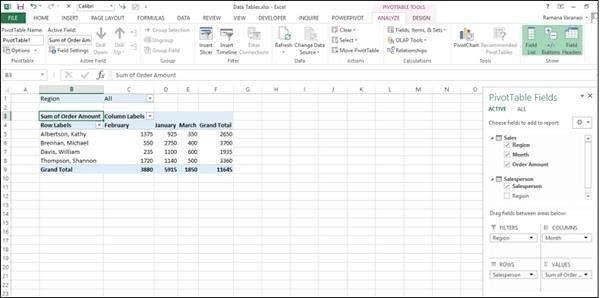
The 昀椀eld Salesperson appears in the Power PivotTable and the table Salesperson appears under ACTIVE tab.

* Click on the ALL tab.
* Click on Month and Order Amount in the Sales table.
* Click on the ACTIVE tab.

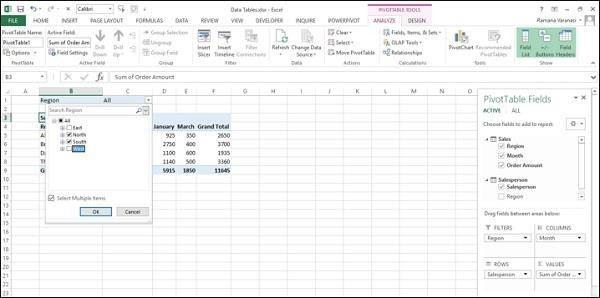
Both the tables – Sales and Salesperson appear under the ACTIVE tab.



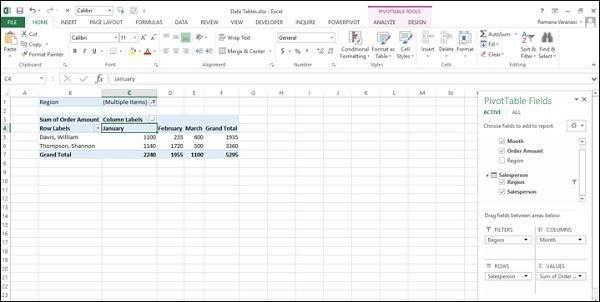
* Drag Month to COLUMNS area.
* Drag Region to FILTERS area.



* Click on arrow next to ALL in the Region 昀椀lter box.
* Click on Select Multiple Items.
* Click on North and South.



* Click the OK button. Sort the column labels in the ascending order.

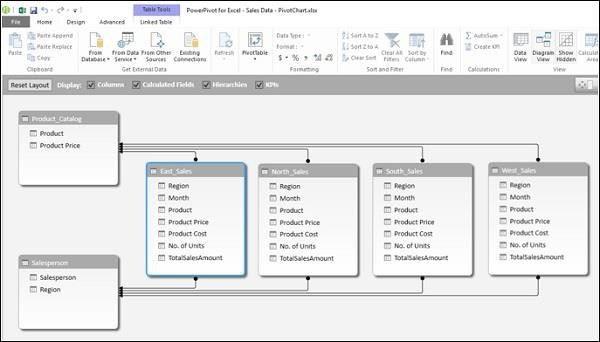


Power PivotTable can be modi昀椀ed dynamically to explore and report data.

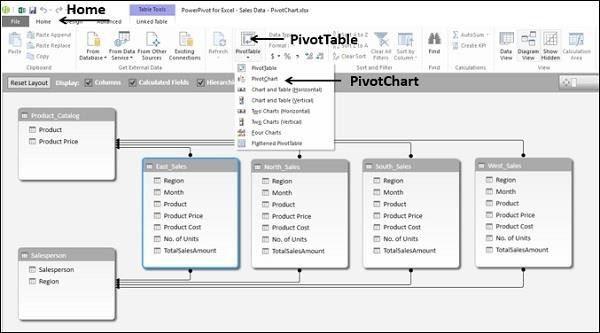
Creating a Power PivotChart

A Power PivotChart is a PivotChart that is based on Data Model and created from the Power Pivot window. Though it has some features similar to Excel PivotChart, there are other features that make it more powerful.

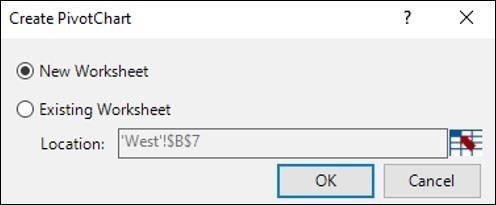
Suppose you want to create a Power PivotChart based on the following Data Model.



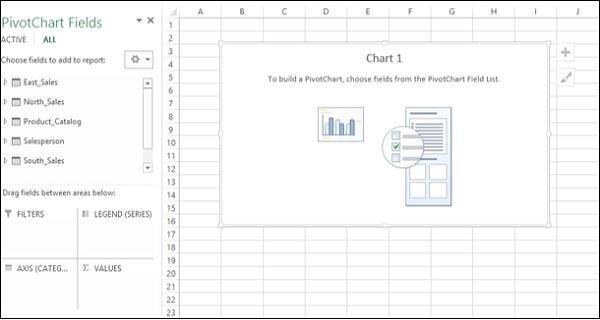
* Click on the Home tab on the Ribbon in the Power Pivot window.
* Click on PivotTable.
* Click on PivotChart in the dropdown list.



Create PivotChart dialog box appears. Click New Worksheet.



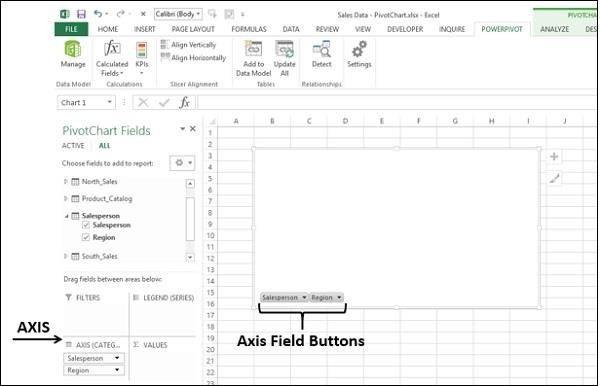
* Click the OK button. An empty PivotChart gets created on a new worksheet in the Excel window. In this chapter, when we say PivotChart, we are referring to Power PivotChart.



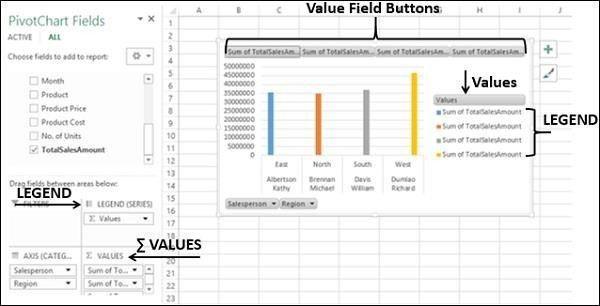
As you can observe, all the tables in the data model are displayed in the PivotChart Fields list.

* Click on the Salesperson table in the PivotChart Fields list.
* Drag the 昀椀elds – Salesperson and Region to AXIS area.

Two 昀椀eld buttons for the two selected 昀椀elds appear on the PivotChart. These are the Axis 昀椀eld buttons. The use of 昀椀eld buttons is to 昀椀lter data that is displayed on the PivotChart.



* Drag TotalSalesAmount from each of the 4 tables – East\_Sales, North\_Sales, South\_Sales and West\_Sales to ∑ VALUES area.

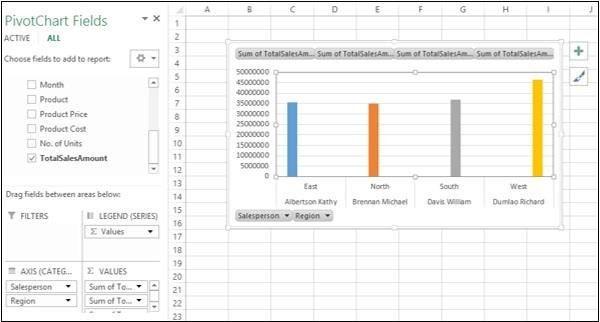


As you can observe, the following appear on the worksheet −

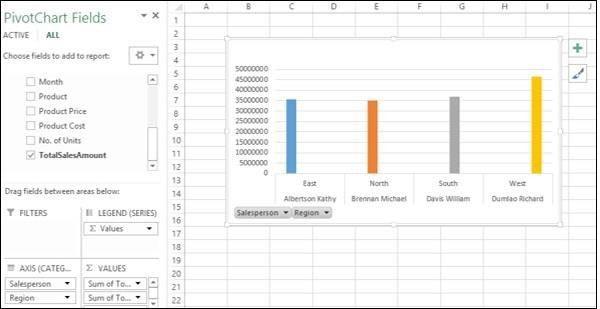
* In the PivotChart, column chart is displayed by default.
* In the LEGEND area, ∑ VALUES gets added.
* The Values appear in the Legend in the PivotChart, with title Values.
* The Value Field Buttons appear on the PivotChart.

You can remove the legend and the value 昀椀eld buttons for a tidier look of the PivotChart.

* Click on the button at the top right corner of the PivotChart.
* Deselect Legend in the Chart Elements.



* Right click on the value 昀椀eld buttons.
* Click on Hide Value Field Buttons on Chart in the dropdown list. The value 昀椀eld buttons on the chart will be



hidden.

Note that display of Field Buttons and/or Legend depends on the context of the PivotChart. You need to decide what is required to be displayed.

As in the case of Power PivotTable, Power PivotChart Fields list also contains two tabs − ACTIVE and ALL. Further, there are 4 areas −

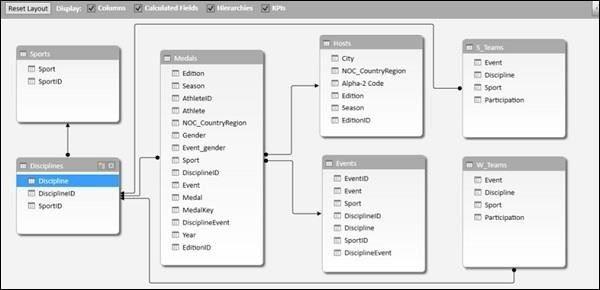
* AXIS (Categories)
* LEGEND (Series)
* ∑ VALUES
* FILTERS

As you can observe, Legend gets populated with ∑ Values. Further, Field Buttons get added to the PivotChart for the ease of 昀椀ltering the data that is being displayed. You can click on the arrow on a Field Button and select/deselect values to be displayed in the Power PivotChart.

Table and Chart Combinations

Power Pivot provides you with di昀昀erent combinations of Power PivotTable and Power PivotChart for data exploration, visualization and reporting.

Consider the following Data Model in Power Pivot that we will use for illustrations −



You can have the following Table and Chart Combinations in Power Pivot.

* Chart and Table (Horizontal) - you can create a Power PivotChart and a Power PivotTable, one next to another horizontally in the same worksheet.

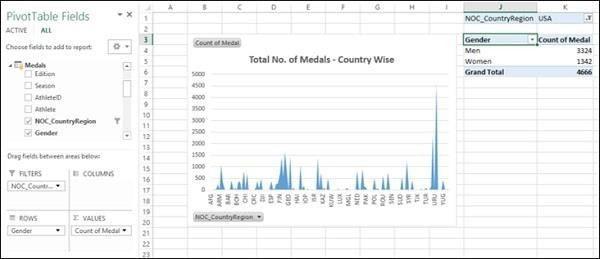
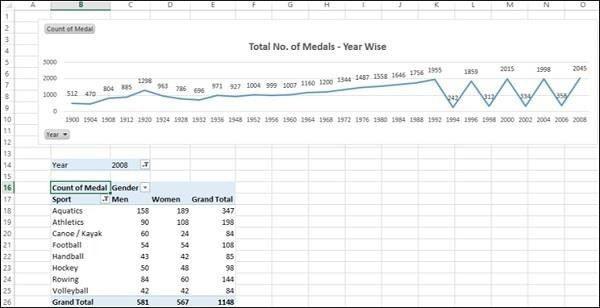
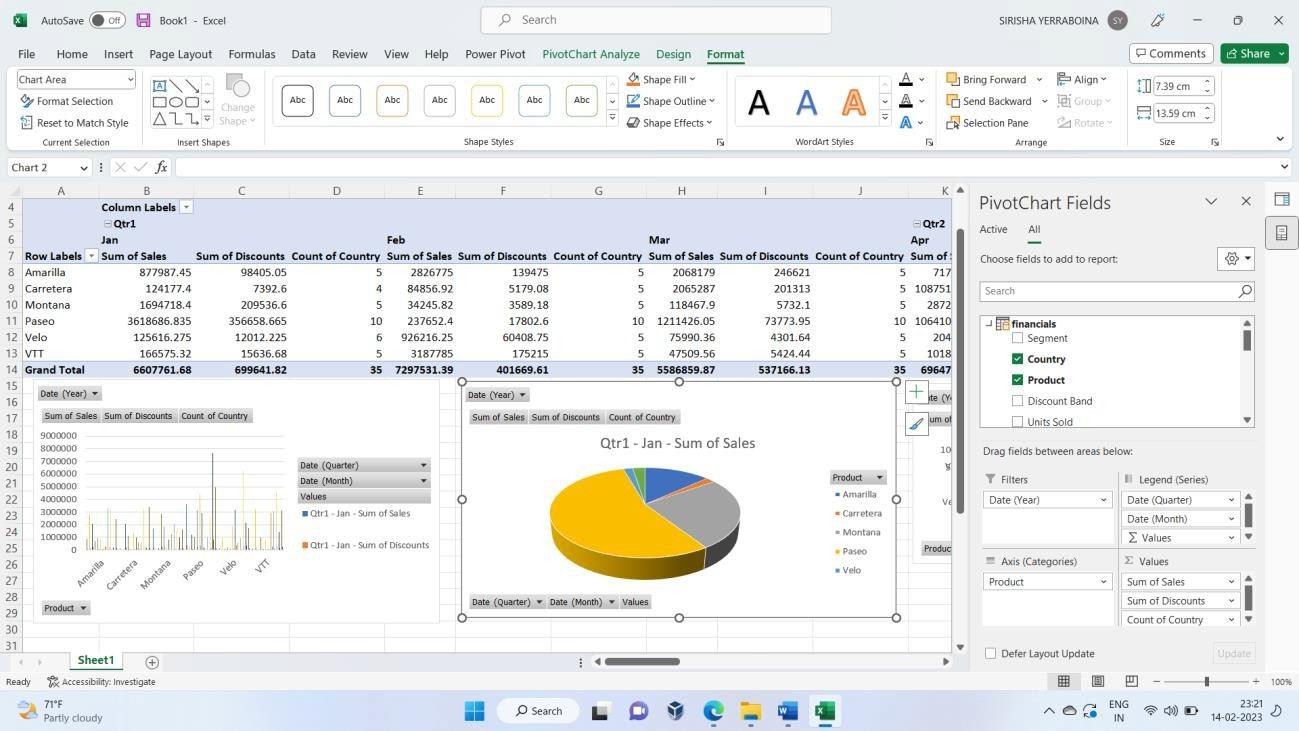


Chart and Table (Vertical) - you can create a Power PivotChart and a Power PivotTable, one below another vertically in the same worksheet.



These combinations and some more are available in the dropdown list that appears when you click on PivotTable on the Ribbon in the Power Pivot window.

Click on the pivot chart and can develop multiple variety of charts Output:



**Experiment 9 :using R project to carry out statistical analysis of big data**

**Aim:To perform the statistical analysis of big data using R**

**Theory:**Statistics is the science of analyzing, reviewing and conclude data.

Some basic statistical numbers include:

* Mean, median and mode
* Minimum and maximum value
* Percentiles
* Variance and Standard Devation
* Covariance and Correlation
* Probability distributions

The R language was developed by two statisticians. It has many built-in functionalities, in addition to libraries for the exact purpose of statistical analysis.

**Procedure:**

Installation of R and Rstudio

step 1:

sudo apt-get update

sudo apt-get install r-base step 2:

Installation of R studio

https://posit.co/download/rstudio-desktop/#download

step 1:download R studio for ubuntu

step 2 :wget -c https://download1.rstudio.org/desktop/jammy/amd64/rstudio-2022.07.2-576-amd64.deb

step 2:sudo dpkg -i rstudio-2022.07.2-576-amd64.deb

step 3 :sudo apt install -f step 4:rstudio launch R studio

**procedure:**

**-->install.packages("gapminder")**

**-->library(gapminder)**

**-->data(gapminder) output:**

**A tibble: 1,704 × 6**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | country  <fct> | continent  <fct> | year  <int> | lifeExp  <dbl> | pop  <int> | gdpPercap  <dbl> |
| 1 | Afghanistan | Asia | 1952 | 28.8 | 8425333 | 779. |
| 2 | Afghanistan | Asia | 1957 | 30.3 | 9240934 | 821. |
| 3 | Afghanistan | Asia | 1962 | 32.0 | 10267083 | 853. |
| 4 | Afghanistan | Asia | 1967 | 34.0 | 11537966 | 836. |
| 5 | Afghanistan | Asia | 1972 | 36.1 | 13079460 | 740. |
| 6 | Afghanistan | Asia | 1977 | 38.4 | 14880372 | 786. |
| 7 | Afghanistan | Asia | 1982 | 39.9 | 12881816 | 978. |
| 8 | Afghanistan | Asia | 1987 | 40.8 | 13867957 | 852. |
| 9 | Afghanistan | Asia | 1992 | 41.7 | 16317921 | 649. |
| 10 | Afghanistan | Asia | 1997 | 41.8 | 22227415 | 635. |

# … with 1,694 more rows

**-->summary(gapminder) summary(gapminder) output:**

country continent year Afghanistan: 12 Africa :624 Min. 1952 Albania : 12 Americas:300 1st Qu.:1966 Algeria : 12 Asia :396 Median :1980

Angola : 12 Europe :360 Mean 1980

Argentina : 12 Oceania : 24 3rd Qu.:1993

Australia : 12 Max. :2007

(Other) 1632

lifeExp pop gdpPercap

Min. :23.60 Min. :6.001e+04 Min. : 241.2 1st Qu.:48.20 1st Qu.:2.794e+06 1st Qu.: 1202.1

Median :60.71 Median :7.024e+06 Median : 3531.8 Mean :59.47 Mean :2.960e+07 Mean : 7215.3 3rd Qu.:70.85 3rd Qu.:1.959e+07 3rd Qu.: 9325.5 Max. :82.60 Max. :1.319e+09 Max. :113523.1

-->**x<-mean(gapminder$gdpPercap)**

**Type X to get mean value of gapminder**

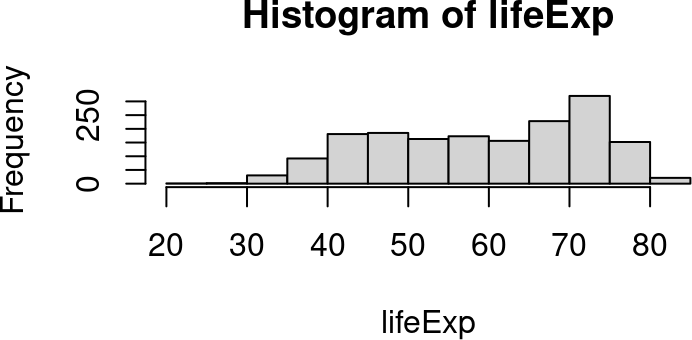
**-->x**

**output:[1] 7215.327**

**-->attach(gapminder)**

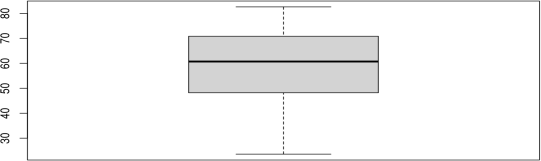
**-->median(pop) output:[1] 7023596**

**-->hist(lifeExp)**

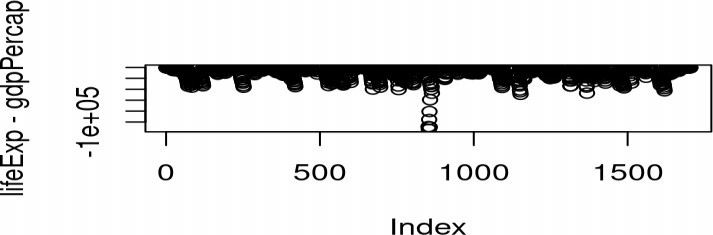


-->**boxplot(lifeExp)**

**will plot the below images**



**-->plot(lifeExp - gdpPercap)**



**-->install.packages("dplyr")**

**-->gapminder %>%**

**+ filter(year == 2007) %>%**

**+ group\_by(continent) %>%**

**+ summarise(lifeExp = median(lifeExp))**

**output:**

# A tibble: 5 × 2 continent lifeExp

<fct> <dbl>

1. Africa 52.9
2. Americas 72.9

3 Asia 72.4

1. Europe 78.6
2. Oceania 80.7

**-->install.packages("ggplot2")**

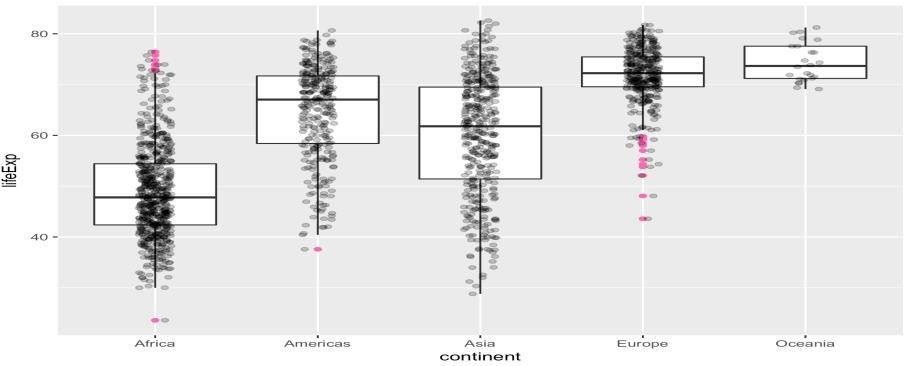
**--> library("ggplot2")**

**-->ggplot(gapminder, aes(x = continent, y = lifeExp))**

**+ geom\_boxplot(outlier.colour = "hotpink") +**

**geom\_jitter(position = position\_jitter(width = 0.1, height = 0), alpha = 1/4)**

**output:**



**-->head(country\_colors, 4) output:**

Nigeria Egypt Ethiopia "#7F3B08" "#833D07" "#873F07"

Congo, Dem. Rep. "#8B4107"

-->head(continent\_colors)

mtcars

mpg cyl disp hp drat wt qsec vs a

m

gear carb

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |
| Duster 360 | 14.3 | 8 | 360.0 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| Merc 450SE | 16.4 | 8 | 275.8 | 180 | 3.07 | 4.070 | 17.40 | 0 | 0 | 3 | 3 |
| Merc 450SL | 17.3 | 8 | 275.8 | 180 | 3.07 | 3.730 | 17.60 | 0 | 0 | 3 | 3 |
| Merc 450SLC | 15.2 | 8 | 275.8 | 180 | 3.07 | 3.780 | 18.00 | 0 | 0 | 3 | 3 |
| Cadillac Fleetwood | 10.4 | 8 | 472.0 | 205 | 2.93 | 5.250 | 17.98 | 0 | 0 | 3 | 4 |
| Lincoln Continental | 10.4 | 8 | 460.0 | 215 | 3.00 | 5.424 | 17.82 | 0 | 0 | 3 | 4 |
| Chrysler Imperial | 14.7 | 8 | 440.0 | 230 | 3.23 | 5.345 | 17.42 | 0 | 0 | 3 | 4 |
| Fiat 128 | 32.4 | 4 | 78.7 | 66 | 4.08 | 2.200 | 19.47 | 1 | 1 | 4 | 1 |
| Honda Civic | 30.4 | 4 | 75.7 | 52 | 4.93 | 1.615 | 18.52 | 1 | 1 | 4 | 2 |
| Toyota Corolla | 33.9 | 4 | 71.1 | 65 | 4.22 | 1.835 | 19.90 | 1 | 1 | 4 | 1 |
| Toyota Corona | 21.5 | 4 | 120.1 | 97 | 3.70 | 2.465 | 20.01 | 1 | 0 | 3 | 1 |
| Dodge Challenger | 15.5 | 8 | 318.0 | 150 | 2.76 | 3.520 | 16.87 | 0 | 0 | 3 | 2 |
| AMC Javelin | 15.2 | 8 | 304.0 | 150 | 3.15 | 3.435 | 17.30 | 0 | 0 | 3 | 2 |
| Camaro Z28 | 13.3 | 8 | 350.0 | 245 | 3.73 | 3.840 | 15.41 | 0 | 0 | 3 | 4 |
| Pontiac Firebird | 19.2 | 8 | 400.0 | 175 | 3.08 | 3.845 | 17.05 | 0 | 0 | 3 | 2 |
| Fiat X1-9 | 27.3 | 4 | 79.0 | 66 | 4.08 | 1.935 | 18.90 | 1 | 1 | 4 | 1 |
| Porsche 914-2 | 26.0 | 4 | 120.3 | 91 | 4.43 | 2.140 | 16.70 | 0 | 1 | 5 | 2 |
| Lotus Europa | 30.4 | 4 | 95.1 | 113 | 3.77 | 1.513 | 16.90 | 1 | 1 | 5 | 2 |
| Ford Pantera L | 15.8 | 8 | 351.0 | 264 | 4.22 | 3.170 | 14.50 | 0 | 1 | 5 | 4 |
| Ferrari Dino | 19.7 | 6 | 145.0 | 175 | 3.62 | 2.770 | 15.50 | 0 | 1 | 5 | 6 |
| Maserati Bora | 15.0 | 8 | 301.0 | 335 | 3.54 | 3.570 | 14.60 | 0 | 1 | 5 | 8 |
| Volvo 142E | 21.4 | 4 | 121.0 | 109 | 4.11 | 2.780 | 18.60 | 1 | 1 | 4 | 2 |

> Data\_Cars <- mtcars

> dim(Data\_Cars) [1] 32 11

> names(Data\_Cars)

[1] "mpg" "cyl" "disp" "hp" "drat" "wt" "qsec" "vs" "am" "gear" "car b"

> Data\_Cars <- mtcars

> Data\_Cars$cyl

[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4

> Data\_Cars <- mtcars

> sort(Data\_Cars$cyl)

[1] 4 4 4 4 4 4 4 4 4 4 4 6 6 6 6 6 6 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8

> Data\_Cars <- mtcars

>

> summary(Data\_Cars)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mpg  Min. :10.40 | cyl  Min. :4.000 | disp  Min. : 71.1 | hp  Min. : 52.0 | drat  Min. :2.760 |
| 1st Qu.:15.43 1st Qu.:4.000 1st Qu.:120.8 1st Qu.: 96.5 1st Qu.:3.080  Median :19.20 Median :6.000 Median :196.3 Median :123.0 Median :3.695  Mean :20.09 Mean :6.188 Mean :230.7 Mean :146.7 Mean :3.597 3rd Qu.:22.80 3rd Qu.:8.000 3rd Qu.:326.0 3rd Qu.:180.0 3rd Qu.:3.920 Max. :33.90 Max. :8.000 Max. :472.0 Max. :335.0 Max. :4.930  wt qsec vs a gear m  Min. :1.513 Min. :14.50 Min. :0.0000 Min. :0.0000 Min. :3.00  0 | | | | |
| 1st Qu.:2.581 1st Qu.:16.89 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:3.00  0 | | | | |
| Median :3.325 | Median :17.71 | Median :0.0000 | Median :0.0000 | Median :4.00 |
| Mean :3.217 | Mean :17.85 | Mean :0.4375 | Mean :0.4062 | Mean :3.68 |
| 3rd Qu.:3.610 | 3rd Qu.:18.90 | 3rd Qu.:1.0000 | 3rd Qu.:1.0000 | 3rd Qu.:4.00 |
| Max. :5.424 | Max. :22.90 | Max. :1.0000 | Max. :1.0000 | Max. :5.00 |

0

8

0

0

carb

Min. :1.000 1st Qu.:2.000

Median :2.000

Mean :2.812 3rd Qu.:4.000 Max. :8.000

> Data\_Cars <- mtcars

>

> max(Data\_Cars$hp) [1] 335

> min(Data\_Cars$hp) [1] 52

> Data\_Cars <- mtcars

>

> which.max(Data\_Cars$hp) [1] 31

> which.min(Data\_Cars$hp) [1] 19

> Data\_Cars <- mtcars

> rownames(Data\_Cars)[which.max(Data\_Cars$hp)] [1] "Maserati Bora"

> rownames(Data\_Cars)[which.min(Data\_Cars$hp)] [1] "Honda Civic"

> median(Data\_Cars$wt) [1] 3.325

> names(sort(-table(Data\_Cars$wt)))[1] [1]

"3.44"

> Data\_Cars <- mtcars

>

> mean(Data\_Cars$wt) [1] 3.21725

Data\_Cars <- mtcars median(Data\_Cars$wt)



Data\_Cars <- mtcars

names(sort(-table(Data\_Cars$wt)))[1] Data\_Cars <- mtcars

# c() specifies which percentile you want quantile(Data\_Cars$wt, c(0.75)) 75%

3.61

Data\_Cars <- mtcars

>

> quantile(Data\_Cars$wt)

0% 25% 50% 75% 100%

1.51300 2.58125 3.32500 3.61000 5.42400

**Regression analysis using R**

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

The general mathematical equation for a linear regression is −



Following is the description of the parameters used −

* + **y** is the response variable.
  + **x** is the predictor variable.
  + **a** and **b** are constants which are called the coefficients.

Steps to Establish a Regression

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is −

* Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
* Create a relationship model using the **lm()** functions in R.
* Find the coefficients from the model created and create the mathematical equation using these
* Get a summary of the relationship model to know the average error in predic- tion. Also called **residuals**.
* To predict the weight of new persons, use the **predict()** function in R.

**Input Data**

Below is the sample data representing the observations −



##### lm() Function

This function creates the relationship model between the predictor and the response vari- able.

**Syntax**

The basic syntax for **lm()** function in linear regression is −



Following is the description of the parameters used −

* **formula** is a symbol presenting the relation between x and y.
* **data** is the vector on which the formula will be applied.

**Create Relationship Model & get the Coefficient**



**Result:**



**To get the summary of the relation ships**



predict() Function

**Syntax**

The basic syntax for predict() in linear regression is −



Following is the description of the parameters used −

* **object** is the formula which is already created using the lm() function.
* **newdata** is the vector containing the new value for predictor variable.

**Predict the weight of new persons**

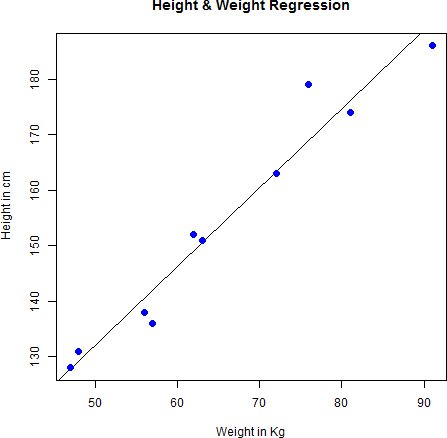


**Result:**



**Visualize the Regression Graphically**





Experiment 10: Using R project for data visualization of social media Aim:To perform data visualization using R programming

Theory:

**Data visualization** is the technique used to deliver insights in data using visual cues such as graphs, charts, maps, and many others. This is useful as it helps in intuitive and easy understanding of the large quantities of data and thereby make better decisions regarding it.

**Data Visualization in R Programming Language**

The popular data visualization tools that are available are Tableau, Plotly, R, Google Charts, Infogram, and Kibana. The various data visualization platforms have different capabilities, functionality, and use cases. They also require a different skill set. This article discusses the use of R for data visualization.

R is a language that is designed for statistical computing, graphical data analysis, and scientific research. It is usually preferred for data visualization as it offers flexibility and minimum required coding through its packages.

**Types of Data Visualizations**

Some of the various types of visualizations offered by R are: Bar Plot

There are two types of bar plots- horizontal and vertical which represent data points as horizontal or vertical bars of certain lengths proportional to the value of the data item. They are generally used for continuous and categorical variable plotting. By setting the **horiz** parameter to true and false, we can get horizontal and vertical bar plots respectively.

Bar plots are used for the following scenarios:

* To perform a comparative study between the various data categories in the data set.
* To analyze the change of a variable over time in months or years.

**Histogram**

A histogram is like a bar chart as it uses bars of varying height to represent data distribution. However, in a histogram values are grouped into consecutive intervals called bins. In a Histogram, continuous values are grouped and displayed in these bins whose size can be varied.

For a histogram, the parameter **xlim** can be used to specify the interval within which all values are to be displayed. Another parameter **freq** when set to *TRUE* denotes the frequency of the various values in the histogram and when set to *FALSE*, the probability densities are represented on the y-axis such that they are of the histogram adds up to one.

**Histograms are used in the following scenarios:**

* To verify an equal and symmetric distribution of the data.
* To identify deviations from expected values.

Box Plot

The statistical summary of the given data is presented graphically using a boxplot. A boxplot depicts information like the minimum and maximum data point, the median value, first and third quartile, and interquartile range.

**Box Plots are used for:**

* To give a comprehensive statistical description of the data through a visual cue.
* To identify the outlier points that do not lie in the inter-quartile range of data.

**Scatter Plot**

A scatter plot is composed of many points on a Cartesian plane. Each point denotes the value taken by two parameters and helps us easily identify the relationship between them.

**Scatter Plots are used in the following scenarios:**

* To show whether an association exists between bivariate data.
* To measure the strength and direction of such a relationship.

**Heat Map**

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. heatmap() function is used to plot heatmap.

***Syntax:*** *heatmap(data)*

***Parameters:*** *data: It represent matrix data, such as values of rows and columns*

***Return:*** *This function draws a heatmap.*

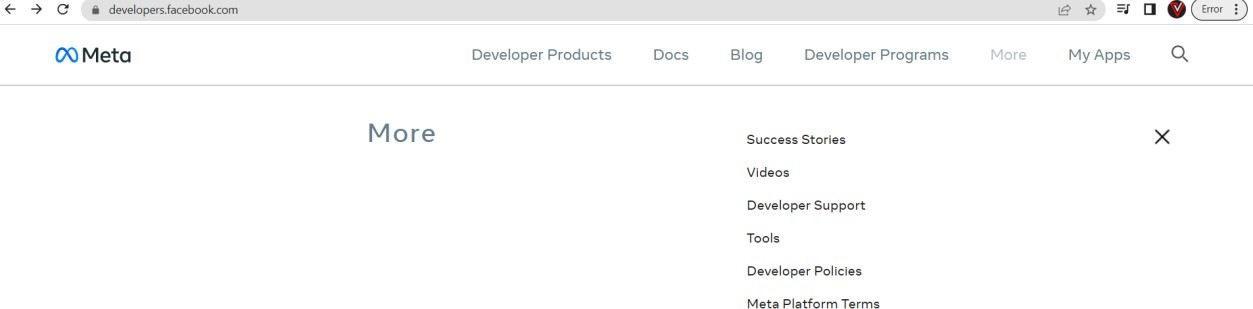
## Procedure:

**Step I : Facebook Developer Registration**

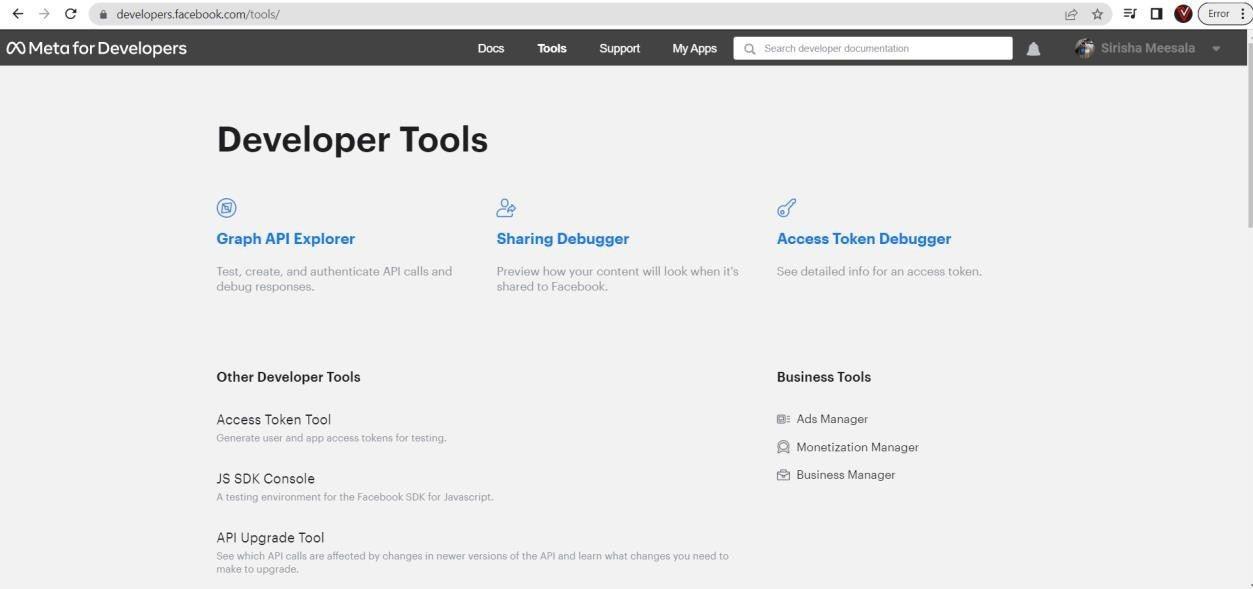
Go to https://developers.facebook.com and register yourself by clicking on **Get Started** button at the top right of page (See the snapshot below). After it would open a form for registration which you need to 昀椀ll it to get yourself registered.



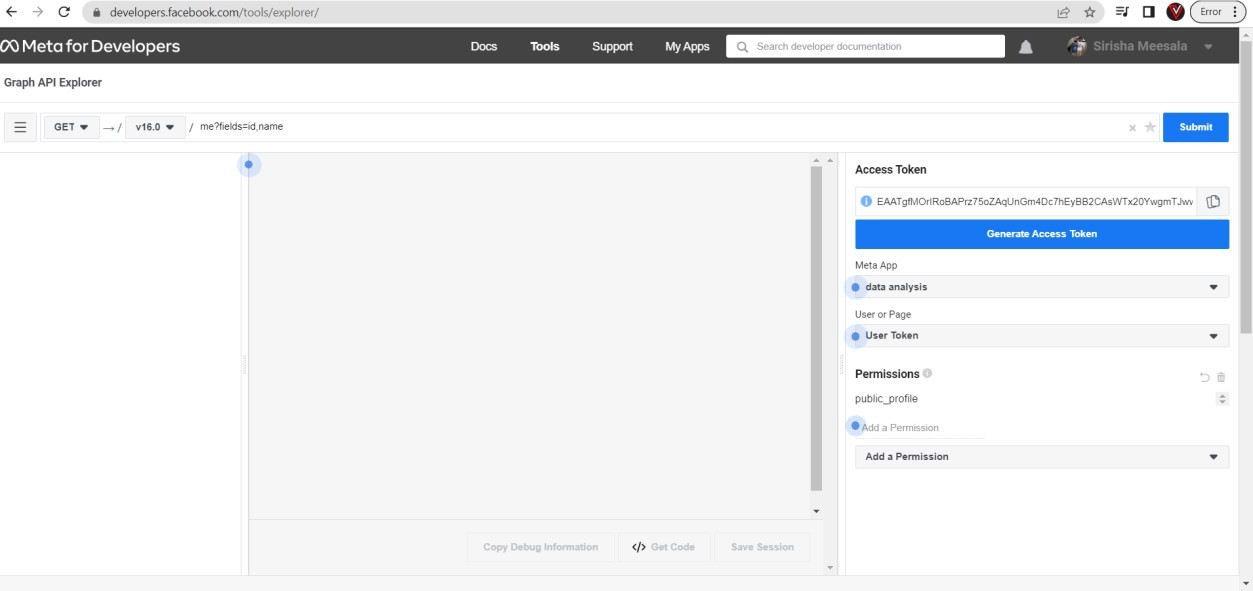
**Step2:click on tools**



**Step3 :click on graphApi explorer**



**Step4:copy the access token**



Copy the access token

Go to R studio and write this Script

install.packages("h琀琀puv") install.packages("Rfacebook") install.packages("RcolorBrewer") install.packages("Rcurl") install.packages("rjson") install.packages("h琀琀r")

library(Rfacebook) library(h琀琀puv) library(RcolorBrewer)

acess\_token="EAATgfMOrIRoBAOR9XUl3VGzbLMuWGb9FqGkTK3PFBuRyUVZA WAL7ZBw0xN3AijCsPiZBylucovck4YUhU昀欀WLMZBo640k2ZAupKgsaKog9736lec P8E52qkl5de8M963oKG8KOCVUXqqLiRcI7yIbEONeQt0eyLI6LdoeZA65Hyxf8so1 UMbywAdZCZAQBpNiZAPPj7G3UX5jZAvUpRLZCQ5SIG"

op琀椀ons(RCurlop琀椀ons=list(verbose=FALSE,capath=system.昀椀le("CurlSSL","cacert. pem",package = "Rcurl"),ssl.verifypeer=FALSE))

me<-getUsers("me",token=acess\_token) View(me)

myFriends<-getFriends(acess\_token,simplify = FALSE) table(myFriends)

pie(table(myFriends$gender))

output

