Practical No.: 1

# Aim: Install, configure and run Hadoop and HDFS ad explore HDFS.

**Step 1: Download and install VirtualBox**

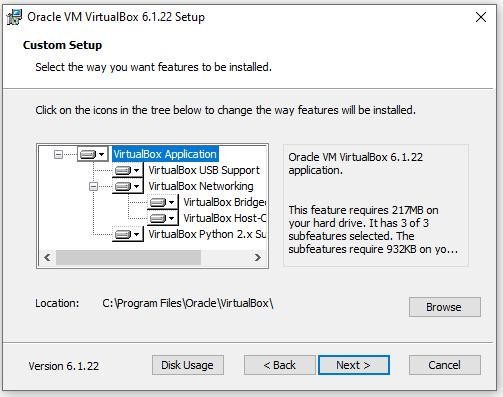
Go to the website of Oracle VirtualBox and get the latest stable version from the following site

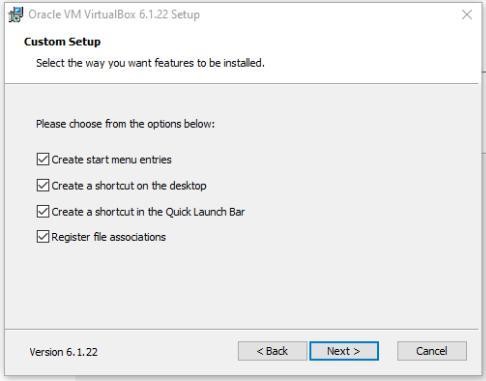
https://[www.virtualbox.org/](http://www.virtualbox.org/)

click on ‘Download’’

You will get VirtualBox-6.1.22-144080-Win.exe file downloaded. Double click and run it. Click on next.

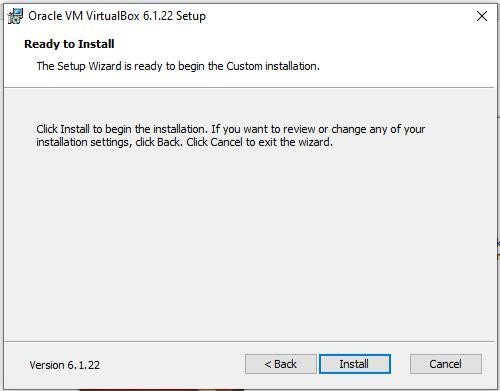


Click on ‘next’ without changing the default folder as shown below

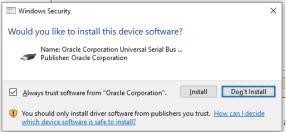
Again, click on next as shown below:

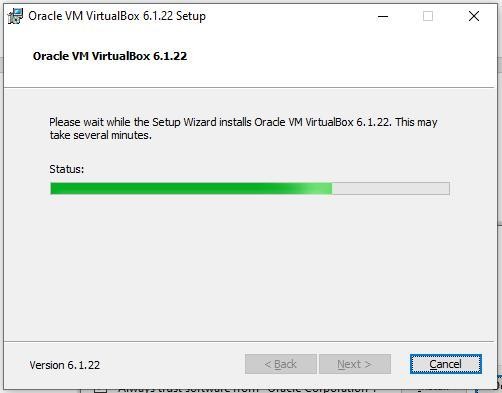
Finally, click on ‘Yes’.



Click on ’Install’.

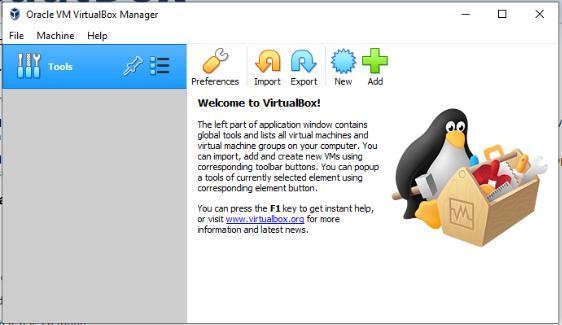
It may ask you for the permission to install, click ’yes’ to allow. Select ‘Install’ as shown below:



You will get the screen as shown below:

Click on ‘Finish’ to finish Installation of virtual box.



You will get the following screen:

# Step 2: download Ubuntu

Download iso file ubuntu-20.04.2.0-desktop-amd64; which is required to install Ubuntu.

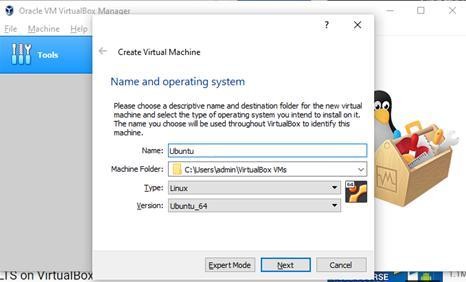
Browse ubuntu.com

Click on download and 20.04 LTS as shown below: LTS stands for Long term support

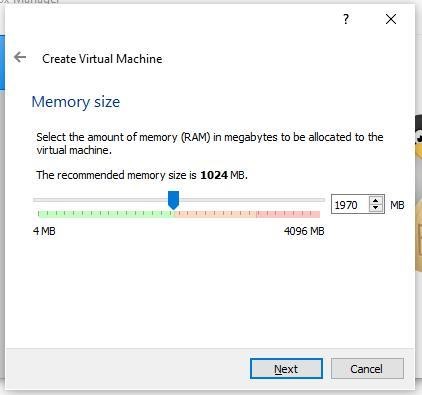


You will get file, which may take few minutes to download.

Now, click on ‘New’ to virtual box and write Name as ‘Ubuntu’ as shown below:

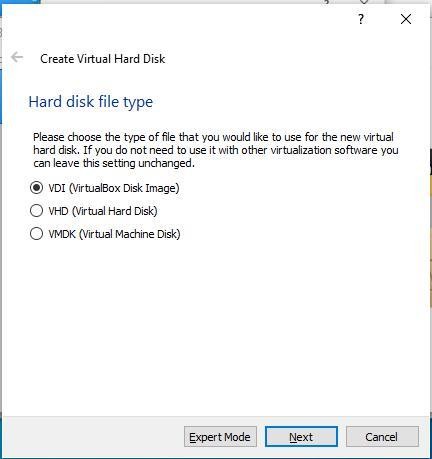


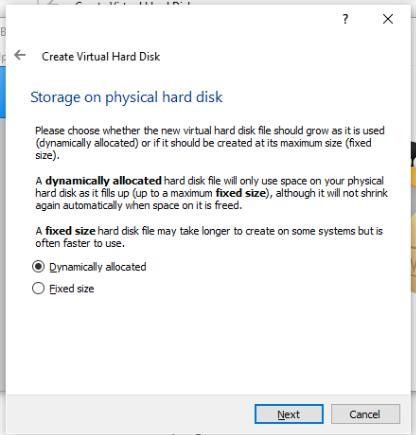
Click on ‘Next’.

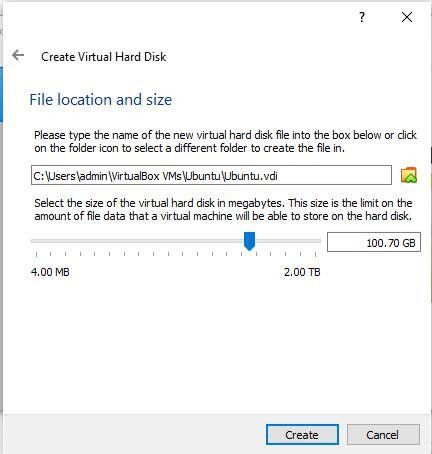


Here, you allow memory size up to green indicator (1970 MB). Click on ‘Next’.



Don’t change anything in this screen and click on ‘Create’.

Click on ‘Next’, keeping the selection as it is (on VDI).’

Keep this screen also as it is and click on ‘Next’.

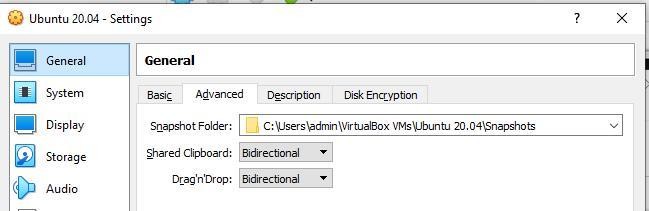
Keep the file location as it is but preferably keep size 100 GB and click on ‘Create’.

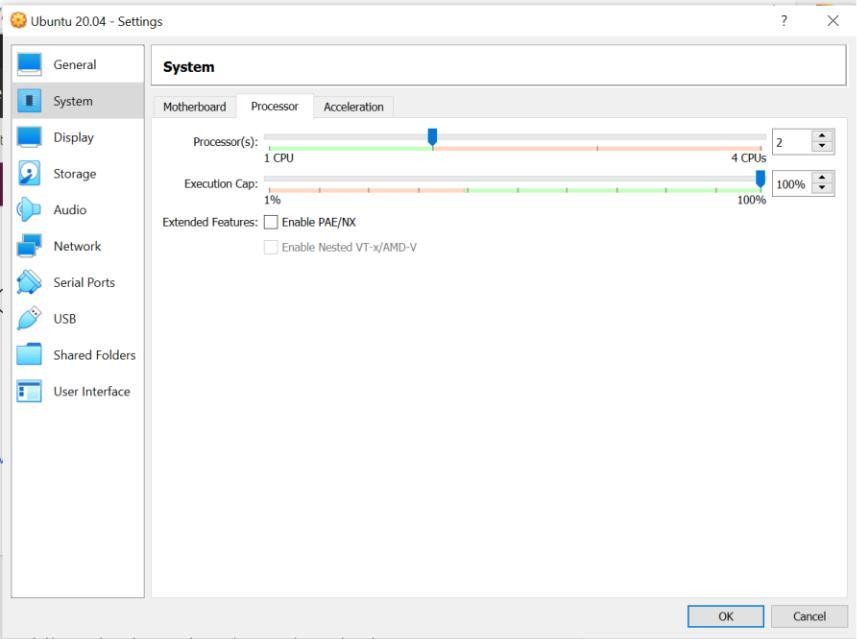
You may see the following screen having Ubuntu on Virtual Machine.

Select ‘General’ -> ’ Basic’ as shown below:

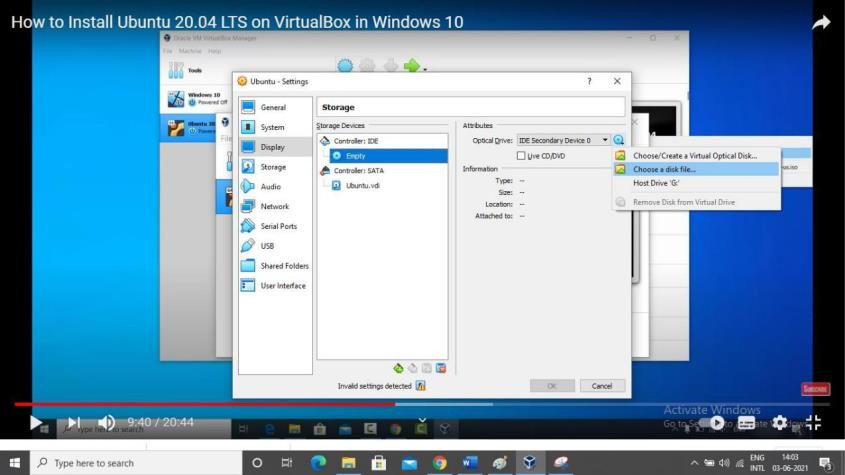
You may change the name from Ubuntu to Ubuntu 20.04

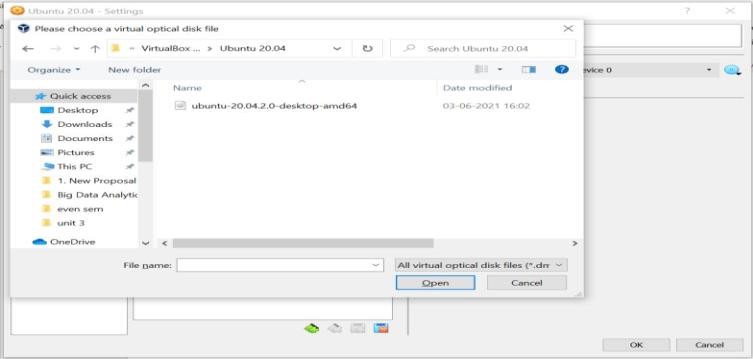
Click on Ubuntu and then click on settings option as shown below: Select bidirectional in ‘General’ -> ’ Advanced’ as shown below:



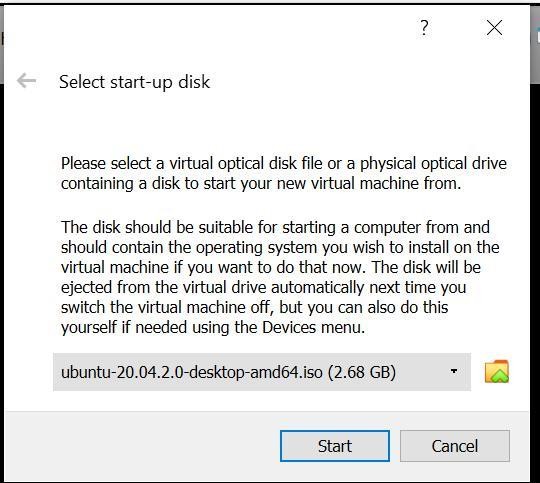
Go to ‘System’ option and change the processor up to green bar, usually 4.(if it allows)

Cut and paste your ubuntu .iso file from current folder to C:\Users\ADMIN\VirtualBox VMs\Ubuntu 20.04 folder.

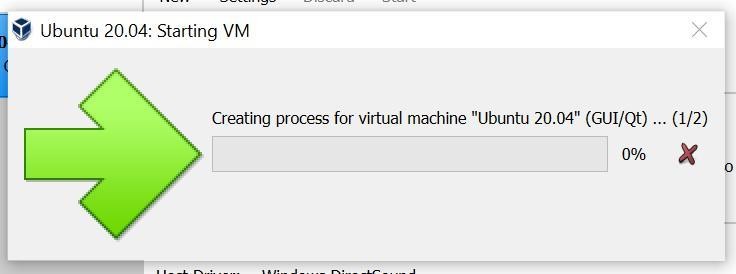
Click on ‘Storage’ and click on ‘Empty’ followed by ‘Choose a disk file’ as shown below:

Browse the folder where you have selected ubuntu iso file.

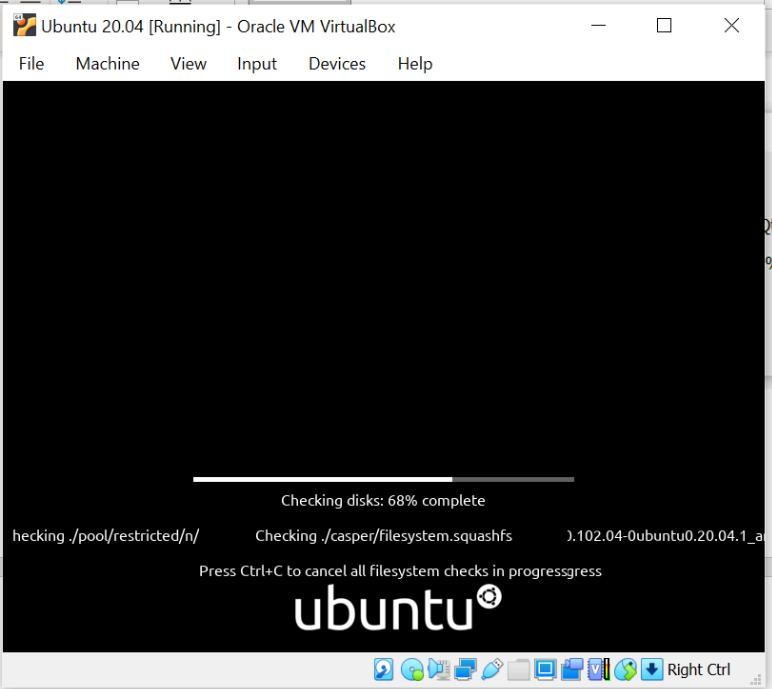
Click on Ubuntu….iso file and click on open and then click on ok. Click on Ubuntu -> start button.



Again, click on ‘Start’ button. It will show you the following screen.



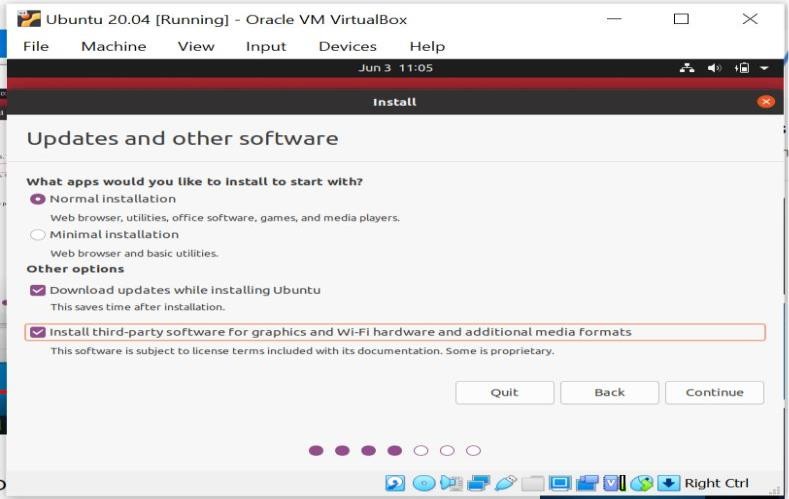
And simultaneously one more screen as follows:



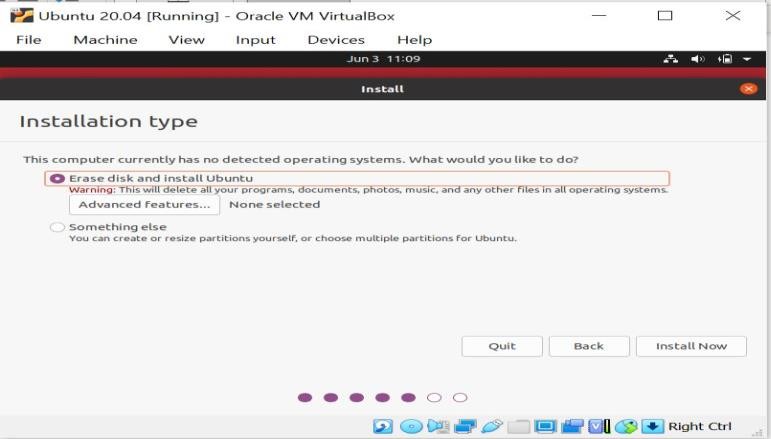
Keep on closing all warnings.

Next you will get following screen automatically.

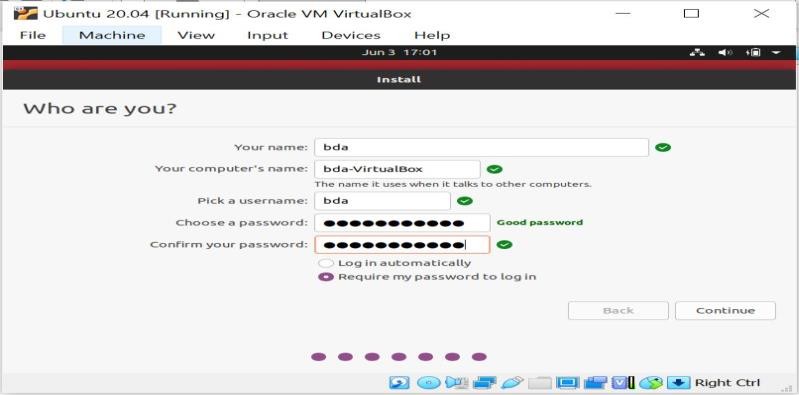
Select language -> English and click on ‘Install Ubuntu’.in ‘Keyboard Layout’ screen, select ‘English UK’. Click on ‘Continue’.

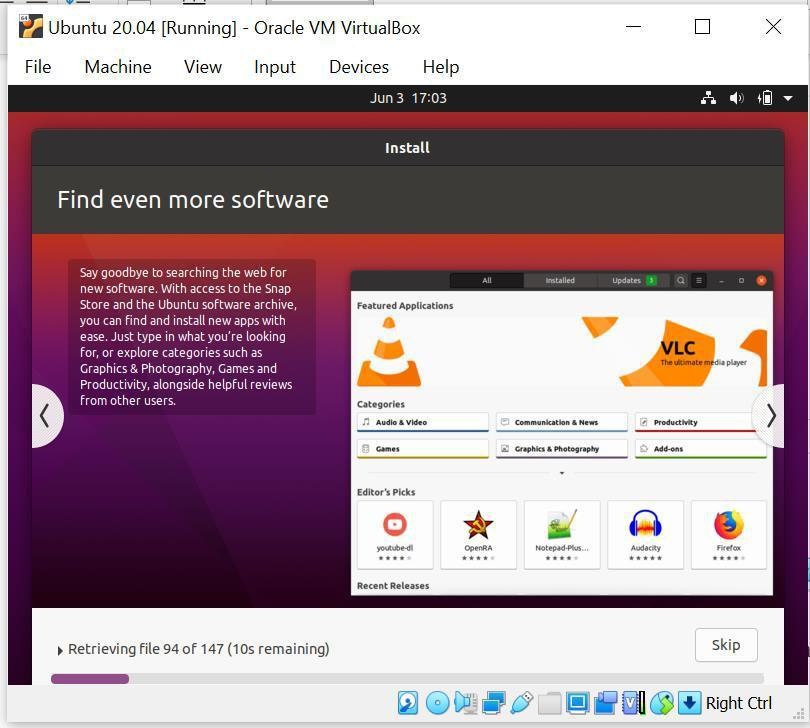
Select the checkbox for third party software as shown below:

Click on ‘continue’.



Select Erase disk and Install Ubuntu and click on **‘Install Now’**. Click on **‘Continue’** on the next screen.

Select “Kolkata” for “where are you?” and click on **‘Continue’.**

Click on continue after entering name, company name, username, password and confirm your password.

Installation of Ubuntu started. Click on finish once installation done. Click on restart and press Enter key.

# Step 3 Install Hadoop

Login to ubuntu

Some keys may change like you try to type @ and it types “.

\*\* please refer to note - Some Keys for Ubuntu under UK keyboard layout – at the end.

\*\* Search for Ubuntu terminal on search bar, after login done.

\*\* Apply following commands from ubuntu terminal

$ sudo apt update

$ sudo apt install default-jdk

$ ava -version'

$ wget [https://hadoop.apache.org/release/3.2.2.html/hadoop-](https://hadoop.apache.org/release/3.2.2.html/hadoop-3.2.2.tar.gz) [3.2.2.tar.gz](https://hadoop.apache.org/release/3.2.2.html/hadoop-3.2.2.tar.gz) $ tar xzvf hadoop-3.2.2.tar.gz

$ sudo mv hadoop-3.2.2 /usr/local/hadoop

$ readlink -f /usr/bin/java | sed "s:bin/java::"

# : Configuring Hadoop’s Java Home; To begin, open hadoop-env.sh $ sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

File will be opened. Add the following line at the end of .sh file export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64/

to save the changes in the file, press ctrl and x together. then press Y

then press Enter key

then apply following commands:

$ /usr/lib/jvm/java-11-openjdk-amd64/

## Step 4: Running Hadoop

$ /usr/local/hadoop/bin/hadoop

$ mkdir ~/input

$ cp /usr/local/hadoop/etc/hadoop/\*.xml ~/input

We can use the following command to run the MapReduce hadoop mapreduce-examples program, a Java archive with several options.

We’ll invoke its grep program, one of the many examples included in hadoop-mapreduce- examples, followed by the input directory, input and the output directory grep\_example. The MapReduce grep program will count the matches of a literal word or regular expression.

Finally, we’ll supply the regular expression allowed\*.+\* to find occurrences of the word allowed within or at the end of a declarative sentence.

he expression is case-sensitive, so we wouldn’t find the word if it were capitalized at the beginning of a sentence:

$ /usr/local/hadoop/bin/hadoop jar

/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep ~/input ~/grep\_example 'allowed[.]\*' $ cat ~/grep\_example/\*

## \*\*Note:

**Some Keys for Ubuntu under UK keyboard layout**

“->@ @-

>“

pipe -> take from this file or on google search for pipe in linux ~ -

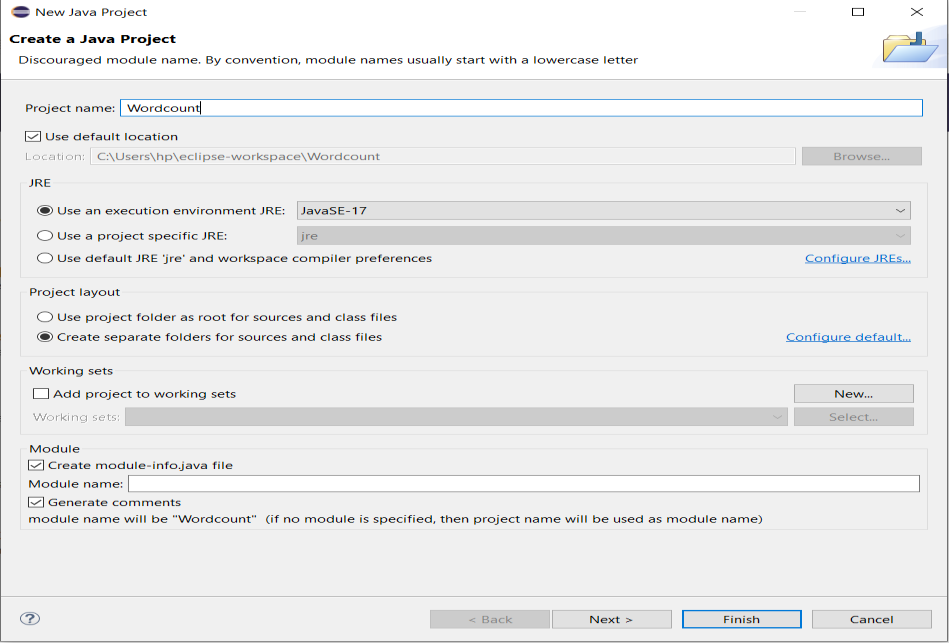
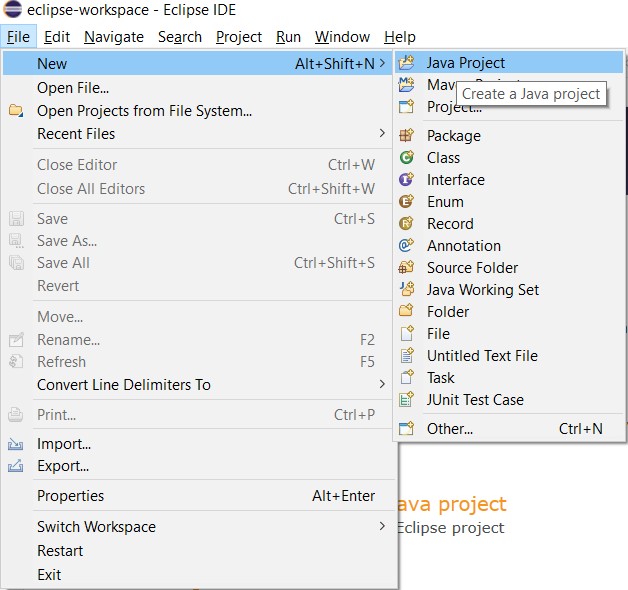
> pipe

**Practical No: 02**

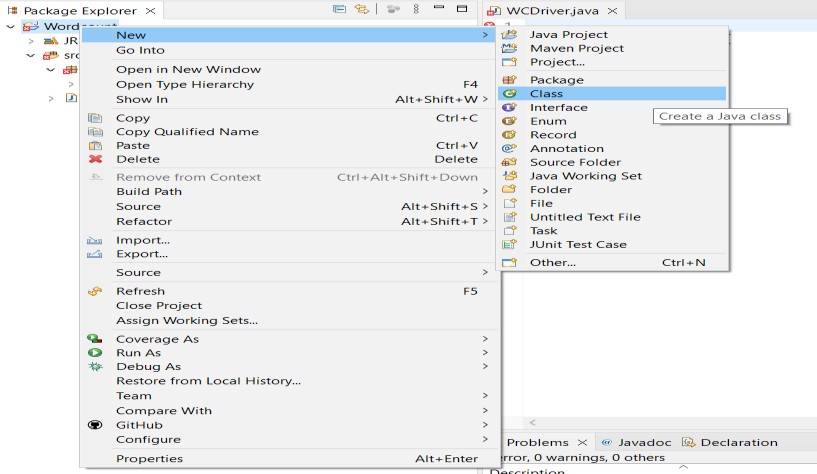
**Aim: Implement word count Program using MapReduce**.

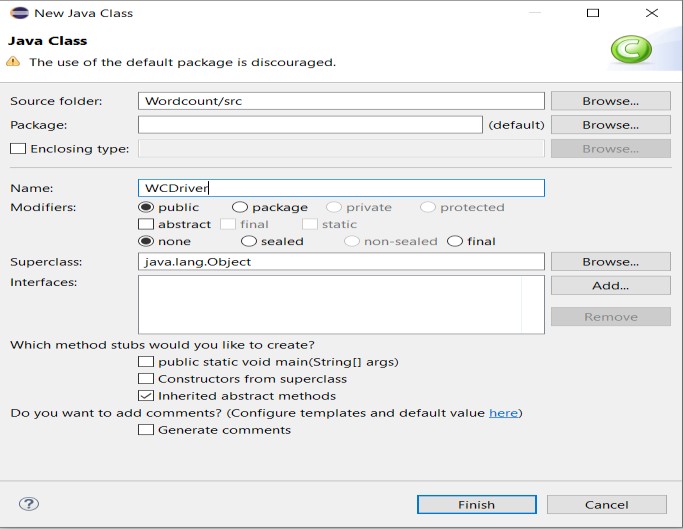
**Step 1:** First Open **Eclipse** -> then select **File** -> **New** -> **Java Project** ->Name it

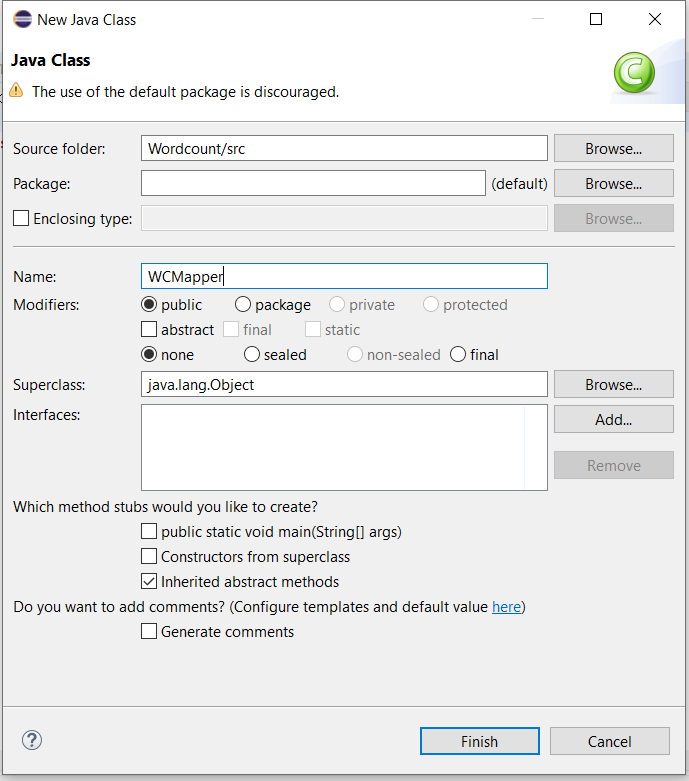
**WordCount** -> then **Finish**.

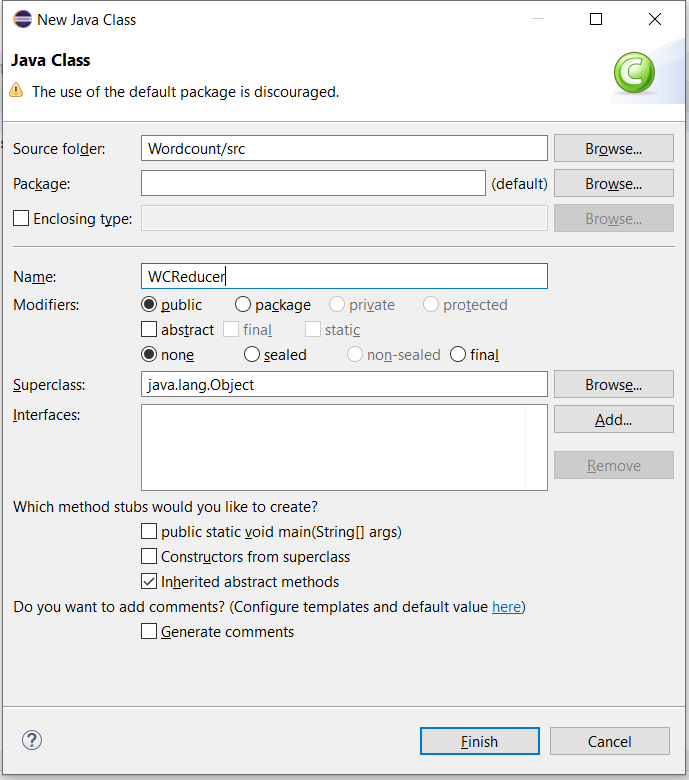


**Step 2:** Create 3 Java Classes into the project. Name them **WCDriver**(having the main function), **WCMapper**, **WCReducer**.

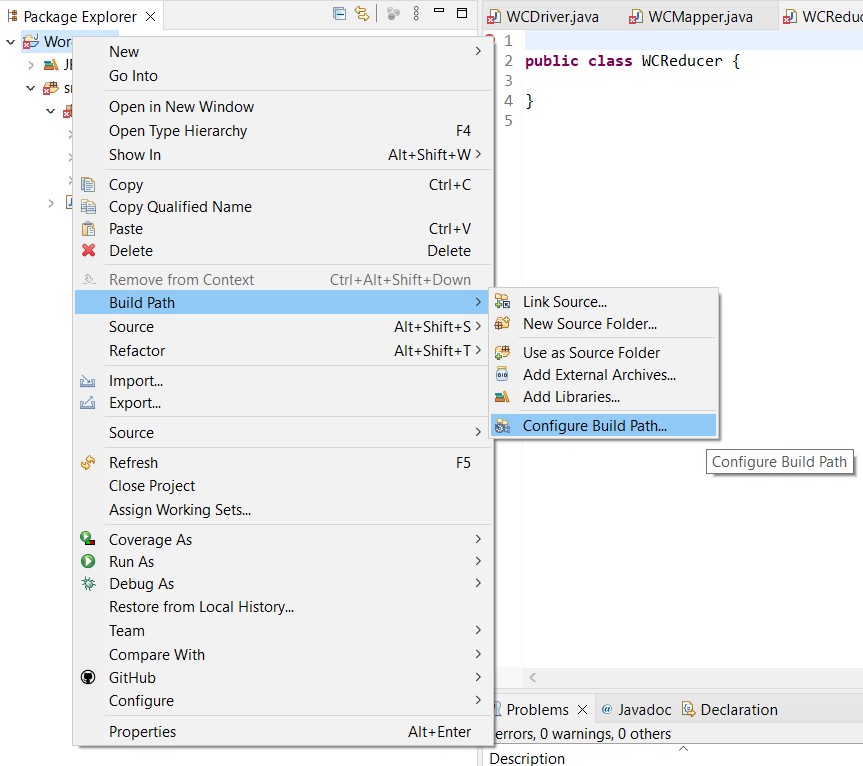








**Step 3:** You have to include two Reference Libraries for that:

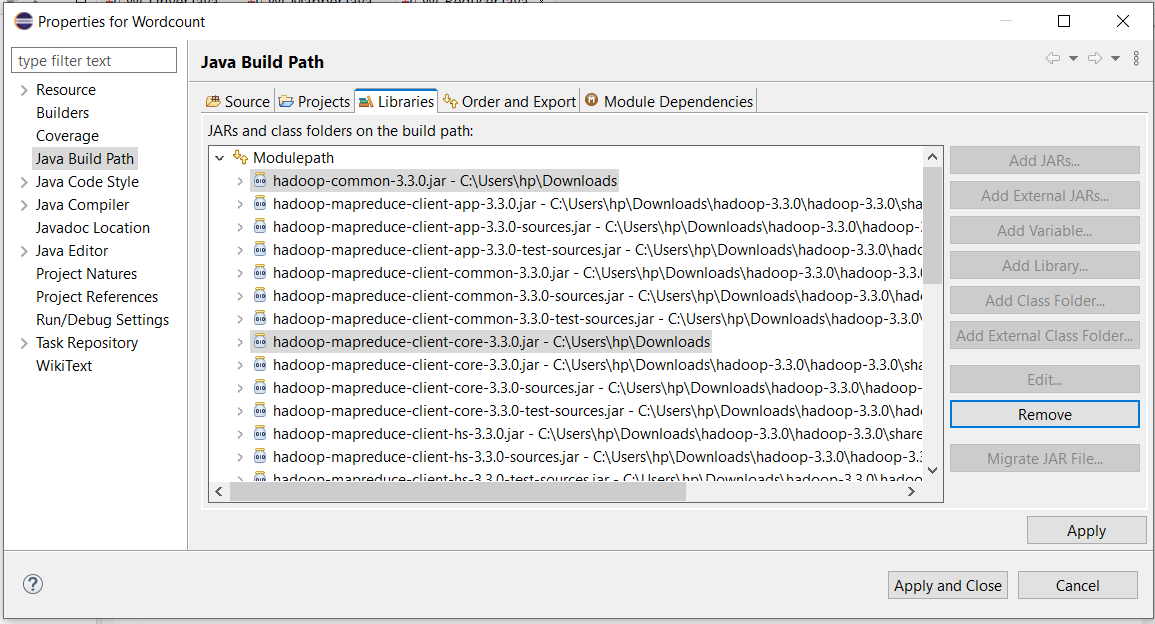
Right Click on **Project** -> then select **Build Path**-> Click on **Configure Build Path**

# In the above figure, you can see the Add External JARs option on the Right Hand Side. Click on it and add the below mentioned files. You can find these files in

***/usr/lib/***

# /usr/lib/hadoop-0.20-mapreduce/hadoop-core-2.6.0-mr1-cdh5.13.0.jar

1. **/usr/lib/hadoop/hadoop-common-2.6.0-cdh5.13.0.jar**



# Now, we have to add this program into the WCMapper Java Class file.

import java.io.IOException;

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

import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,

Text, Text, IntWritable> {

// Map function

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces for (String word : line.split(" "))

{

if (word.length() > 0)

{

output.collect(new Text(word), new IntWritable(1));

}

}

}

## Now, we have to add this program into the WCReducer Java Class file.

import java.io.IOException; import java.util.Iterator;

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

import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class WCReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

// Reduce function

public void reduce(Text key, Iterator<IntWritable> value, OutputCollector<Text, IntWritable> output,

Reporter rep) throws IOException

{

int count = 0;

// Counting the frequency of each words while (value.hasNext())

{

IntWritable i = value.next(); count += i.get();

}

output.collect(key, new IntWritable(count));

}

}

## Now, we have to add this program into the WCDriver Java Class file.

import java.io.IOException;

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.fs.Path;

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

import org.apache.hadoop.mapred.FileInputFormat; import org.apache.hadoop.mapred.FileOutputFormat; import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf; import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

public int run(String args[]) throws IOException

{

if (args.length < 2)

{

System.out.println("Please give valid inputs"); return -1;

}

JobConf conf = new JobConf(WCDriver.class); FileInputFormat.setInputPaths(conf, new Path(args[0])); FileOutputFormat.setOutputPath(conf, new Path(args[1])); conf.setMapperClass(WCMapper.class); conf.setReducerClass(WCReducer.class); conf.setMapOutputKeyClass(Text.class); conf.setMapOutputValueClass(IntWritable.class); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); JobClient.runJob(conf);

return 0;

}

// Main Method

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

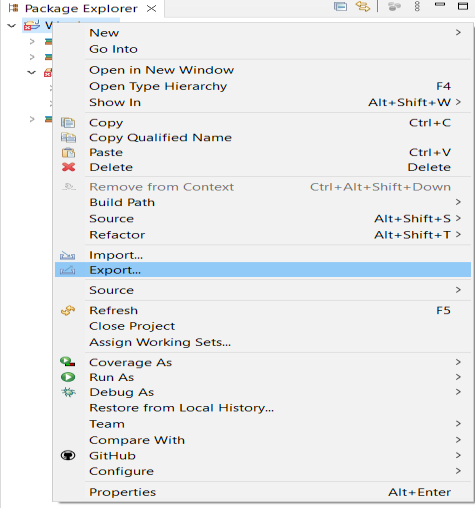
{

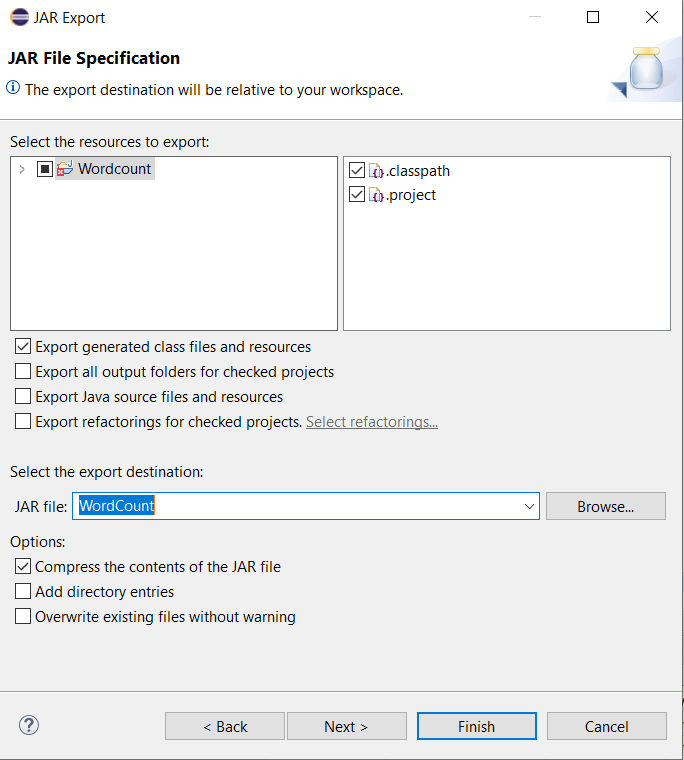
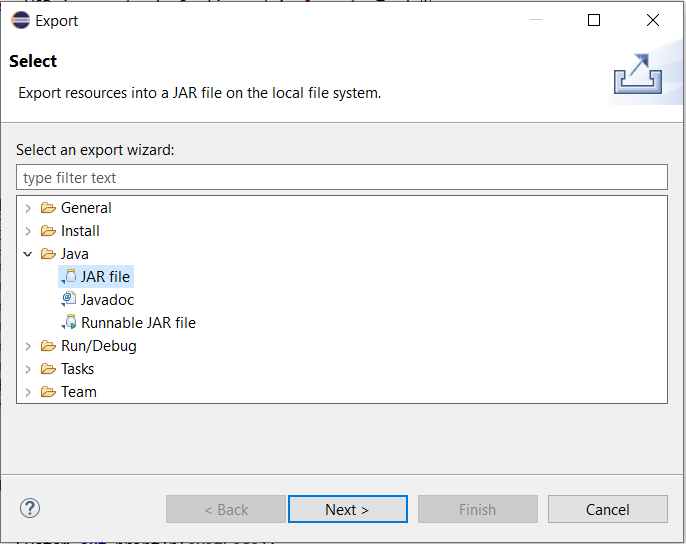
int exitCode = ToolRunner.run(new WCDriver(), args); System.out.println(exitCode);

}

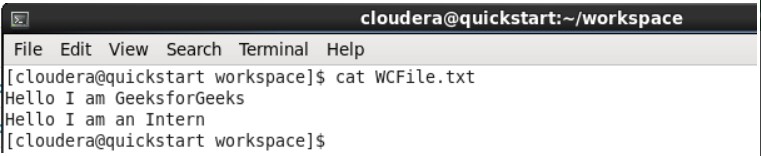
}

**Step 4:** Now you have to make a jar file. Right Click on Project-> Click on Export-> Select export destination as Jar File-> Name the jar File (WordCount.jar) -> Click on next -> at last Click on Finish. Now copy this file into the Workspace directory of Cloudera

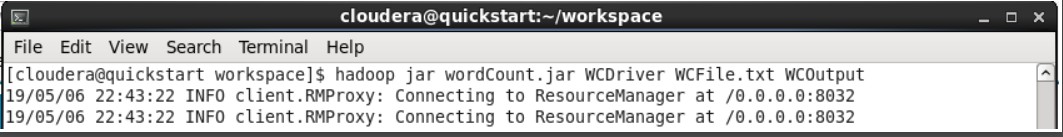




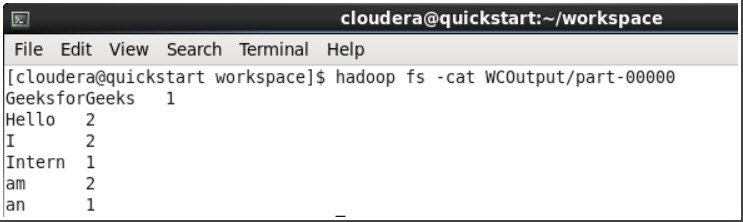
**Step 5:** Open the terminal on CDH and change the directory to the workspace. You can do this by using the “cd workspace/” command. Now, Create a text file (WCFile.txt) and move it to HDFS. Open the terminal and write this code (remember you should be in the same directory as the jar file you have created just now).



# Now, run this command to copy the file input file into the HDFS.

**Now to run the jar file by writing the code as shown in the screenshot.**

# After Executing the code, you can see the result in the WCOutput file or by writing the following command on terminal.



**Practical No:3**

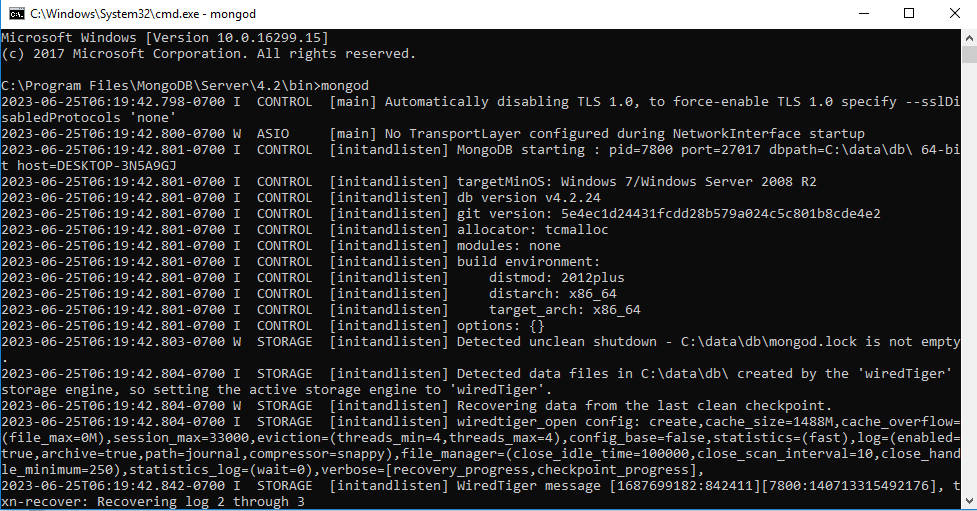
# Aim : Implement an application that store big data in Hbase/MongoDB and manipulate it using R/Python.

Requirements :

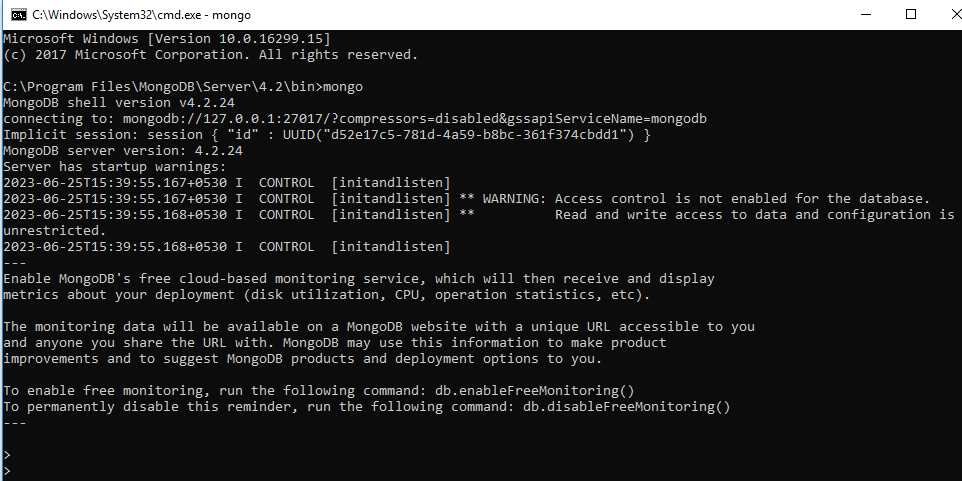
* 1. PyMongo
  2. Mongo Database

# Step 1 :- Install Mongo Database

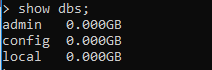
After Installation, in C:\ create “data” folder and in data “db” folder to store database and open C:\ProgramFiles\MongoDB\Server\4.2\bin folder then open cmd and write command mongod :-



Open 2nd cmd and write command “Mongo”:-

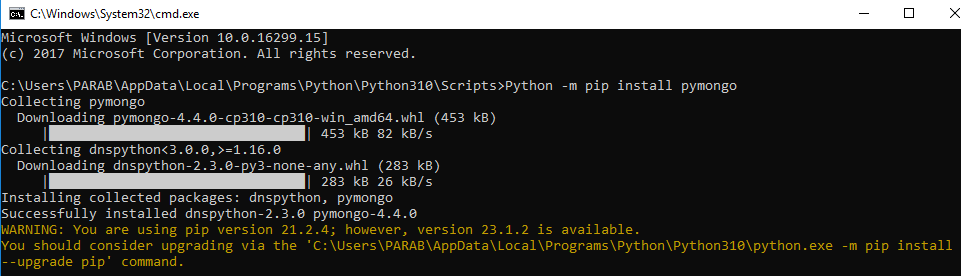


Aslo check the Default Database :-



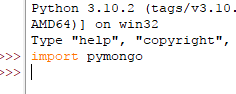
# Step 2 :- install pymongo

Install Python,then go to python file select scripts folder and open cmd to write following command :- Python –m pip install pymongo

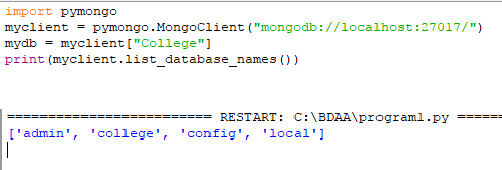


# Step 3 :- Test pymongo :-

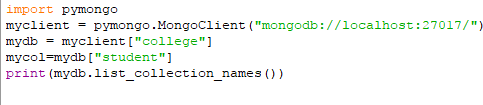
Open python command prompt and write :-



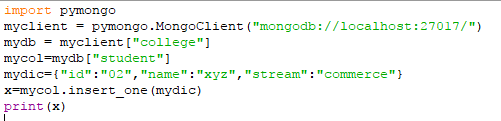
# Step 4:- Creating a database :-



**Step 5 :- Creating a collection:-**

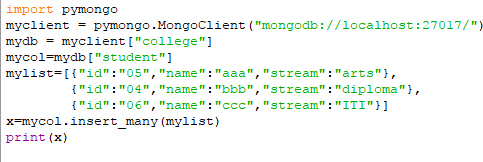


# Step 6:- Inserting one record in collection:



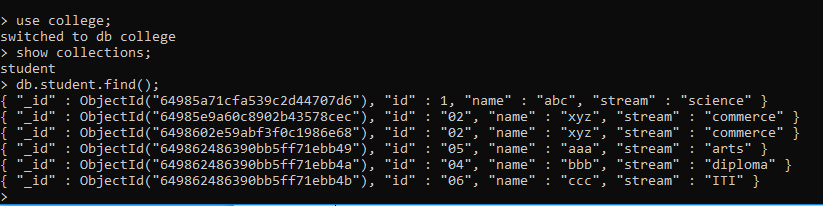


**Step 7 :-Inserting many records in collection.**



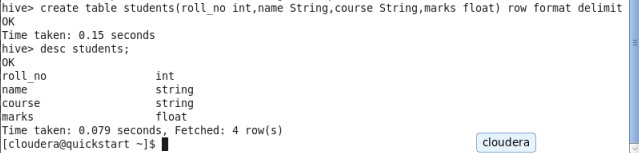
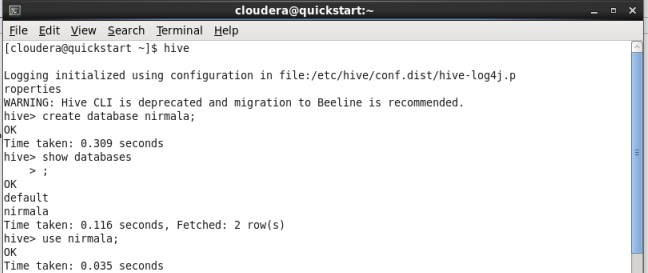


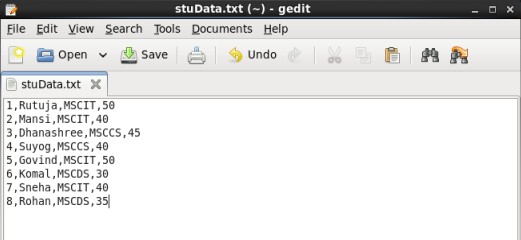
# Checking created db through mongodb server :-

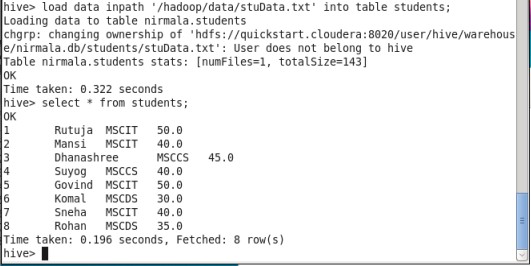


**Practical No:4**

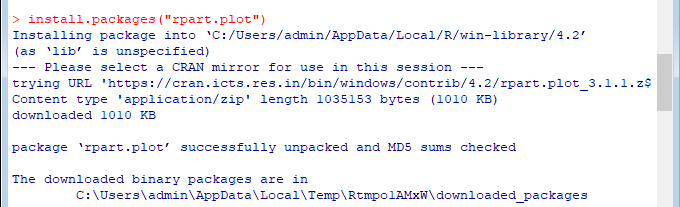
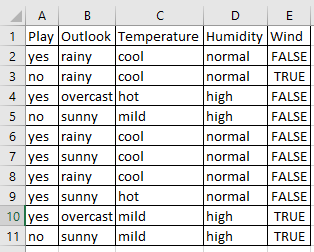
# Aim: Configure hive and implement application in hive.

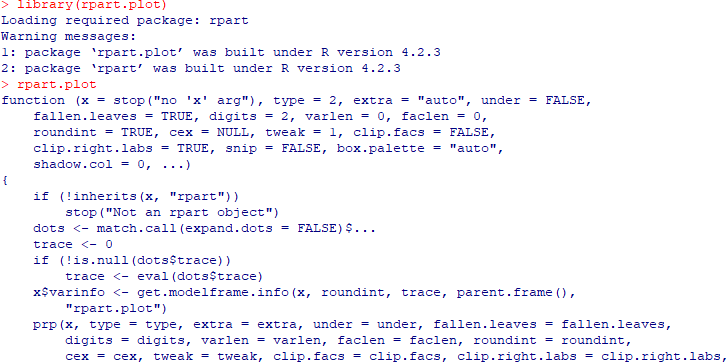


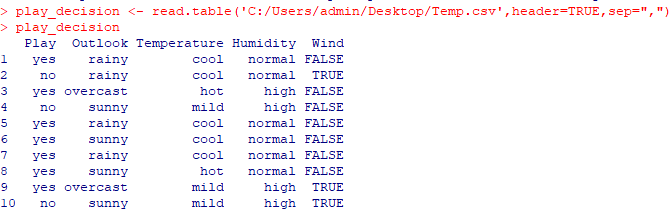
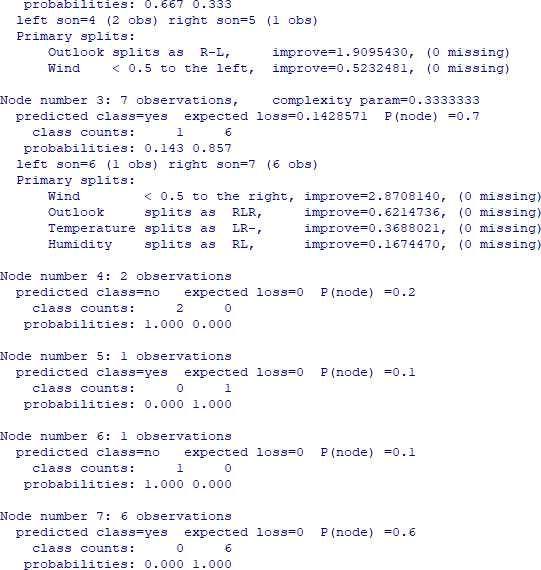
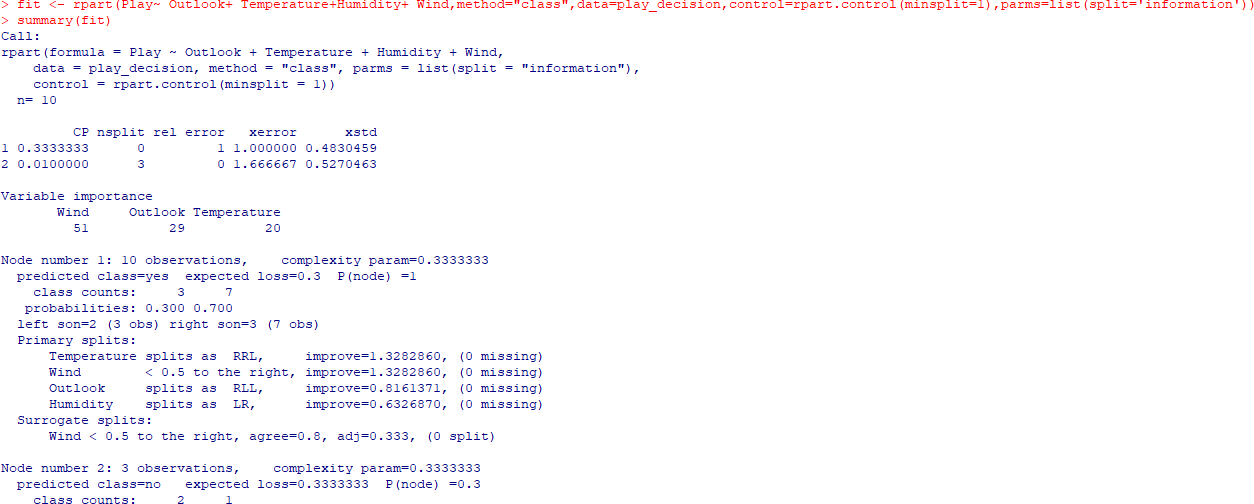
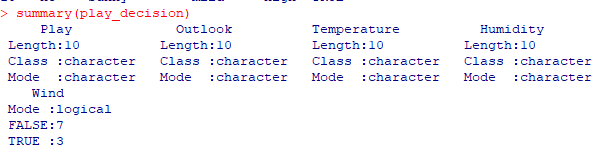




**Practical No:5 Aim :- Implement Decision Tree Techniques**

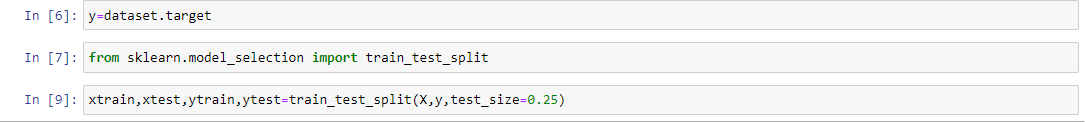
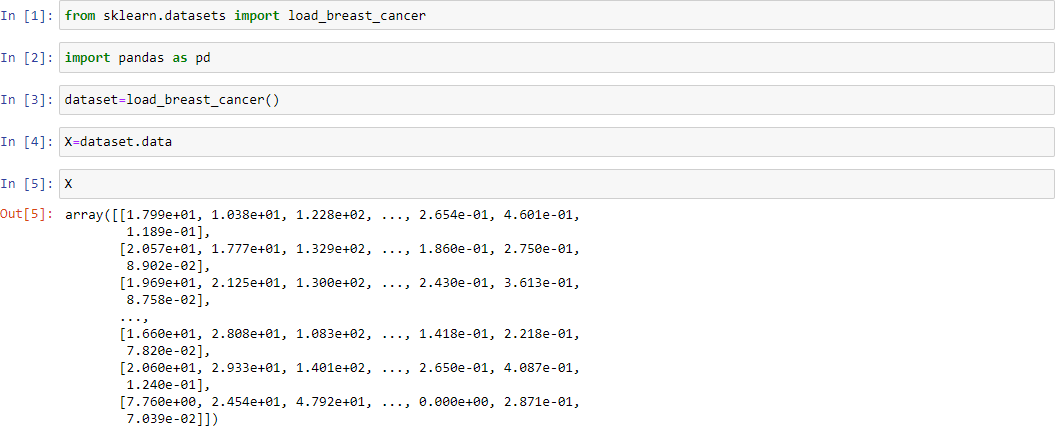


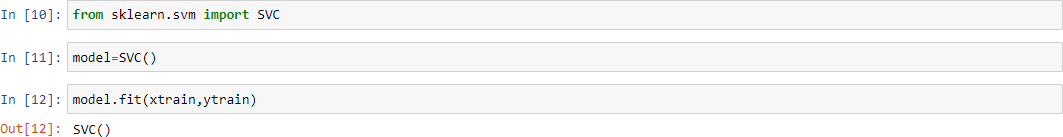


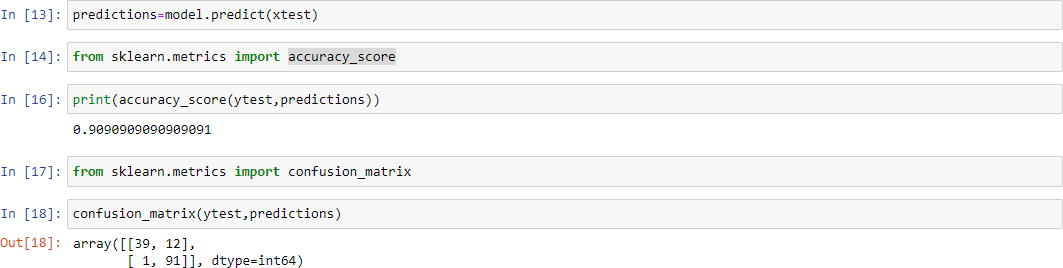




# Output:

**Practical No:6 Aim: Implement SVM Techniques**

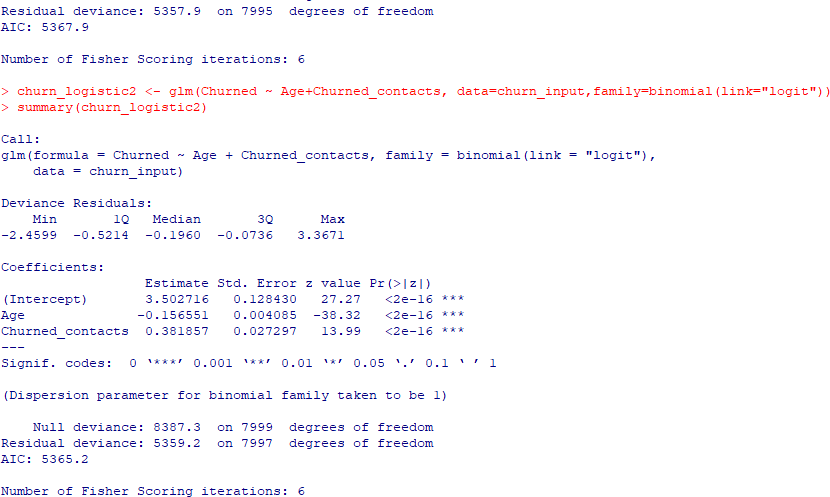
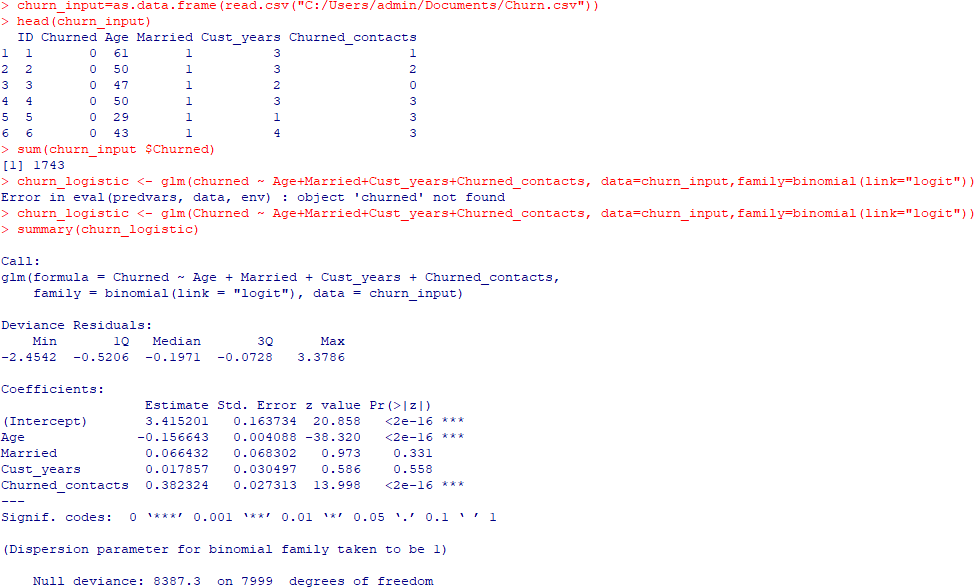


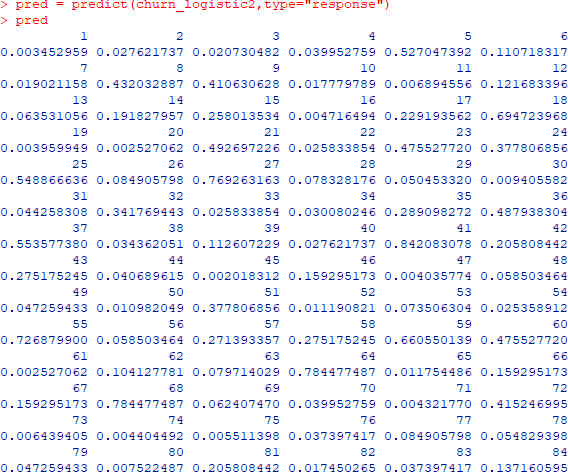


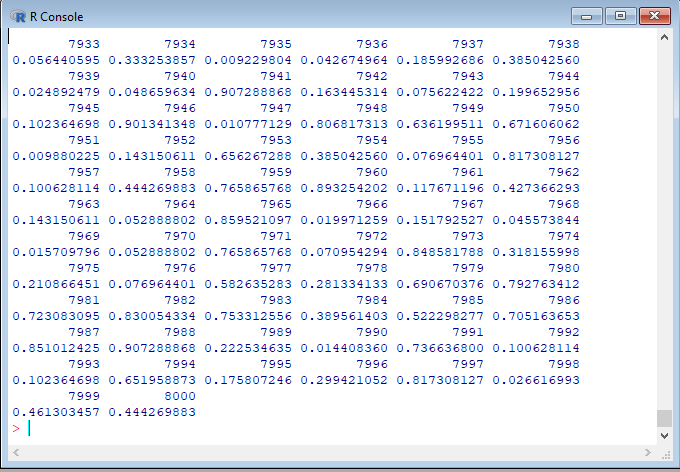
**Practical No:7**

1. **Logistic Regression**

Churn.csv

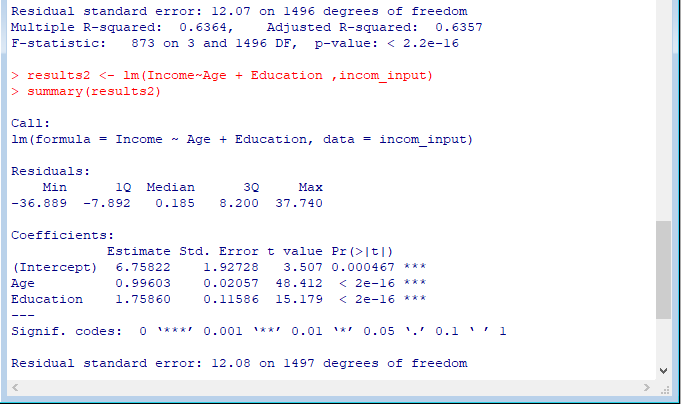
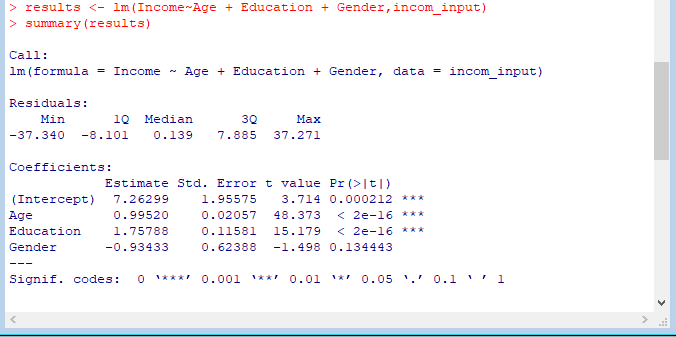
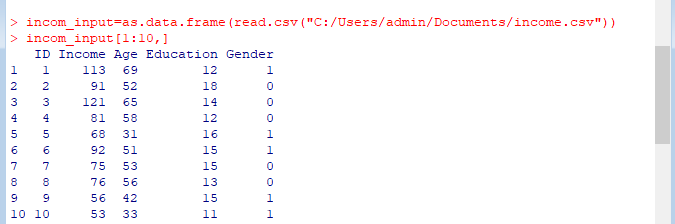


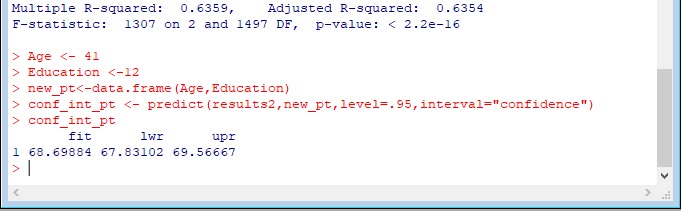




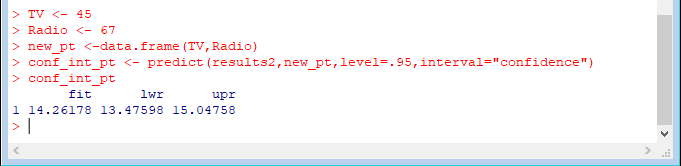
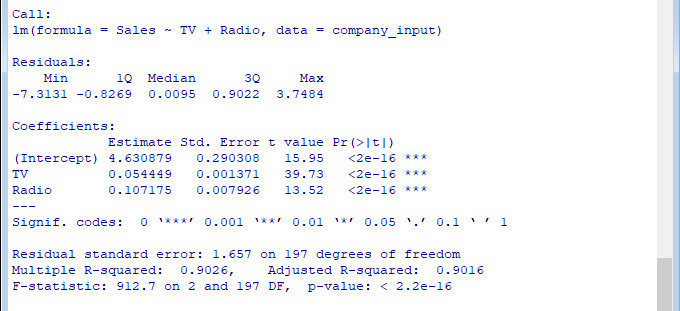
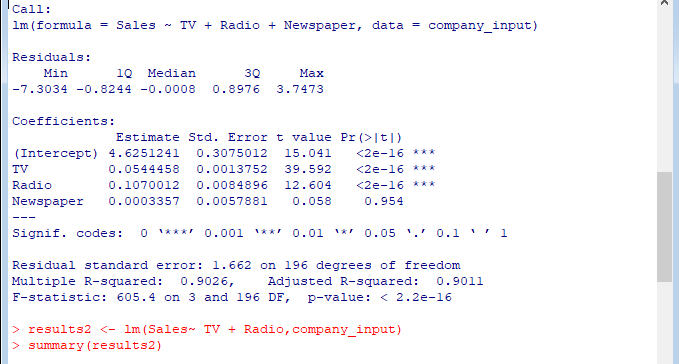
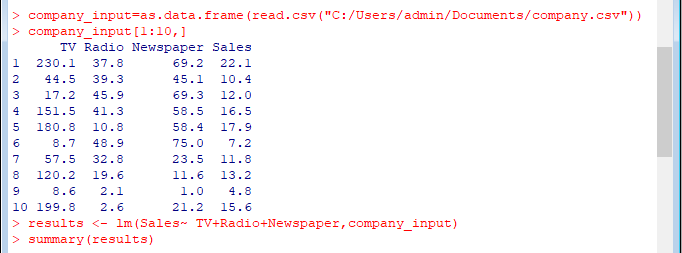
# Linear Regression

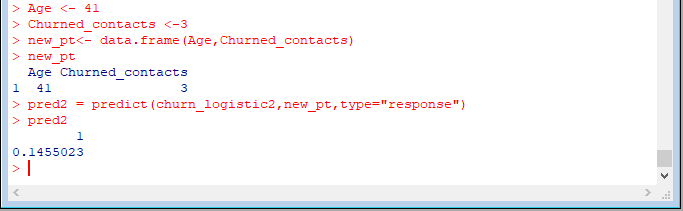
**Income.csv**





Company.csv

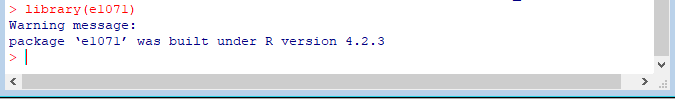
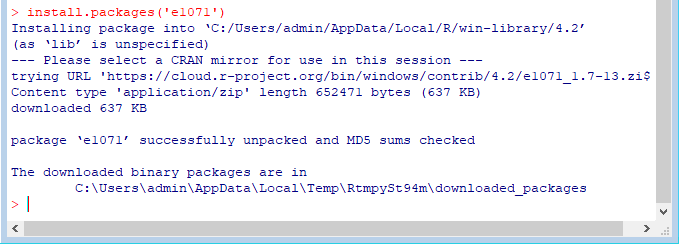


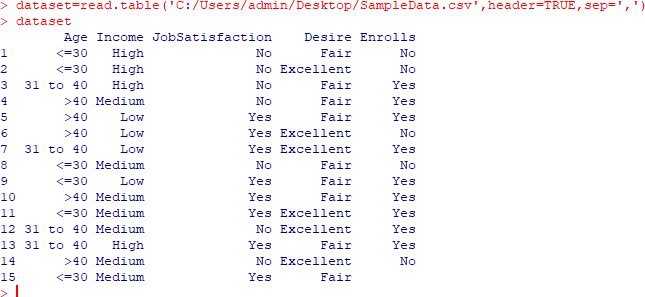
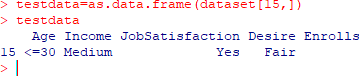
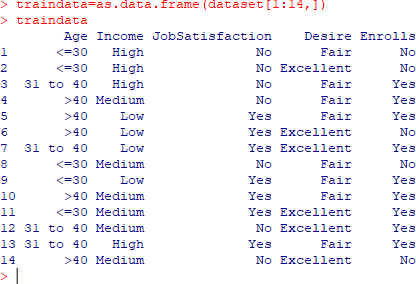


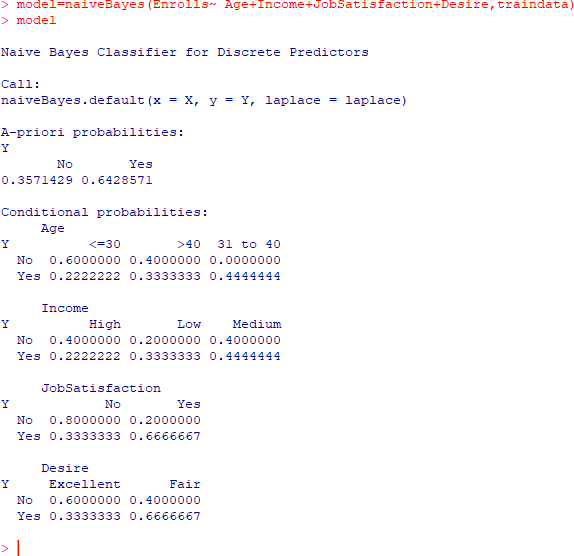
**Practical No:8**

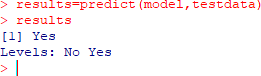
1. **CLASSIFICATION**

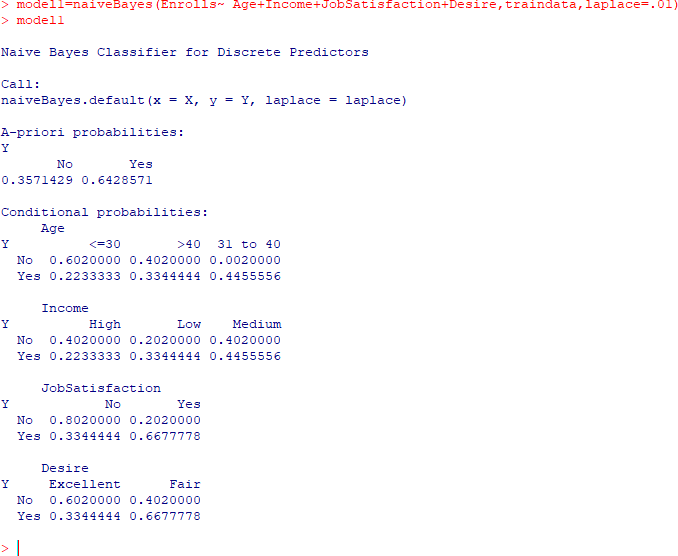
SampleData.csv

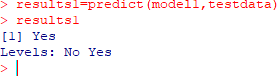






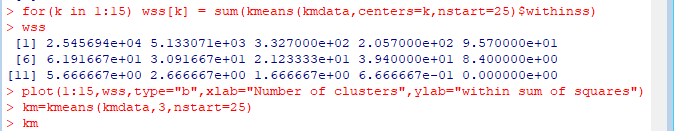
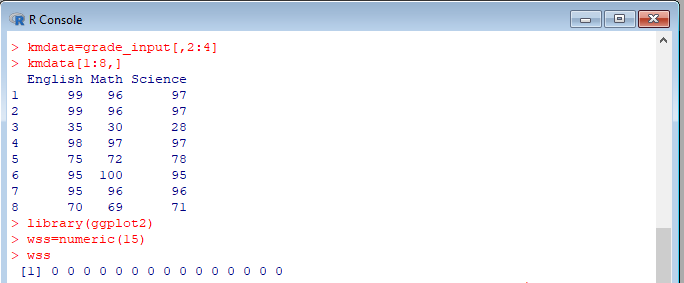
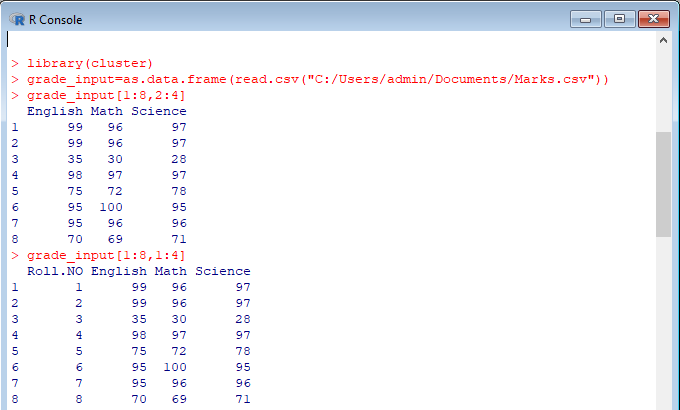
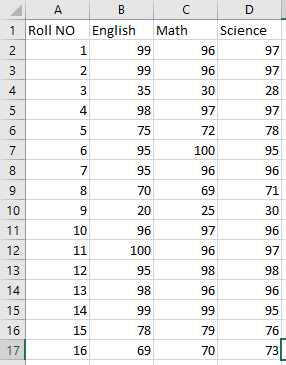


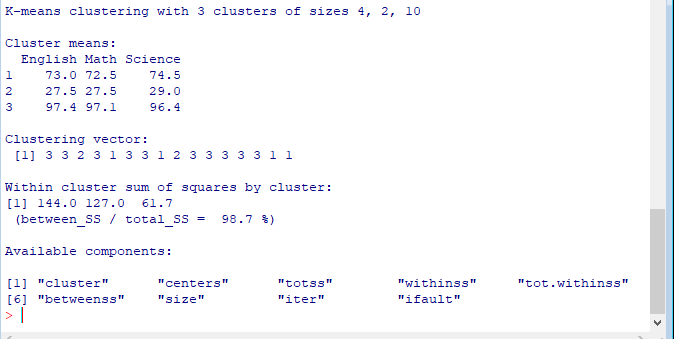


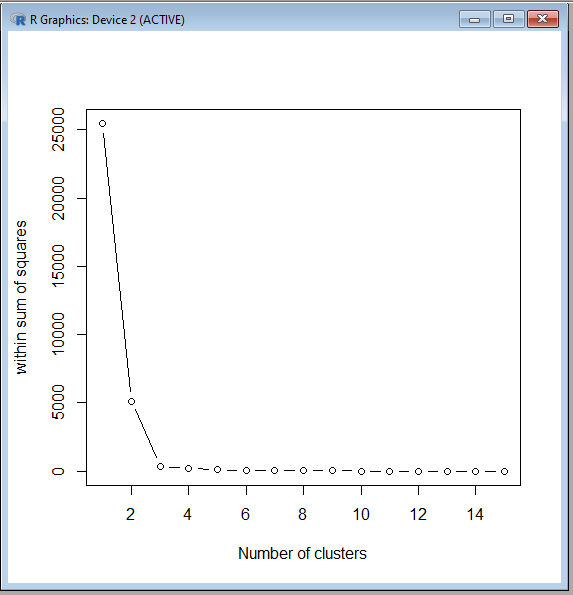


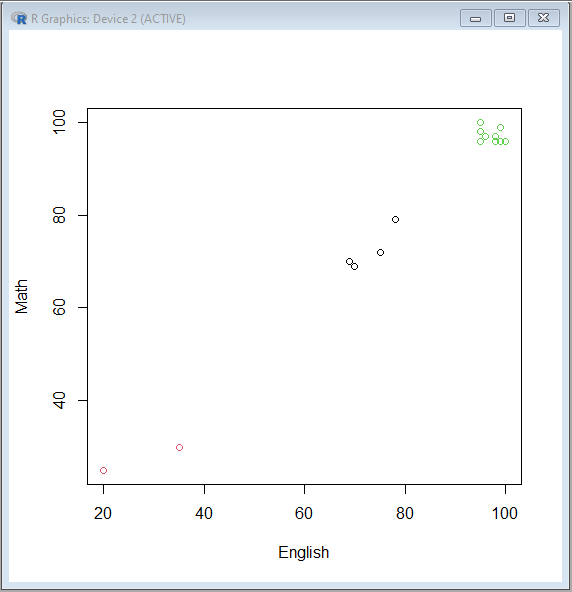
# CLUSTERING

**Marks.csv**









# Flower.CSV

