University Of Mumbai

Institute of Distance & Open Learning

PRACTICAL JOURNAL

BIG DATA ANALYTICS

SUBMITTED BY

DRIZZLE GONSALVES

SEAT NO:8166014

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY
PART-I SEMESTER II

ACADEMIC YEAR
2024-2025

INSTITUTE OF DISTANCE AND OPEN LEARNING IDOL BUILDING,
VIDYANAGARI, SANTACRUZ (EAST), MUMBAI-400 098

CONDUCTED AT
RIZVI COLLEGE OF ARTS, SCIENCE AND COMMERCE BANDRA (W),
MUMBAI 400050

University Of Mumbai
Institute of Distance & Open Learning

Dr.Shankar Dayal Sharama Bhavan, Kalina, Vidanagari,
Santacruz (E), Mumbai-400 098.

PCP CENTRE :
RIZVI COLLEGE OF ARTS, SCIENCE AND COMMERCE, BANDRA(W), MUMBAI

Certificate

This is to certify that

Ms. DRIZZLE GONSALVES **of M.Sc.-I.T. SEM-II, Seat No: *8166014* has performed all the Practical of *BIG DATA ANALYTICS* for MSC.IT Part I. This Journal represents her Bonafide work during the academic year: 2024-2025.**

MSc (IT) Co-ordinator, IDOL

External Examiner

INDEX

Practical 1

Install, configure and run Hadoop and HDFS ad explore HDFS.

Step 1 Install Hadoop

Login to ubuntu

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-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/
```

Steps to get hdfs file system working:

1.In this section, we’ll go over basic interactions with the distributed file system via the command line. It is assumed that these commands are performed on a client that can connect to a remote Hadoop cluster, or which is running a pseudo-distributed clusteron the localhost. It is also assumed that the hadoop command and other utilities from $HADOOP_HOME/bin are on the system path and can be found by the operating system

All of the usual file system operations are available to the user, such as creating directories; moving, removing, and copying files; listing directories; and modifying permissions of files on the cluster. To see the available commands in the fs shell, type:

```
hostname $ hadoop fs -help
```

```
Usage: hadoop fs [generic options]
```

/usr/local/hadoop/bin/hadoop fs -help

As you can see, many of the familiar commands for interacting with the file system are there, specified as arguments to the hadoop fs command as flag arguments in the Java style—that is, as a single dash (–) supplied to the command. Secondary flags or options to the command are specified with additional Java style arguments delimited by spaces following the initial command. Be aware that order can matter when specifying such options.

To get started, let's copy some data from the local file system to the remote (dis-tributed) file system. To do this, use either the put or **copyFromLocal** commands. These commands are identical and write files to the distributed file system without removing the local copy. The **moveFromLocal** command is similar, but the local copy is deleted after a successful transfer to the distributed file system. In the /data directory of the GitHub repository for this book's code and resources, there is a shakespeare.txt file containing the complete works of William Shakespeare.

Download this file to your local working directory. After download, move the file to the distributed file system as follows:

hostname $ hadoop fs –copyFromLocal shakespeare.txt shakespeare.txt

This example invokes the Hadoop shell command copyFromLocal with two arguments, <src> and <dst>, both of which are specified as relative paths to a file called shakespeare.txt. To be explicit about what's happening, the command searches your current working directory for the shakespeare.txt file and copies it to the /user/analyst/shakespeare.txt path on HDFS by first requesting information about that path from the NameNode, then directly communicating with the DataNodes to transfer the file. Because Shakespeare's complete works are less than 64 MB, it is not broken up into blocks. Note, however, that on both your local machine, as well as the remote HDFS system, relative and absolute paths must be taken into account. The preceding command is shorthand for:

```
hostname $ hadoop fs –put /home/analyst/shakespeare.txt \

hdfs://localhost/user/analyst/shakespeare.txt
```

You'll note that there exists a home directory on HDFS that is similar to the home directory on POSIX systems; this is what the /user/analyst/ directory is—the home directory of the analyst user. Relative paths in reference to the remote file system treat the user's HDFS home directory as the current working directory. In fact, HDFS has a permissions model for files and directories that are very similar to POSIX. In order to better manage the HDFS file system, create a hierarchical tree of directories just as you would on your local file system:

hostname $ hadoop fs -mkdir corpora

To list the contents of the remote home directory, use the ls command:

hostname $ hadoop fs –ls .

```
drwxr-xr-x - analyst analyst 0 2015-05-04 17:58 corpora

-rw-r--r-- 3 analyst analyst 8877968 2015-05-04 17:52 shakespeare.txt
```

The HDFS file listing command is similar to the Unix ls –l command with some HDFS-specific features. Specified without any arguments, this command provides a listing of the user's HDFS home directory. The first column shows the permissions mode of the file. The second column is the replication of the file; by default, the replication is 3. Note that directories are not replicated, so this column is a dash (-) in that case. The user and group follow, then the size of the file in bytes (zero for directories). The last modified date and time is up next, with the name of the file appearing last. Other basic file operations like mv, cp, and rm will all work as expected on the remote file system. There is, however, no rmdir command; instead, use rm –R to recursively remove a directory with all files in it. Reading and moving files from the distributed file system to the local file system should be attempted with care, as the distributed file system is maintaining files that are extremely large. However, there are cases when files need to be inspected in detail by the user, particularly output files that are produced as the result of MapReduce jobs. Typically these are not read to the standard output stream but are piped to other programs like less or more. To read the contents of a file, use the cat command, then pipe the output to less in order view the contents of the remote file:

hostname $ hadoop fs –cat shakespeare.txt | less

There is no similar hadoop fs -head command to inspect the first kilobyte of the file. Instead, it is efficient to hadoop fs -cat the file and pipe it to the local shell's head, as the head command terminates the remote stream before the entire file is read. However, using the shell's tail in this manner would be less efficient, as all of the data would have to be streamed from the remote file system to your local file system before the output could be computed. Instead the, hadoop fs –tail command seeks to the correct position in the remote file and returns only the required data over the network. To transfer entire files from the distributed file system to the local file system, use get or copyToLocal, which are identical commands. Similarly, use the moveToLocal command, which also removes the file from the distributed file system. Finally, the get merge command merges all files that match a given pattern or directory are copied and merged into a single file on the localhost. If files on the remote system are large, you may want to pipe them to a compression utility:

hostname $ hadoop fs –get shakespeare.txt ./shakespeare.from-remote.txt

Comparing the original shakespeare.txt file should prove that it is identical to the shakespeare.from-remote.txt file. Hopefully we have demonstrated that the hadoop fs command is a fully featured command-line interface to HDFS and is used routinely when developing analytical jobs. Table 2-1 demonstrates other useful commands that are provided by hadoop fs.

hadoop fs -help <cmd> Provides information and flags specifically about the <cmd> in question.

hadoop fs -test <path> Answer various questions about <path> (e.g., exists, is directory, is file, etc.)

hadoop fs -count <path> Count the number of directories, files, and bytes under the paths that match the

specified file pattern.

hadoop fs -du -h <path> Show the amount of space, in bytes, used by the files that match the specified file pattern.

hadoop fs -stat <path> Print statistics about the file/directory at <path>.

hadoop fs -text <path> Takes a source file and outputs the file in text format. Currently Zip,

TextRecordInputStream, and Avro sources are supported

File Permissions in HDFS

As mentioned earlier, HDFS has POSIX-like file permissions. There are three types of permissions: read (r), write (w), and execute (x). These permissions define the access levels for the owner, the group, and any other system users. For directories, the exe- cute permission allows access to the contents of the directory; however, execute per-missions are ignored on HDFS for files.

Read and write permissions in the context of HDFS specify who can access the data and who can append to the file. Permissions are expressed during the directory listing command ls. Each mode has 10 slots. The first slot is a d for directories, otherwise a – for files. Each of the follow- ing groups of three indicates the rwx permissions for the owner, group, and other users, respectively.

There are several HDFS shell commands that will allow you to manage the permissions of files and directories, namely the familiar chmod, chgrp, and chown commands: hostname $ hadoop fs –chmod 664 shakespeare.txt This command changes the permissions of shakespeare.txt to -rw-rw-r--. The 664 is an octal representation of the flags to set for the permission triple. Consider 6 in binary, 110—this means set the read and write flags but not the execute flag. Com- pletely permissible is 7, 111 in binary and read-only is 4, 100 in binary. The chgrp and chown commands change the group and owner of the files on the distributed file system.

A caveat with file permissions on HDFS: the identity of the client is determined by the username and groups of the process operating across HDFS, which means remote clients can create arbitrary users on the system. These permissions, therefore, should only be used to prevent accidental data loss and to share file system resources between known users, not as a security mechanism

Other HDFS Interfaces

Programmatic access to HDFS is made available to software developers through a Java API, and any serious data ingestion into a Hadoop cluster should consider utiliz- ing that API. There are also other tools for integrating HDFS with other file systems or network protocols—for example, FTP or Amazon S3. In Chapter 6, we'll focus more on data management issues and how to acquire and store data from a variety of sources into HDFS.

There are also HTTP interfaces to HDFS, which can be used for routine administra- tion of the cluster file system and programmatic access to HDFS with Python. HTTP access to HDFS comes in two primary interfaces: direct access through the HDFS daemons that serve HTTP requests, or via proxy servers that expose an HTTP inter- face then directly access HDFS on the client's behalf using the Java API.

Examples of proxies include HttpFS, Hoop, and WebHDFS, each of which allow RESTful network access to the Hadoop cluster, which is easily programmed against using Python. The NameNode also supplies direct, read-only access to HDFS over HTTP through an HTTP server that runs on port 50070. If running in pseudo-distributed mode, simply open a browser and navigate to http://127.0.0.1:50070; otherwise, use the host name of the NameNode on your cluster. The NameNode Web UI provides a high

level overview of the cluster status, including the amount of storage available and used, the number of alive and dead DataNodes, and warning information about under-replicated blocks. The NameNode also allows users to browse the file system using a search and naviga- tion utility that is found under the Utilities drop-down tab. File meta information is listed similar to the command-line interface, and specific files may even be made available for download. DataNodes themselves can be directly browsed for informa- tion, accessing the DataNode host on port 50075; all active DataNodes are listed on the NameNode HTTP site.

By default, the direct HTTP interface is read-only. In order to provide write access to an HDFS cluster, a proxy such as WebHDFS must be used. WebHDFS secures the cluster via authentication with Kerberos. Accessing secure resources on Hadoop depends primarily on the specific configuration of the cluster, and if any third-party management tools are being used. Hadoop was designed to run on completely man- aged internal clusters without exposure to the outside world, and as a result security in Hadoop is not as mature as the platform itself—although this is one of the prime considerations of Hadoop development moving forward

Practical 2

Implement word count / frequency programs using MapReduce
Map Reduce as two component Map and Reduce.

Java program:

Write program save as WordCount.java

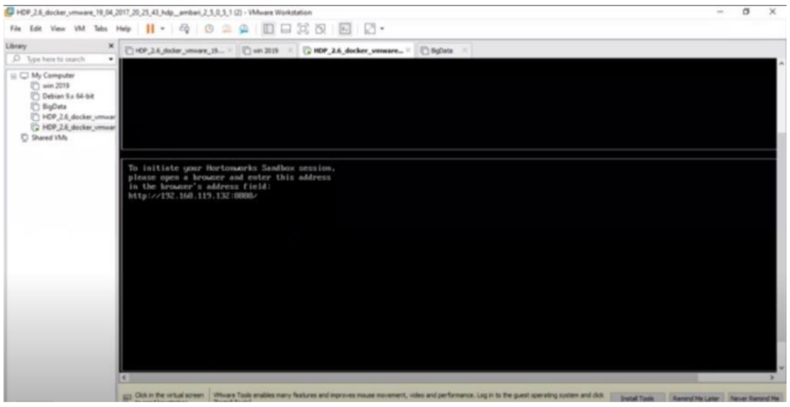
```
///////////////////////
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {

 public static class TokenizerMapper extends Mapper<Object, Text, Text, IntWritable>{
 private final static IntWritable one = new IntWritable(1);
 private Text word = new Text();
 public void map(Object key, Text value, Context context
 ) throws IOException, InterruptedException {
 StringTokenizer itr = new StringTokenizer(value.toString());
 while (itr.hasMoreTokens()) {//"This is the output is the"
 word.set(itr.nextToken());
 context.write(word, one);
 }
 }
 }
 public static class IntSumReducer extends Reducer<Text,IntWritable,Text,IntWritable> {
 private IntWritable result = new IntWritable();
 public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException
 {//is,3
 int sum = 0;
 for (IntWritable val : values) {
 sum += val.get();
 }
 result.set(sum);
 context.write(key, result);
 }
 }

public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
 Job job = Job.getInstance(conf, "word count");
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setCombinerClass(IntSumReducer.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 FileInputFormat.addInputPath(job, new Path(args[0]));
 FileOutputFormat.setOutputPath(job, new Path(args[1]));
 System.exit(job.waitForCompletion(true)?0:1);
 }
}
```

Text File:

Hello World

This is the output is the

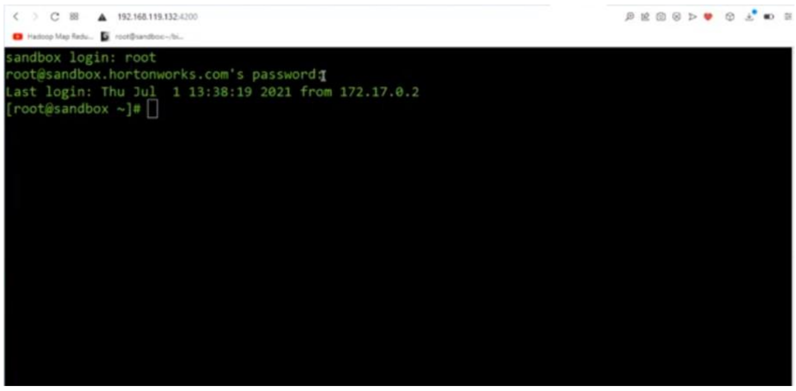
Start the server (Horton Sandbox)



```
To initiate your Hortonworks Sandbox session,
please open a browser and enter this address
in the browser's address field:
http://192.168.119.132:8888/
```

Open the terminal with 192.168.119.132/4200

Enter the login: root and the password and enter

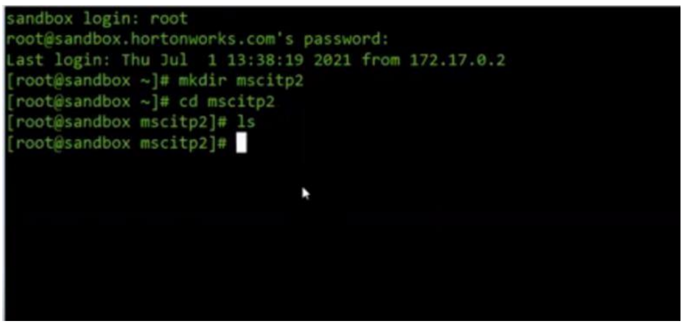


```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Thu Jul  1 13:38:19 2021 from 172.17.0.2
[root@sandbox ~]# 
```

Create a folder in local directory.

Command: mkdir mscitp2

Change the directory cd mscitp2



```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Thu Jul  1 13:38:19 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp2
[root@sandbox ~]# cd mscitp2
[root@sandbox mscitp2]# ls
[root@sandbox mscitp2]#
```

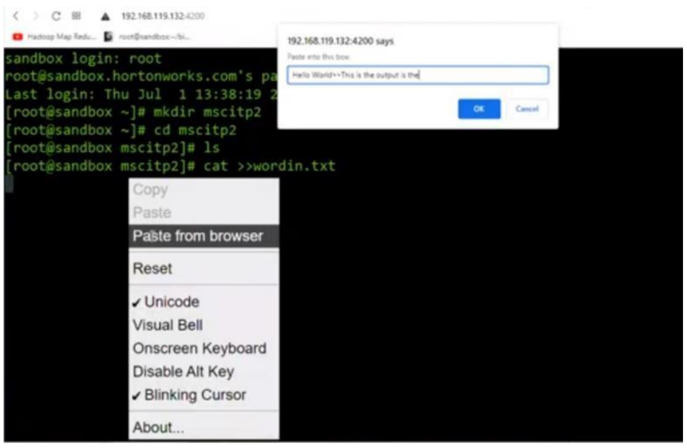
Now create input file

Command: cat >> wordin.txt

Paste the text by right clicking on terminal

Hello World

This is the output is the

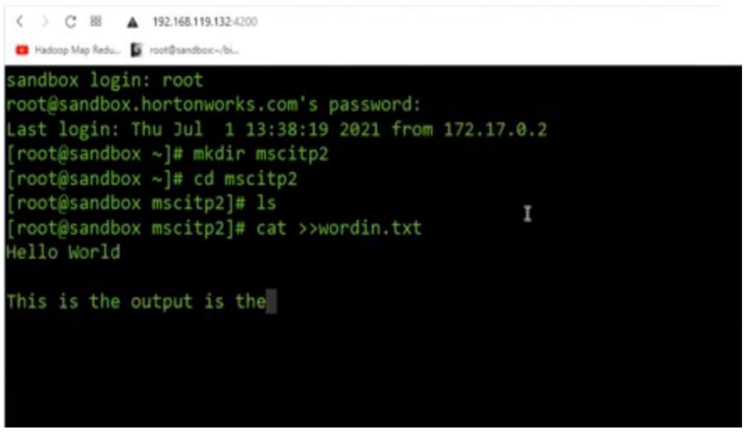


```
sandbox login: root
root@sandbox.hortonworks.com's pa
Last login: Thu Jul  1 13:38:19 2
[root@sandbox ~]# mkdir mscitp2
[root@sandbox ~]# cd mscitp2
[root@sandbox mscitp2]# ls
[root@sandbox mscitp2]# cat >>wordin.txt
```

192.168.119.132:4200 says

Paste into this box:

Hello World↵↵This is the output is the



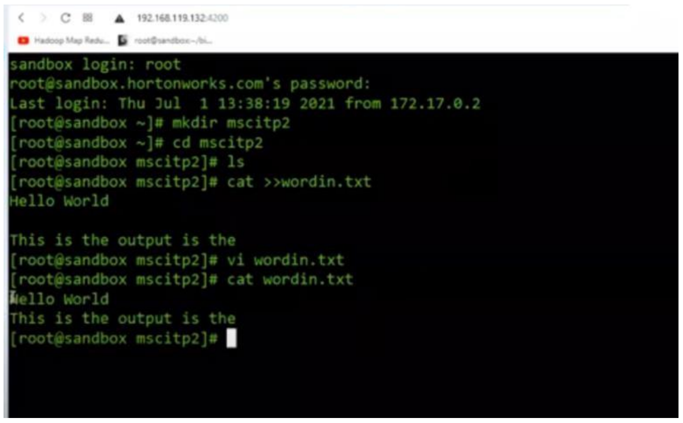
```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Thu Jul  1 13:38:19 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp2
[root@sandbox ~]# cd mscitp2
[root@sandbox mscitp2]# ls
[root@sandbox mscitp2]# cat >>wordin.txt
Hello World

This is the output is the
```

To remove the extra space type command

vi wordin.txt

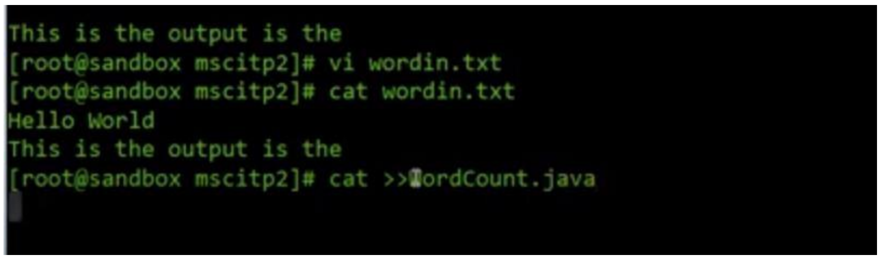
After removing the extra space check the content of the file

cat wordin.txt



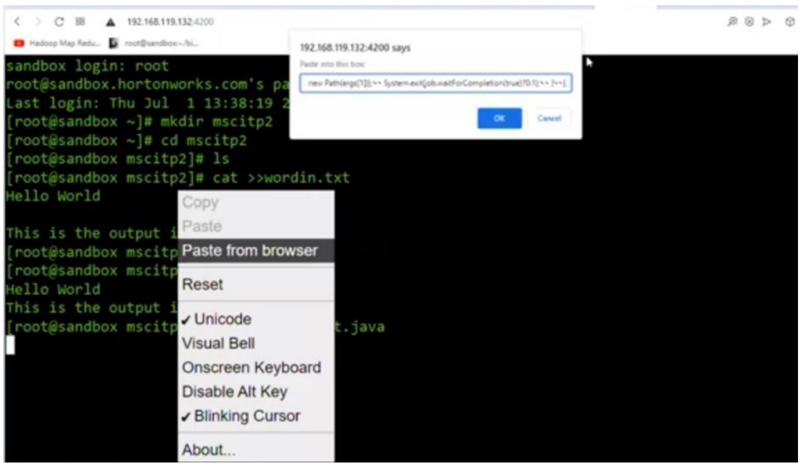
```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Thu Jul  1 13:38:19 2021 from 172.17.0.2
[root@sandbox ~]# mkdir mscitp2
[root@sandbox ~]# cd mscitp2
[root@sandbox mscitp2]# ls
[root@sandbox mscitp2]# cat >>wordin.txt
Hello World

This is the output is the
[root@sandbox mscitp2]# vi wordin.txt
[root@sandbox mscitp2]# cat wordin.txt
Hello World
This is the output is the
[root@sandbox mscitp2]#
```

Create another file wordcount.java



```
This is the output is the
[root@sandbox mscitp2]# vi wordin.txt
[root@sandbox mscitp2]# cat wordin.txt
Hello World
This is the output is the
[root@sandbox mscitp2]# cat >>WordCount.java
```

Paste the java code.



192.168.119.132:4200 says

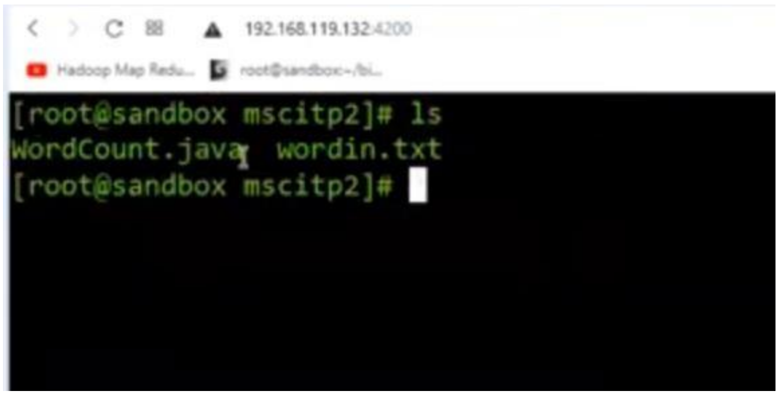
Paste into this box:

```
new Path(args[1]);\~~ System.exit(job.waitForCompletion(true)?0:1);\~~ }\~~}
```

```
sandbox login: root
root@sandbox.hortonworks.com's pa
Last login: Thu Jul  1 13:38:19 2
[root@sandbox ~]# mkdir mscitp2
[root@sandbox ~]# cd mscitp2
[root@sandbox mscitp2]# ls
[root@sandbox mscitp2]# cat >>wordin.txt
Hello World

This is the output i
[root@sandbox mscitp
[root@sandbox mscitp
Hello World
This is the output i
[root@sandbox mscitp                t.java
```

Press control d to save the file

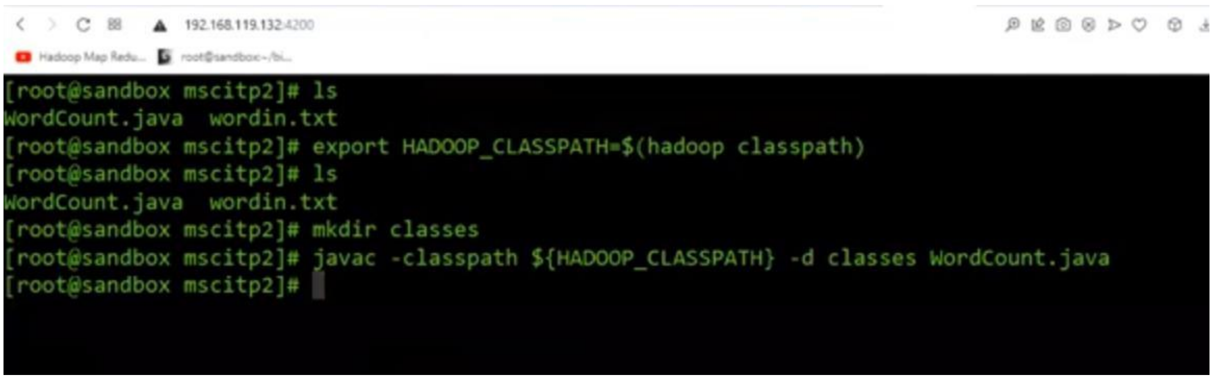
Check both the files create with command ls



```
[root@sandbox mscitp2]# ls
WordCount.java  wordin.txt
[root@sandbox mscitp2]#
```

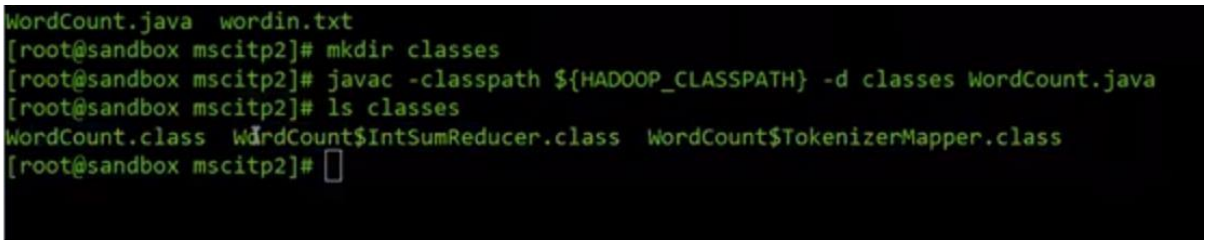
Now, to compile the java file

export HADOOP_CLASSPATH=$(hadoop classpath)
mkdir classes (To keep the compile files)
javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java



```
[root@sandbox mscitp2]# ls
WordCount.java  wordin.txt
[root@sandbox mscitp2]# export HADOOP_CLASSPATH=$(hadoop classpath)
[root@sandbox mscitp2]# ls
WordCount.java  wordin.txt
[root@sandbox mscitp2]# mkdir classes
[root@sandbox mscitp2]# javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java
[root@sandbox mscitp2]#
```

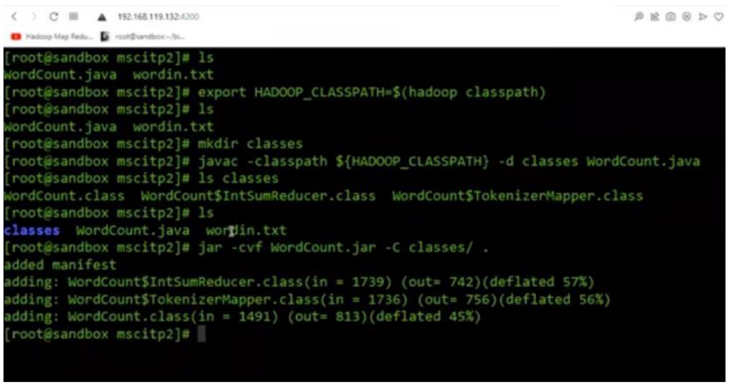
Check class files are created with command ls classes



```
WordCount.java  wordin.txt
[root@sandbox mscitp2]# mkdir classes
[root@sandbox mscitp2]# javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java
[root@sandbox mscitp2]# ls classes
WordCount.class  WordCount$IntSumReducer.class  WordCount$TokenizerMapper.class
[root@sandbox mscitp2]#
```

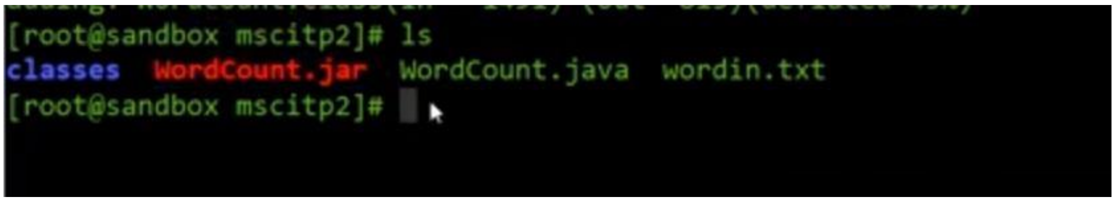
Now we have to bind all the class into single jar file with below command

jar -cvf WordCount.jar -C classes/ .



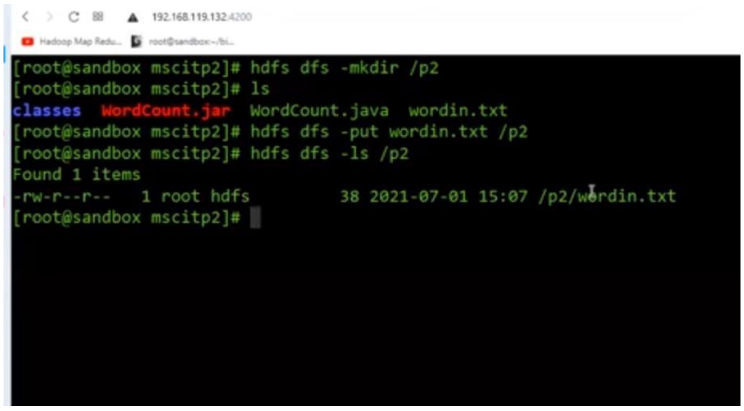
```
[root@sandbox mscitp2]# ls
WordCount.java  wordin.txt
[root@sandbox mscitp2]# export HADOOP_CLASSPATH=$(hadoop classpath)
[root@sandbox mscitp2]# ls
WordCount.java  wordin.txt
[root@sandbox mscitp2]# mkdir classes
[root@sandbox mscitp2]# javac -classpath ${HADOOP_CLASSPATH} -d classes WordCount.java
[root@sandbox mscitp2]# ls classes
WordCount.class  WordCount$IntSumReducer.class  WordCount$TokenizerMapper.class
[root@sandbox mscitp2]# ls
classes  WordCount.java  wordin.txt
[root@sandbox mscitp2]# jar -cvf WordCount.jar -C classes/ .
added manifest
adding: WordCount$IntSumReducer.class(in = 1739) (out= 742)(deflated 57%)
adding: WordCount$TokenizerMapper.class(in = 1736) (out= 756)(deflated 56%)
adding: WordCount.class(in = 1491) (out= 813)(deflated 45%)
[root@sandbox mscitp2]#
```

Run ls command we can see jar file is created.



```
[root@sandbox mscitp2]# ls
classes  WordCount.jar  WordCount.java  wordin.txt
[root@sandbox mscitp2]#
```

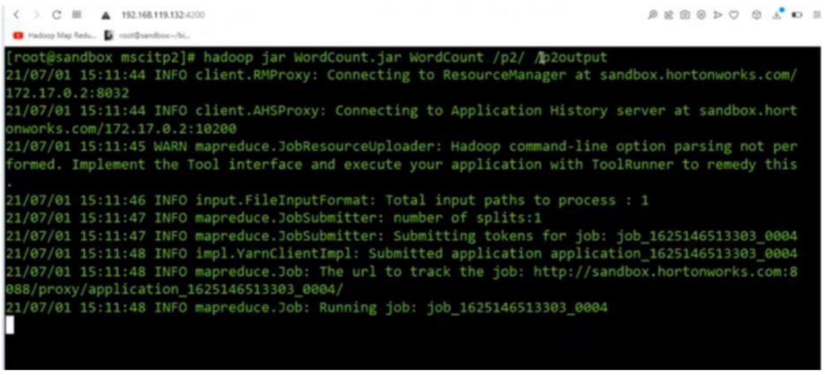
wordin.txt should be present in word directory of hdfs. So we need to upload wordin.txt file.



```
[root@sandbox mscitp2]# hdfs dfs -mkdir /p2
[root@sandbox mscitp2]# ls
classes  WordCount.jar  WordCount.java  wordin.txt
[root@sandbox mscitp2]# hdfs dfs -put wordin.txt /p2
[root@sandbox mscitp2]# hdfs dfs -ls /p2
Found 1 items
-rw-r--r--   1 root hdfs         38 2021-07-01 15:07 /p2/wordin.txt
[root@sandbox mscitp2]#
```

We need to put the final output p2output.

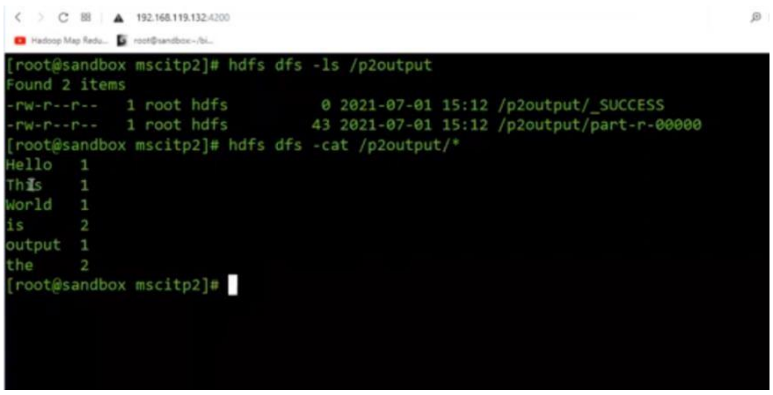
hadoop jar WordCount.jar WordCount /p2/ /p2output



```
[root@sandbox mscitp2]# hadoop jar WordCount.jar WordCount /p2/ /p2output
21/07/01 15:11:44 INFO client.RMProxy: Connecting to ResourceManager at sandbox.hortonworks.com/
172.17.0.2:8032
21/07/01 15:11:44 INFO client.AHSProxy: Connecting to Application History server at sandbox.hort
onworks.com/172.17.0.2:10200
21/07/01 15:11:45 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not per
formed. Implement the Tool interface and execute your application with ToolRunner to remedy this
.
21/07/01 15:11:46 INFO input.FileInputFormat: Total input paths to process : 1
21/07/01 15:11:47 INFO mapreduce.JobSubmitter: number of splits:1
21/07/01 15:11:47 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1625146513303_0004
21/07/01 15:11:48 INFO impl.YarnClientImpl: Submitted application application_1625146513303_0004
21/07/01 15:11:48 INFO mapreduce.Job: The url to track the job: http://sandbox.hortonworks.com:8
088/proxy/application_1625146513303_0004/
21/07/01 15:11:48 INFO mapreduce.Job: Running job: job_1625146513303_0004
```

Print the content of the output file

Command: hdfs dfs -cat /p2ouput/*



```
[root@sandbox mscitp2]# hdfs dfs -ls /p2output
Found 2 items
-rw-r--r--   1 root hdfs          0 2021-07-01 15:12 /p2output/_SUCCESS
-rw-r--r--   1 root hdfs         43 2021-07-01 15:12 /p2output/part-r-00000
[root@sandbox mscitp2]# hdfs dfs -cat /p2output/*
Hello   1
This    1
World   1
is      2
output  1
the     2
[root@sandbox mscitp2]#
```

Ctrl + l to clear the screen.

vi filename.txt= this command will create/ open filename.txt

two modes of vi editor

1) Insert mode – i (press i key)
2) Command mode – esc key

: wq is to save and exit

Practical 3

Implement an MapReduce program that processes a weather dataset.

Java program:

MyMaxMin.java

```
// importing Libraries

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

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

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

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

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.Configuration;

public class MyMaxMin {

// Mapper

        /*MaxTemperatureMapper class is static

        * and extends Mapper abstract class

        * having four Hadoop generics type

        * LongWritable, Text, Text, Text.

        */

        public static class MaxTemperatureMapper extends

                        Mapper<LongWritable, Text, Text, Text> {
```

```
            /**
            * @method map
            * This method takes the input as a text data type.
            * Now leaving the first five tokens, it takes
            * 6th token is taken as temp_max and
            * 7th token is taken as temp_min. Now
            * temp_max > 30 and temp_min < 15 are
            * passed to the reducer.
            */
    // the data in our data set with
    // this value is inconsistent data
    public static final int MISSING = 9999;
    @Override
            public void map(LongWritable arg0, Text Value, Context context)
                            throws IOException, InterruptedException {
            // Convert the single row(Record) to
            // String and store it in String
            // variable name line
            String line = Value.toString();
                    // Check for the empty line
                    if (!(line.length() == 0)) {
                            // from character 6 to 14 we have
                            // the date in our dataset
                            String date = line.substring(6, 14);
                            // similarly we have taken the maximum
                            // temperature from 39 to 45 characters
                            float temp_Max = Float.parseFloat(line.substring(39, 45).trim());
```

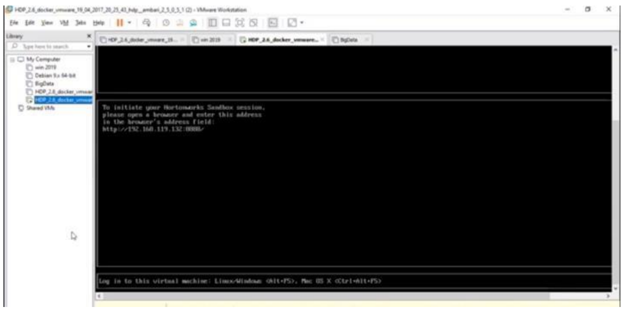
```
                        // similarly we have taken the minimum
                        // temperature from 47 to 53 characters
                        float temp_Min = Float.parseFloat(line.substring(47, 53).trim());
                        // if maximum temperature is
                        // greater than 30, it is a hot day
                        if (temp_Max > 30.0) {
                             // Hot day
                             context.write(new Text("The Day is Hot Day :" + date),
                             new Text(String.valueOf(temp_Max)));
                        }
                        // if the minimum temperature is
                        // less than 15, it is a cold day
                        if (temp_Min < 15) {
                             // Cold day
                             context.write(new Text("The Day is Cold Day :" + date),
                                       new Text(String.valueOf(temp_Min)));
                        }
                   }
          }
 // Reducer
          /*MaxTemperatureReducer class is static
          and extends Reducer abstract class
          having four Hadoop generics type
          Text, Text, Text, Text.
          */
          //The Day is Cold Day :20150101 ,-21.8
```

```
public static class MaxTemperatureReducer extends
        Reducer<Text, Text, Text, Text> {
    /**
    * @method reduce
    * This method takes the input as key and
    * list of values pair from the mapper,
    * it does aggregation based on keys and
    * produces the final context.
    */
    public void reduce(Text Key, Iterator<Text> Values, Context context)
            throws IOException, InterruptedException {
        // putting all the values in
        // temperature variable of type String
        String temperature = Values.next().toString();
        context.write(Key, new Text(temperature));
    }
}
/**
* @method main
* This method is used for setting
* all the configuration properties.
* It acts as a driver for map-reduce
* code.
*/
public static void main(String[] args) throws Exception {
    // reads the default configuration of the
    // cluster from the configuration XML files
    Configuration conf = new Configuration();
```

```
// Initializing the job with the
// default configuration of the cluster
Job job = new Job(conf, "weather example");
// Assigning the driver class name
job.setJarByClass(MyMaxMin.class);
// Key type coming out of mapper
job.setMapOutputKeyClass(Text.class);
// value type coming out of mapper
job.setMapOutputValueClass(Text.class);
// Defining the mapper class name
job.setMapperClass(MaxTemperatureMapper.class);
// Defining the reducer class name
job.setReducerClass(MaxTemperatureReducer.class);
// Defining input Format class which is
// responsible to parse the dataset
// into a key value pair
job.setInputFormatClass(TextInputFormat.class);
// Defining output Format class which is
// responsible to parse the dataset
// into a key value pair
job.setOutputFormatClass(TextOutputFormat.class);
// setting the second argument
// as a path in a path variable
Path OutputPath = new Path(args[1]);
// Configuring the input path
// from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
```

```
            // Configuring the output path from
            // the filesystem into the job
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
            // deleting the context path automatically
            // from hdfs so that we don't have
            // to delete it explicitly
            OutputPath.getFileSystem(conf).delete(OutputPath);
            // exiting the job only if the
            // flag value becomes false
            System.exit(job.waitForCompletion(true) ? 0 : 1);
        }
}
```

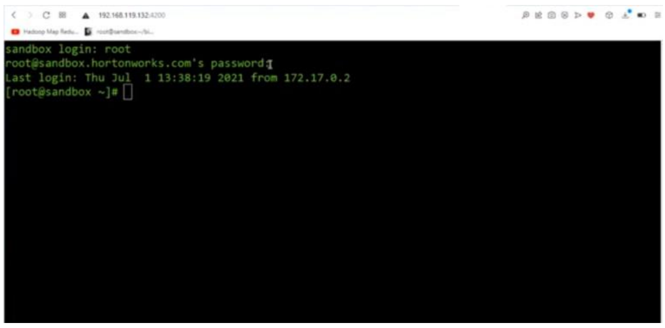
Start the server



```
To initiate your Hortonworks Sandbox session,
please open a browser and enter this address
in the browser's address field:
http://192.168.119.132:8888/
```

Open the terminal with 192.168.119.132/4200

Enter the login: root and the password and enter

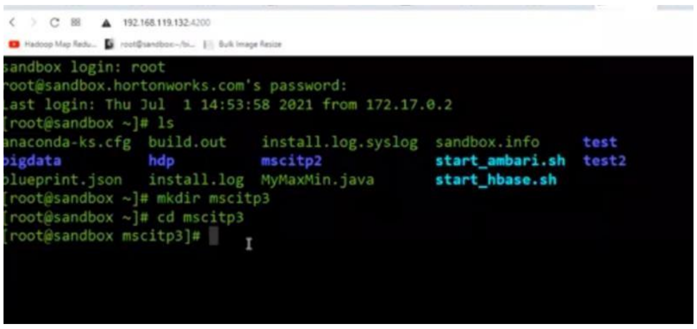


```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Thu Jul  1 13:38:19 2021 from 172.17.0.2
[root@sandbox ~]#
```

Create a folder in local directory.

Command: mkdir mscitp3

Change the directory cd mscitp3



```
sandbox login: root
root@sandbox.hortonworks.com's password:
Last login: Thu Jul  1 14:53:58 2021 from 172.17.0.2
[root@sandbox ~]# ls
anaconda-ks.cfg  build.out    install.log.syslog  sandbox.info     test
bigdata          hdp          mscitp2             start_ambari.sh  test2
blueprint.json   install.log  MyMaxMin.java       start_hbase.sh
[root@sandbox ~]# mkdir mscitp3
[root@sandbox ~]# cd mscitp3
[root@sandbox mscitp3]#
```

Now create input file

Command: cat >> weatherin2.txt

Paste the weather dataset by right clicking on terminal

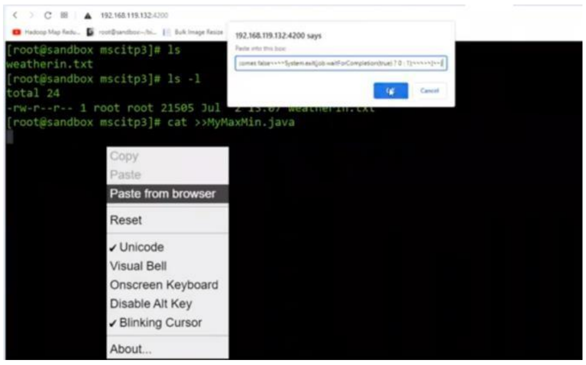
Ctrl d will save the file

Run command ls to see the file.

Create java file

Command: cat >>MyMaxMin.java

Paste the java code and ctrl d to save the file



```
[root@sandbox mscitp3]# ls
weatherin.txt
[root@sandbox mscitp3]# ls -l
total 24
-rw-r--r-- 1 root root 21505 Jul  2 13:07 weatherin.txt
[root@sandbox mscitp3]# cat >>MyMaxMin.java
```

192.168.119.132:4200 says

Paste into this box:

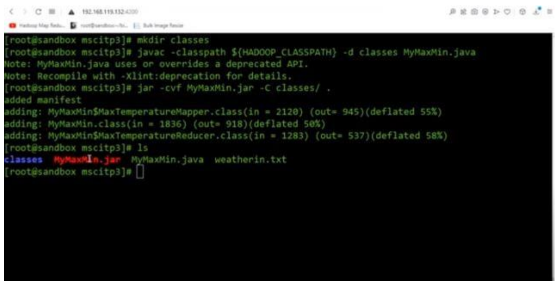
export HADOOP_CLASSPATH=$(hadoop classpath) ////compile and to create jar file

mkdir classes

javac -classpath ${HADOOP_CLASSPATH} -d classes MyMaxMin.java

After compile need to create a jar file

jar -cvf MyMaxMin.jar -C classes/ .



```
[root@sandbox mscitp3]# mkdir classes
[root@sandbox mscitp3]# javac -classpath ${HADOOP_CLASSPATH} -d classes MyMaxMin.java
Note: MyMaxMin.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
[root@sandbox mscitp3]# jar -cvf MyMaxMin.jar -C classes/ .
added manifest
adding: MyMaxMin$MaxTemperatureMapper.class(in = 2120) (out= 945)(deflated 55%)
adding: MyMaxMin.class(in = 1836) (out= 918)(deflated 50%)
adding: MyMaxMin$MaxTemperatureReducer.class(in = 1283) (out= 537)(deflated 58%)
[root@sandbox mscitp3]# ls
classes  MyMaxMin.jar  MyMaxMin.java  weatherin.txt
[root@sandbox mscitp3]# 
```

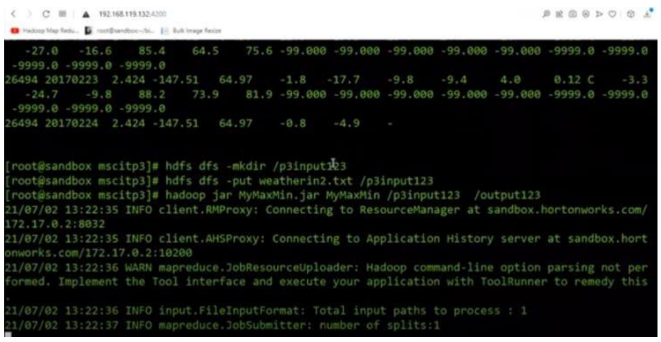
Now, put weatherin.txt in hdfs

Before that create a folder

Command: hdfs dfs -mkdir /p3input123

Then run command: hdfs dfs -put weatherin2.txt /p3input123

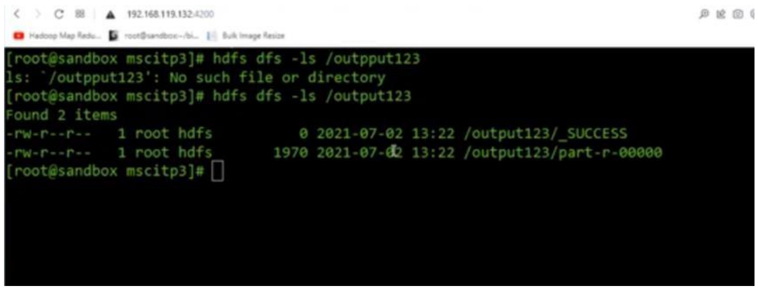
hadoop jar MyMaxMin.jar MyMaxMin /p3inputw /output123



```
   -27.0    -16.6    85.4    64.5    75.6 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0
 -9999.0 -9999.0 -9999.0
26494 20170223  2.424 -147.51   64.97    -1.8   -17.7    -9.8    -9.4     4.0     0.12 C    -3.3
   -24.7     -9.8    88.2    73.9    81.9 -99.000 -99.000 -99.000 -99.000 -99.000 -9999.0 -9999.0
 -9999.0 -9999.0 -9999.0
26494 20170224  2.424 -147.51   64.97    -0.8    -4.9    -


[root@sandbox mscitp3]# hdfs dfs -mkdir /p3input123
[root@sandbox mscitp3]# hdfs dfs -put weatherin2.txt /p3input123
[root@sandbox mscitp3]# hadoop jar MyMaxMin.jar MyMaxMin /p3input123  /output123
21/07/02 13:22:35 INFO client.RMProxy: Connecting to ResourceManager at sandbox.hortonworks.com/
172.17.0.2:8032
21/07/02 13:22:35 INFO client.AHSProxy: Connecting to Application History server at sandbox.hort
onworks.com/172.17.0.2:10200
21/07/02 13:22:36 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not per
formed. Implement the Tool interface and execute your application with ToolRunner to remedy this
.
21/07/02 13:22:36 INFO input.FileInputFormat: Total input paths to process : 1
21/07/02 13:22:37 INFO mapreduce.JobSubmitter: number of splits:1
```

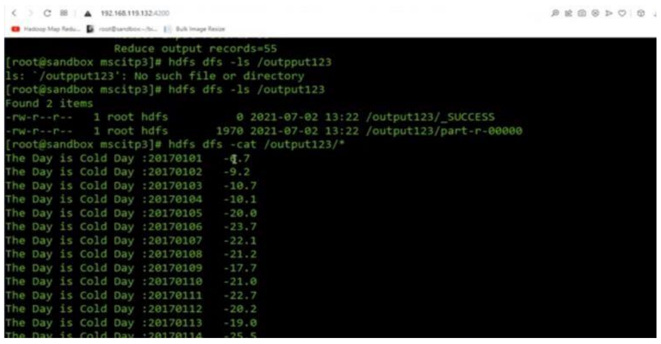
Check outfile is created

Command: hdfs dfs -ls /output123



```
[root@sandbox mscitp3]# hdfs dfs -ls /outpput123
ls: `/outpput123': No such file or directory
[root@sandbox mscitp3]# hdfs dfs -ls /output123
Found 2 items
-rw-r--r--   1 root hdfs          0 2021-07-02 13:22 /output123/_SUCCESS
-rw-r--r--   1 root hdfs       1970 2021-07-02 13:22 /output123/part-r-00000
[root@sandbox mscitp3]# 
```

hdfs dfs -cat /output123/*



```
                Reduce output records=55
[root@sandbox mscitp3]# hdfs dfs -ls /outpput123
ls: `/outpput123': No such file or directory
[root@sandbox mscitp3]# hdfs dfs -ls /output123
Found 2 items
-rw-r--r--   1 root hdfs          0 2021-07-02 13:22 /output123/_SUCCESS
-rw-r--r--   1 root hdfs       1970 2021-07-02 13:22 /output123/part-r-00000
[root@sandbox mscitp3]# hdfs dfs -cat /output123/*
The Day is Cold Day :20170101   -[illegible].7
The Day is Cold Day :20170102   -9.2
The Day is Cold Day :20170103   -10.7
The Day is Cold Day :20170104   -10.1
The Day is Cold Day :20170105   -20.0
The Day is Cold Day :20170106   -23.7
The Day is Cold Day :20170107   -22.1
The Day is Cold Day :20170108   -21.2
The Day is Cold Day :20170109   -17.7
The Day is Cold Day :20170110   -21.0
The Day is Cold Day :20170111   -22.7
The Day is Cold Day :20170112   -20.2
The Day is Cold Day :20170113   -19.0
The Day is Cold Day :20170114   -25.5
```

Practical 4

Aim : Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python
Requirement

a. PyMongo

b. Mongo Database

Step A: Install Mongo database

Step 1) Go to (https://www.mongodb.com/download-center/community) and Download MongoDB Community Server. We will install the 64-bit version for Windows.

Step 2) Once download is complete open the msi file. Click Next in the start up screen

Step 3) 1. Accept the End-User License Agreement 2. Click Next

Step 4) Click on the "complete" button to install all of the components. The custom option can be used to install selective components or if you want to change the location of the installation.

Step 5) 1. Select "Run service as Network Service user". make a note of the data directory, we'll need this later. 2. Click Next

Step 6) Click on the Install button to start the installation

Step 7) Installation begins. Click Next once completed

Step 8) Click on the Finish button to complete the installation

Test Mongodb:

Step 1) Go to " C:\Program Files\MongoDB\Server\4.0\bin" and double click on mongo.exe. Alternatively, you can also click on the MongoDB desktop item

• Create the directory where MongoDB will store it's files. From the command prompt run md \data\db . This is the default location. However, other locations can be specified using the --dbpath parameter. See the Mongo docs for more information. o C:\>md data o C:\md data\db o C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe --dbpath "C:\data"

• Start the mongodb daemon by running C:\mongodb\bin\mongod.exe in the Command Prompt. Or by running, C:\path\to\mongodb\bin\mongod.exe

• Connect to MongoDB using the Mongo shell While the MongoDB daemon is running, from a different Command prompt window run C:\mongodb\bin\mongo.exe

• C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe --dbpath "C:\data"

• C:\Program Files\MongoDB\Server\4.05\bin>mongo.exe

Step B: Install PyMongo

```
C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>python -m pip install pymongo
```

Now you have downloaded and installed a mongoDB driver

Test PyMongo

demo_mongodb_test.py

Program 1: Creating a Database

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mybigdata"]

print(myclient.list_database_names())
```

Output :

```
Mybigdata
```

Progam 2: Creating a Collection

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mybigdata"]

mycol=mydb[“student’]

print(mydb.list_collection_names())
```

Output:

```
Student
```

Progam 3: Insert into Collection

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

 mydb = myclient["mybigdata"]

mycol=mydb[“student’]

mydict={“name”:”Kaushal”, “address”:”Mumbai”}

x=mycol.insert_one(mydict)

print(mydb.list_collection_names())

# insert_one(containing the name(s) and value(s) of each field
```

Output:

Kaushal Mumbai

Program 4: Insert Multiple data into Collection

```
import pymongo

myclient = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = myclient["mybigdata"]

mycol=mydb[“student’]

mylist=[{“name”:”Kaushal”, “address”:”Mumbai”}, {“name”:”A”, “address”:”Mumbai”}, {“name”:”B”, “address”:”Pune”}, {“name”:”C”, “address”:”Pune”},]

x=mycol.insert_many(mylist)
```

Steps and Output

Install mongo db

1. Open cmd

Start the mongodb daemon by running C:\mongodb\bin\mongod.exe in the

Command Prompt. Or by running, C:\path\to\mongodb\bin\mongod.exe

2. Open another cmd

Connect to MongoDB using the Mongo shell While the MongoDB daemon is

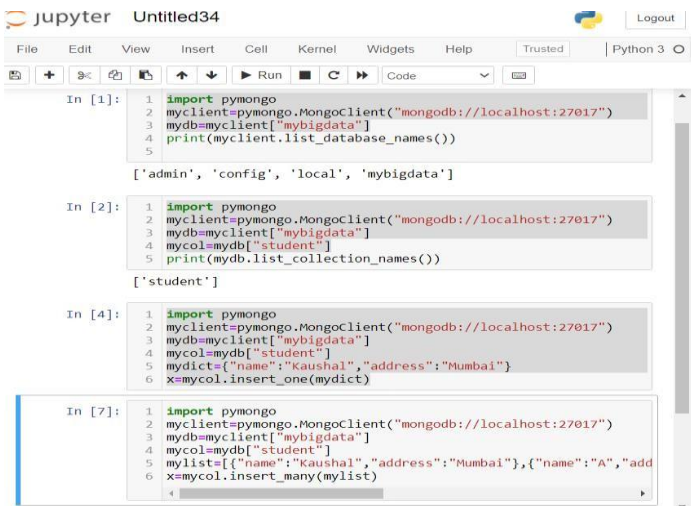
running, from a different Command prompt window

run C:\mongodb\bin\mongo.exe

3. Open Anaconda Prompt

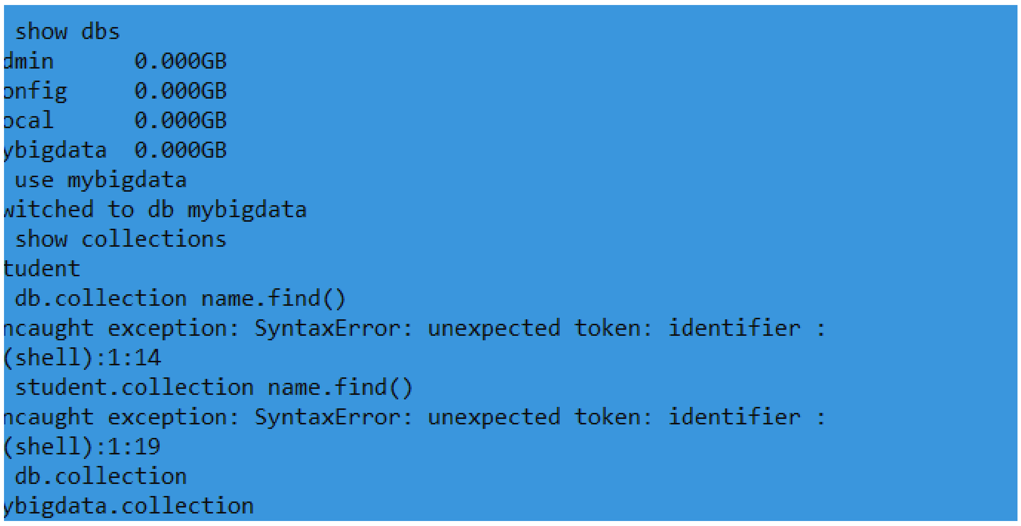
Install pip install pymongo

Then in Anaconda Launch Jupyter Notebook



```
In [1]:
import pymongo
myclient=pymongo.MongoClient("mongodb://localhost:27017")
mydb=myclient["mybigdata"]
print(myclient.list_database_names())

['admin', 'config', 'local', 'mybigdata']

In [2]:
import pymongo
myclient=pymongo.MongoClient("mongodb://localhost:27017")
mydb=myclient["mybigdata"]
mycol=mydb["student"]
print(mydb.list_collection_names())

['student']

In [4]:
import pymongo
myclient=pymongo.MongoClient("mongodb://localhost:27017")
mydb=myclient["mybigdata"]
mycol=mydb["student"]
mydict={"name":"Kaushal","address":"Mumbai"}
x=mycol.insert_one(mydict)

In [7]:
import pymongo
myclient=pymongo.MongoClient("mongodb://localhost:27017")
mydb=myclient["mybigdata"]
mycol=mydb["student"]
mylist=[{"name":"Kaushal","address":"Mumbai"},{"name":"A","add
x=mycol.insert_many(mylist)
```

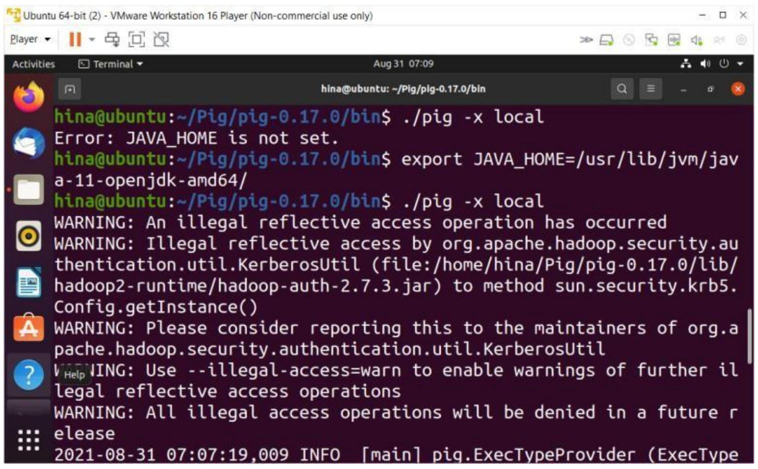
In MongoDB



```
 show dbs
dmin      0.000GB
onfig     0.000GB
ocal      0.000GB
ybigdata  0.000GB
 use mybigdata
witched to db mybigdata
 show collections
tudent
 db.collection name.find()
ncaught exception: SyntaxError: unexpected token: identifier :
(shell):1:14
 student.collection name.find()
ncaught exception: SyntaxError: unexpected token: identifier :
(shell):1:19
 db.collection
ybigdata.collection
```

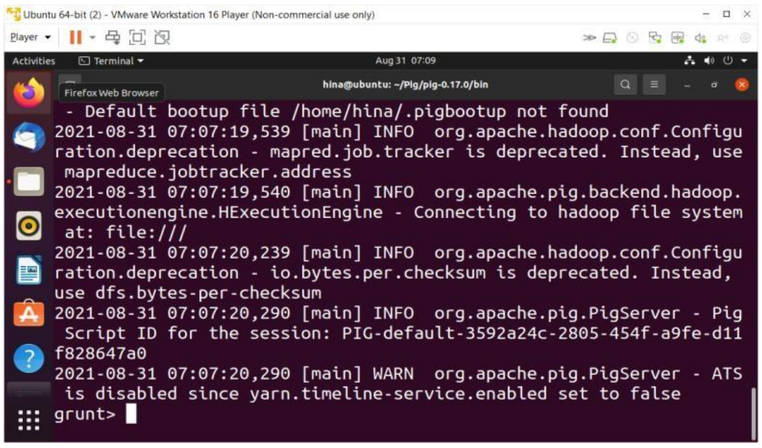
Practical 5

Implement the program in practical 4 using Pig



```
hina@ubuntu:~/Pig/pig-0.17.0/bin$ ./pig -x local
Error: JAVA_HOME is not set.
hina@ubuntu:~/Pig/pig-0.17.0/bin$ export JAVA_HOME=/usr/lib/jvm/jav
a-11-openjdk-amd64/
hina@ubuntu:~/Pig/pig-0.17.0/bin$ ./pig -x local
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.au
thentication.util.KerberosUtil (file:/home/hina/Pig/pig-0.17.0/lib/
hadoop2-runtime/hadoop-auth-2.7.3.jar) to method sun.security.krb5.
Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.a
pache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further il
legal reflective access operations
WARNING: All illegal access operations will be denied in a future r
elease
2021-08-31 07:07:19,009 INFO  [main] pig.ExecTypeProvider (ExecType
```

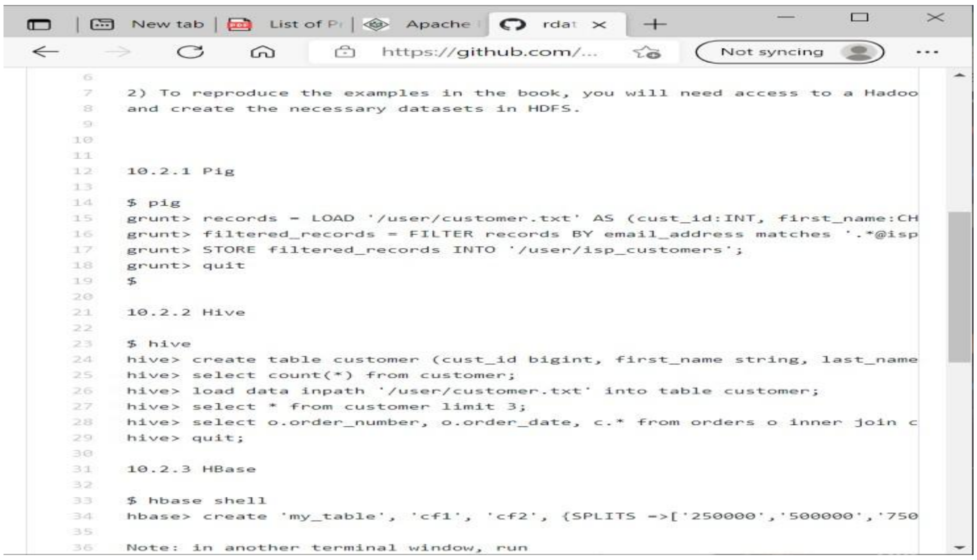
create a directory and get into that directory



```
 - Default bootup file /home/hina/.pigbootup not found
2021-08-31 07:07:19,539 [main] INFO  org.apache.hadoop.conf.Configu
ration.deprecation - mapred.job.tracker is deprecated. Instead, use
 mapreduce.jobtracker.address
2021-08-31 07:07:19,540 [main] INFO  org.apache.pig.backend.hadoop.
executionengine.HExecutionEngine - Connecting to hadoop file system
 at: file:///
2021-08-31 07:07:20,239 [main] INFO  org.apache.hadoop.conf.Configu
ration.deprecation - io.bytes.per.checksum is deprecated. Instead,
use dfs.bytes-per-checksum
2021-08-31 07:07:20,290 [main] INFO  org.apache.pig.PigServer - Pig
 Script ID for the session: PIG-default-3592a24c-2805-454f-a9fe-d11
f828647a0
2021-08-31 07:07:20,290 [main] WARN  org.apache.pig.PigServer - ATS
 is disabled since yarn.timeline-service.enabled set to false
grunt>
```

Create a file

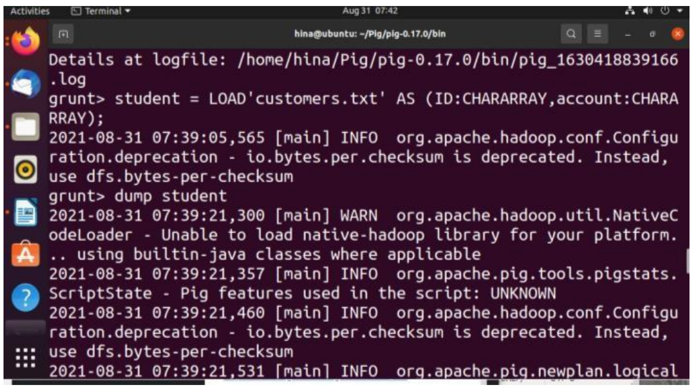
Command: cat >>student.txt



```
2) To reproduce the examples in the book, you will need access to a Hadoo
and create the necessary datasets in HDFS.



10.2.1 Pig

$ pig
grunt> records = LOAD '/user/customer.txt' AS (cust_id:INT, first_name:CH
grunt> filtered_records = FILTER records BY email_address matches '.*@isp
grunt> STORE filtered_records INTO '/user/isp_customers';
grunt> quit
$

10.2.2 Hive

$ hive
hive> create table customer (cust_id bigint, first_name string, last_name
hive> select count(*) from customer;
hive> load data inpath '/user/customer.txt' into table customer;
hive> select * from customer limit 3;
hive> select o.order_number, o.order_date, c.* from orders o inner join c
hive> quit;

10.2.3 HBase

$ hbase shell
hbase> create 'my_table', 'cf1', 'cf2', {SPLITS =>['250000','500000','750

Note: in another terminal window, run
```

Create a program file

script start
student = LOAD 'student.txt' USING PigStorage(',')
as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);
student_order = ORDER student BY age DESC;
student_limit = LIMIT student_order 4;
Dump student_limit;
script end

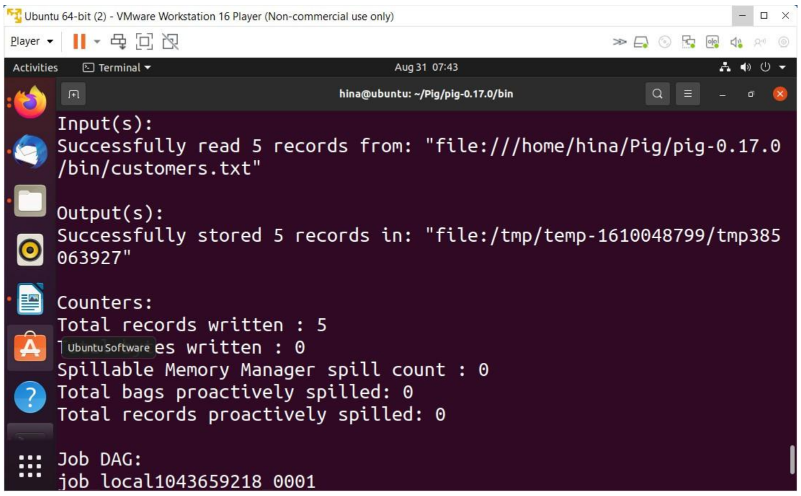


```
Details at logfile: /home/hina/Pig/pig-0.17.0/bin/pig_1630418839166
.log
grunt> student = LOAD'customers.txt' AS (ID:CHARARRAY,account:CHARA
RRAY);
2021-08-31 07:39:05,565 [main] INFO  org.apache.hadoop.conf.Configu
ration.deprecation - io.bytes.per.checksum is deprecated. Instead,
use dfs.bytes-per-checksum
grunt> dump student
2021-08-31 07:39:21,300 [main] WARN  org.apache.hadoop.util.NativeC
odeLoader - Unable to load native-hadoop library for your platform.
.. using builtin-java classes where applicable
2021-08-31 07:39:21,357 [main] INFO  org.apache.pig.tools.pigstats.
ScriptState - Pig features used in the script: UNKNOWN
2021-08-31 07:39:21,460 [main] INFO  org.apache.hadoop.conf.Configu
ration.deprecation - io.bytes.per.checksum is deprecated. Instead,
use dfs.bytes-per-checksum
2021-08-31 07:39:21,531 [main] INFO  org.apache.pig.newplan.logical
```

Upload student on hdfs

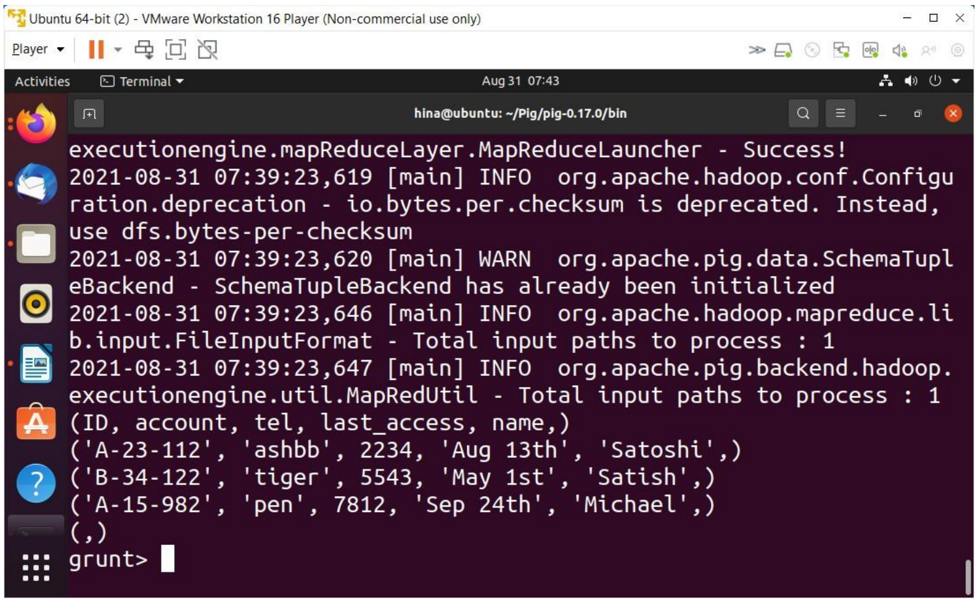
Command: hdfs dfs -put student.txt /user/root/

Run the pig program



```
Input(s):
Successfully read 5 records from: "file:///home/hina/Pig/pig-0.17.0
/bin/customers.txt"

Output(s):
Successfully stored 5 records in: "file:/tmp/temp-1610048799/tmp385
063927"

Counters:
Total records written : 5
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_local1043659218_0001
```

OUTPUT:



```
executionengine.mapReduceLayer.MapReduceLauncher - Success!
2021-08-31 07:39:23,619 [main] INFO  org.apache.hadoop.conf.Configu
ration.deprecation - io.bytes.per.checksum is deprecated. Instead,
use dfs.bytes-per-checksum
2021-08-31 07:39:23,620 [main] WARN  org.apache.pig.data.SchemaTupl
eBackend - SchemaTupleBackend has already been initialized
2021-08-31 07:39:23,646 [main] INFO  org.apache.hadoop.mapreduce.li
b.input.FileInputFormat - Total input paths to process : 1
2021-08-31 07:39:23,647 [main] INFO  org.apache.pig.backend.hadoop.
executionengine.util.MapRedUtil - Total input paths to process : 1
(ID, account, tel, last_access, name,)
('A-23-112', 'ashbb', 2234, 'Aug 13th', 'Satoshi',)
('B-34-122', 'tiger', 5543, 'May 1st', 'Satish',)
('A-15-982', 'pen', 7812, 'Sep 24th', 'Michael',)
(,)
grunt>
```

Practical 6

Configure the Hive and implement the application in Hive.

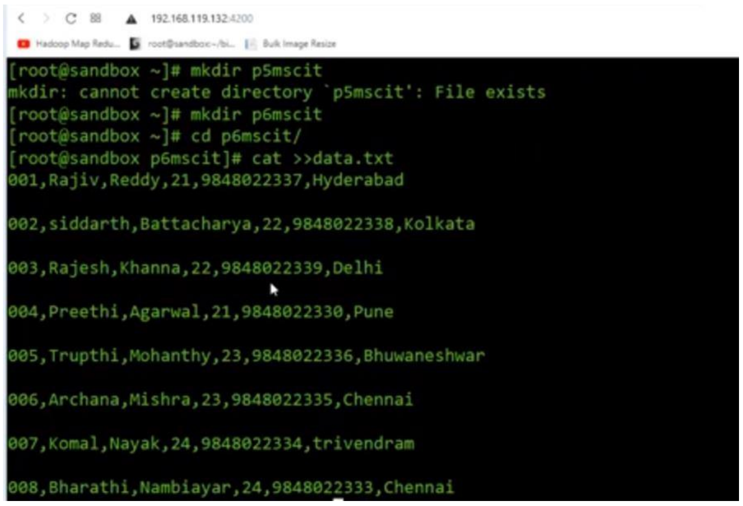
Create a directory and get into that directory

Command: mkdir p6mscit

Create a file

Command: cat >>data.txt

Right click and paste the text



```
[root@sandbox ~]# mkdir p5mscit
mkdir: cannot create directory `p5mscit': File exists
[root@sandbox ~]# mkdir p6mscit
[root@sandbox ~]# cd p6mscit/
[root@sandbox p6mscit]# cat >>data.txt
001,Rajiv,Reddy,21,9848022337,Hyderabad

002,siddarth,Battacharya,22,9848022338,Kolkata

003,Rajesh,Khanna,22,9848022339,Delhi

004,Preethi,Agarwal,21,9848022330,Pune

005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar

006,Archana,Mishra,23,9848022335,Chennai

007,Komal,Nayak,24,9848022334,trivendram

008,Bharathi,Nambiayar,24,9848022333,Chennai
```

Remove the space with vi editor

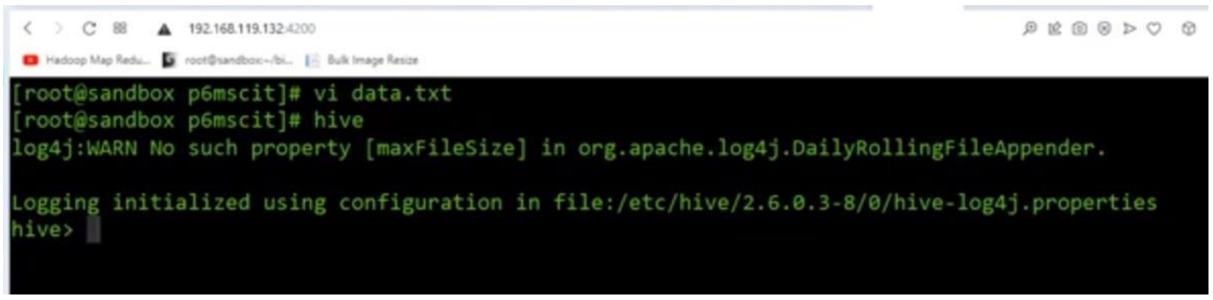
Command: vi student.txt and press i for insert mode

After editing: wq and enter

Print the content and see the text

Now start the hive terminal

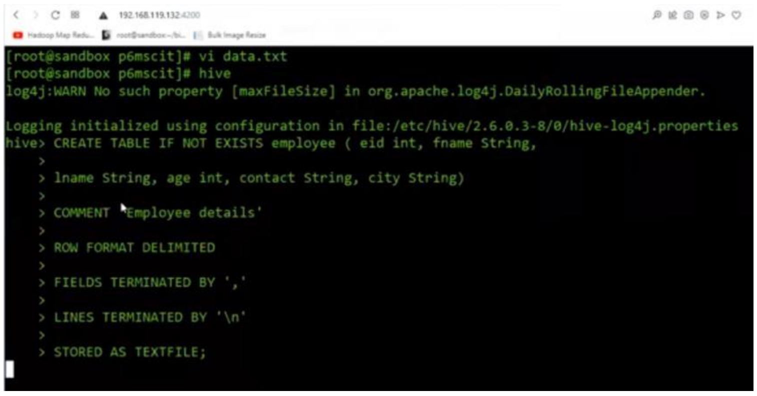
Command: hive



```
[root@sandbox p6mscit]# vi data.txt
[root@sandbox p6mscit]# hive
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.

Logging initialized using configuration in file:/etc/hive/2.6.0.3-8/0/hive-log4j.properties
hive>
```

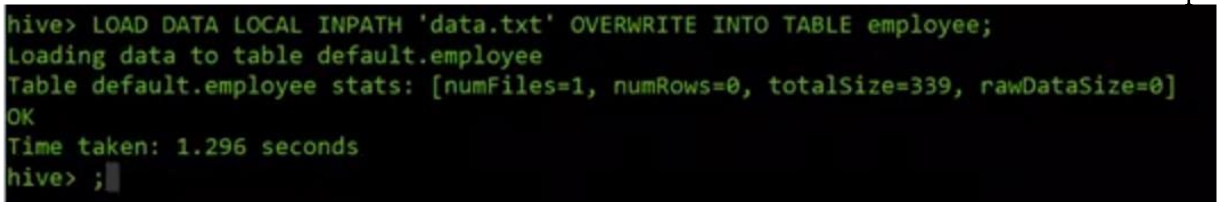
Copy paste below command on hive and enter

create table
CREATE TABLE IF NOT EXISTS employee (eid int, fname String,
lname String, age int, contact String, city String)
COMMENT 'Employee details'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;



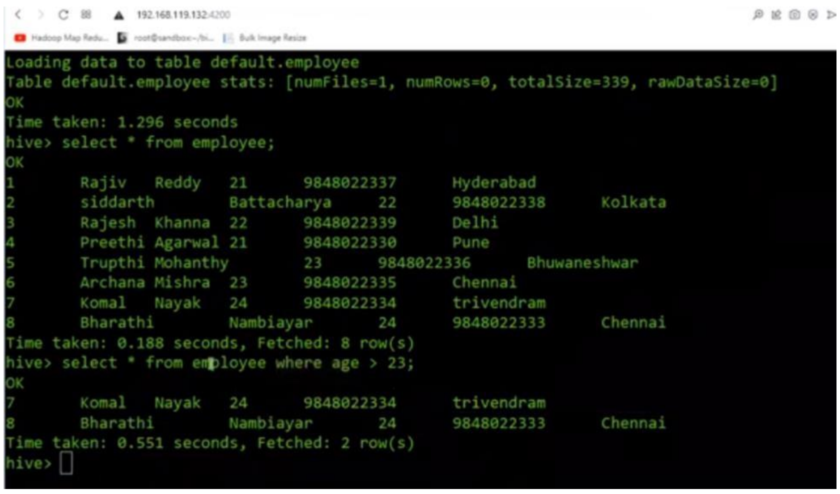
```
[root@sandbox p6mscit]# vi data.txt
[root@sandbox p6mscit]# hive
log4j:WARN No such property [maxFileSize] in org.apache.log4j.DailyRollingFileAppender.

Logging initialized using configuration in file:/etc/hive/2.6.0.3-8/0/hive-log4j.properties
hive> CREATE TABLE IF NOT EXISTS employee ( eid int, fname String,
    >
    > lname String, age int, contact String, city String)
    >
    > COMMENT 'Employee details'
    >
    > ROW FORMAT DELIMITED
    >
    > FIELDS TERMINATED BY ','
    >
    > LINES TERMINATED BY '\n'
    >
    > STORED AS TEXTFILE;
```

Run command: LOAD DATA LOCAL INPATH 'data.txt' OVERWRITE INTO TABLE employee;



```
hive> LOAD DATA LOCAL INPATH 'data.txt' OVERWRITE INTO TABLE employee;
Loading data to table default.employee
Table default.employee stats: [numFiles=1, numRows=0, totalSize=339, rawDataSize=0]
OK
Time taken: 1.296 seconds
hive> ;
```

Run the command like select * from employee;



```
Loading data to table default.employee
Table default.employee stats: [numFiles=1, numRows=0, totalSize=339, rawDataSize=0]
OK
Time taken: 1.296 seconds
hive> select * from employee;
OK
1       Rajiv   Reddy   21      9848022337      Hyderabad
2       siddarth        Battacharya     22      9848022338      Kolkata
3       Rajesh  Khanna  22      9848022339      Delhi
4       Preethi Agarwal 21      9848022330      Pune
5       Trupthi Mohanthy        23      9848022336      Bhuwaneshwar
6       Archana Mishra  23      9848022335      Chennai
7       Komal   Nayak   24      9848022334      trivendram
8       Bharathi        Nambiayar       24      9848022333      Chennai
Time taken: 0.188 seconds, Fetched: 8 row(s)
hive> select * from employee where age > 23;
OK
7       Komal   Nayak   24      9848022334      trivendram
8       Bharathi        Nambiayar       24      9848022333      Chennai
Time taken: 0.551 seconds, Fetched: 2 row(s)
hive>
```

Practical 8

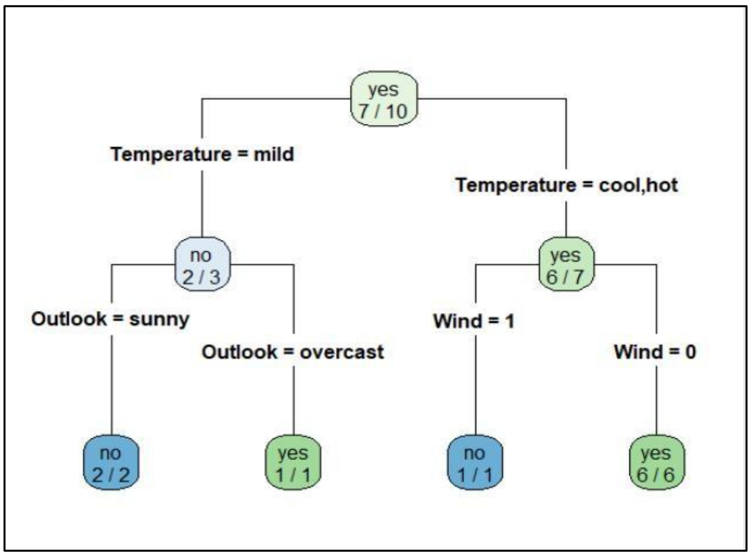
Implement the following:

a. **Implement Decision tree classification techniques**

Code:

```
library("rpart") # load libraries

library("rpart.plot")

play_decision <- read.table ( "C:/2020-21/IDOL/Big Data/Practical/DTdata.csv", header=TRUE, sep=",")

play_decision

summary(play_decision)

fit <- rpart(Play ~ Outlook+ Temperature +Humidity+ Wind,

            method="class",

            data=play_decision,

            control=rpart.control(minsplit=1),

            parms=list(split='information'))

summary(fit)

rpart.plot(fit, type=4, extra=1)

rpart.plot(fit, type=4, extra=2, clip.right.labs=FALSE,

          varlen=0, faclen=0)

newdata <- data.frame(Outlook="rainy", Temperature="mild",

                  Humidity="high", Wind=FALSE)

newdata

#predict(object, newdata = list(),type= c("vector", "prob", "Class","matrix"))

predict(fit,newdata=newdata,type="prob")

predict(fit,newdata=newdata,type="class")
```

Output:



b. Implement SVM classification techniques

Code:

```
library(e1071)

# read the data into a table from the file

sample <- read.table("C:/2020-21/IDOL/Big Data/Practical/sample2.csv",header=TRUE,sep=",")

# define the data frames for the NB classifier

traindata<-as.data.frame(sample[1:14,])

testdata<-as.data.frame(sample[15,])

traindata

testdata

tprior<-table(traindata$Enrolls)

tprior

tprior<- tprior/sum(tprior)

tprior

ageCounts <- table(traindata[,c("Enrolls","Age")])

ageCounts
```

```
ageCounts <- ageCounts/rowSums(ageCounts)


ageCounts

incomeCounts <- table(traindata[,c("Enrolls", "Income")])

incomeCounts <- incomeCounts/rowSums(incomeCounts)

incomeCounts

jsCounts <- table(traindata[,c("Enrolls","JobSatisfaction")])

jsCounts <- jsCounts/rowSums(jsCounts)

jsCounts

desireCounts <- table(traindata[,c("Enrolls","Desire")])

desireCounts <- desireCounts/rowSums(desireCounts)

desireCounts

#prob_yes<-ageCounts["Yes",testdata[,c("Age")]]*

#incomeCounts["Yes",testdata[,c("Income")]]*

#jsCounts["Yes",testdata[,c("JobSatisfaction")]]*

#desireCounts["Yes",testdata[,c("Desire")]]*tprior["Yes"]

model <- naiveBayes(Enrolls ~Age+Income+JobSatisfaction+Desire,traindata)

model

results<- predict(model,testdata)

results

model1=naiveBayes(Enrolls ~., traindata, laplace=.01)

model1

results1 <- predict(model1,testdata)

results1
```

Output:

traindata

```
Age Income JobSatisfaction    Desire Enrolls
1      <=30   High             No      Fair    No
2      <=30   High             No Excellent     No
3  31 to 40   High             No      Fair    Yes
```

```
4      >40 Medium            No      Fair    Yes
5      >40    Low           Yes      Fair    Yes
6      >40    Low           Yes Excellent     No
7  31 to 40    Low           Yes Excellent    Yes
8      <=30 Medium           No      Fair     No
9      <=30    Low          Yes      Fair    Yes
10      >40 Medium           Yes      Fair    Yes
11     <=30 Medium           Yes Excellent    Yes
12 31 to 40 Medium            No Excellent    Yes
13 31 to 40   High           Yes      Fair    Yes
14      >40 Medium            No Excellent     No
```

```
> testdata
   Age Income JobSatisfaction Desire Enrolls
   15 <=30      Medium             Yes  Fair
> tprior<-table(traindata$Enrolls)
> tprior

 No Yes
  5   9

> tprior<- tprior/sum(tprior)
> tprior

       No       Yes
0.3571429 0.6428571

> ageCounts <- table(traindata[,c("Enrolls","Age")])
> ageCounts

       Age
Enrolls <=30 >40 31 to 40
    No     3   2        0
    Yes    2   3        4

> ageCounts <- ageCounts/rowSums(ageCounts)
> ageCounts

       Age
Enrolls      <=30       >40  31 to 40
    No  0.6000000 0.4000000 0.0000000
    Yes 0.2222222 0.3333333 0.4444444

> incomeCounts <- table(traindata[,c("Enrolls", "Income")])
> incomeCounts <- incomeCounts/rowSums(incomeCounts)
> incomeCounts

       Income
Enrolls      High       Low    Medium
    No  0.4000000 0.2000000 0.4000000
    Yes 0.2222222 0.3333333 0.4444444
```

```
> jsCounts <- table(traindata[,c("Enrolls","JobSatisfaction")])
> jsCounts <- jsCounts/rowSums(jsCounts)

> jsCounts

       JobSatisfaction
Enrolls        No       Yes
    No   0.8000000 0.2000000
    Yes  0.3333333 0.6666667

> desireCounts <- table(traindata[,c("Enrolls","Desire")])
> desireCounts <- desireCounts/rowSums(desireCounts)
> desireCounts

       Desire
Enrolls Excellent      Fair
    No   0.6000000 0.4000000
    Yes  0.3333333 0.6666667

> #prob_yes<-ageCounts["Yes",testdata[,c("Age")]]*
> #incomeCounts["Yes",testdata[,c("Income")]]*
> #jsCounts["Yes",testdata[,c("JobSatisfaction")]]*
> #desireCounts["Yes",testdata[,c("Desire")]]*tprior["Yes"]
> model <- naiveBayes(Enrolls ~Age+Income+JobSatisfaction+Desire,traindata)
> model

Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
       No       Yes
0.3571429 0.6428571

Conditional probabilities:
      Age
Y          <=30       >40  31 to 40
  No   0.6000000 0.4000000 0.0000000
  Yes  0.2222222 0.3333333 0.4444444

      Income
Y          High       Low    Medium
  No   0.4000000 0.2000000 0.4000000
  Yes  0.2222222 0.3333333 0.4444444

      JobSatisfaction
Y            No       Yes
  No   0.8000000 0.2000000
  Yes  0.3333333 0.6666667

      Desire
Y     Excellent      Fair
```

```
  No   0.6000000 0.4000000
  Yes  0.3333333 0.6666667


> results<- predict(model,testdata)
> results
factor(0)
Levels:
> model1=naiveBayes(Enrolls ~., traindata, laplace=.01)
> model1

Naive Bayes Classifier for Discrete Predictors
Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
       No       Yes
0.3571429 0.6428571

Conditional probabilities:
     Age
Y         <=30       >40  31 to 40
  No   0.6020000 0.4020000 0.0020000
  Yes  0.2233333 0.3344444 0.4455556

     Income
Y          High       Low    Medium
  No   0.4020000 0.2020000 0.4020000
  Yes  0.2233333 0.3344444 0.4455556

     JobSatisfaction
Y            No       Yes
  No   0.8020000 0.2020000
  Yes  0.3344444 0.6677778

     Desire
Y      Excellent      Fair
  No   0.6020000 0.4020000
  Yes  0.3344444 0.6677778

> results1 <- predict(model1,testdata)
> results1
factor(0)
Levels:
```

Practical 9a Logistic Regression

A wireless telecommunications company wants to estimate the probability that a customer will churn (switch to a different company) in the next six months. With a reasonably accurate prediction of a person's likelihood of churning, the sales and marketing groups can attempt to retain the customer by offering various incentives. Data on 8,000 current and prior customers was obtained. The variables collected for each customer follow:

o Age (years)

o Married(true/false)

o Duration as a customer (years)

o Churned_ con tact s (count)-Number ofthe customer's contacts that have churned (count)

o Churned (true/false)-Whether the customer churned

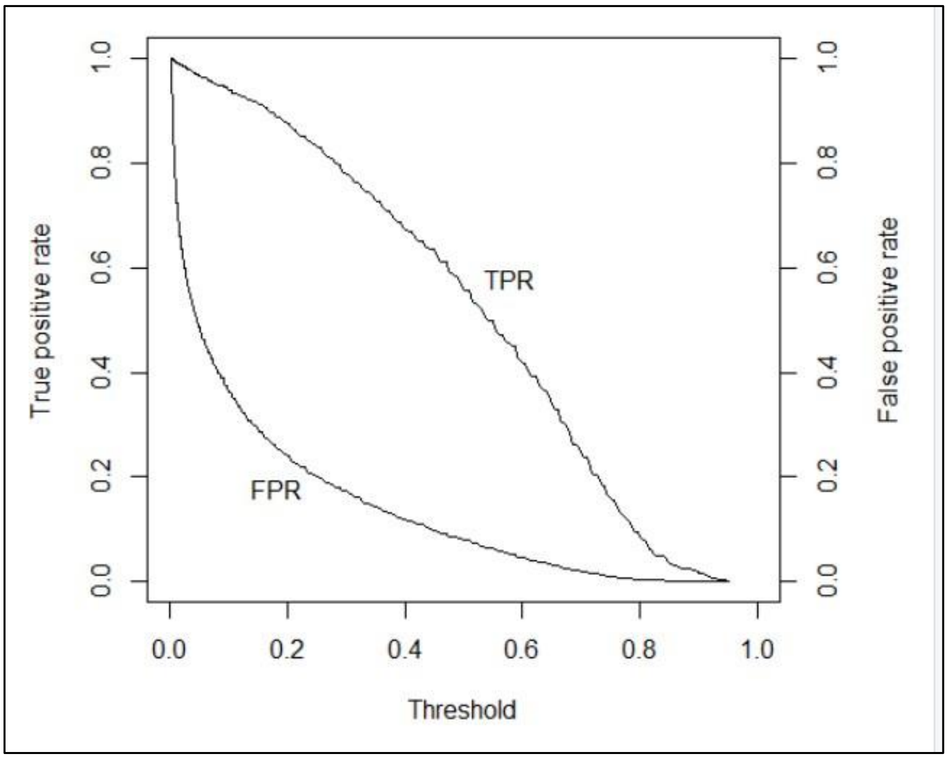
Code

```
churn_input = as.data.frame(read.csv("C:/2020-21/IDOL/Big Data/Practical/churn.csv")  )

head(churn_input)

sum(churn_input$Churned)

Churn_logistic1 <- glm (Churned~Age + Married + Cust_years + Churned_contacts,

                    data=churn_input, family=binomial(link="logit"))

summary(Churn_logistic1)

Churn_logistic2 <- glm (Churned~Age + Married +  Churned_contacts,

               data=churn_input, family=binomial(link="logit"))

summary(Churn_logistic2)

Churn_logistic3 <- glm (Churned~Age + Churned_contacts,

               data=churn_input, family=binomial(link="logit"))

summary(Churn_logistic3)

# Deviance and the Log Likelihood Ratio Test

# Using the residual deviances from Churn_logistics2 and Churn_logistic3

# determine the signficance of the computed test statistic

summary(Churn_logistic2)
```

```
pchisq(.9 , 1, lower=FALSE)

# Receiver Operating Characteristic (ROC) Curve

#install.packages("ROCR")      #install, if necessary

library(ROCR)

pred = predict(Churn_logistic3, type="response")

predObj = prediction(pred, churn_input$Churned )

rocObj = performance(predObj, measure="tpr", x.measure="fpr")

aucObj = performance(predObj, measure="auc")

plot(rocObj, main = paste("Area under the curve:", round(aucObj@y.values[[1]] ,4)))

# extract the alpha(threshold), FPR, and TPR values from rocObj

alpha <- round(as.numeric(unlist(rocObj@alpha.values)),4)

fpr <- round(as.numeric(unlist(rocObj@x.values)),4)

tpr <- round(as.numeric(unlist(rocObj@y.values)),4)

# adjust margins and plot TPR and FPR

par(mar = c(5,5,2,5))

plot(alpha,tpr, xlab="Threshold", xlim=c(0,1), ylab="True positive rate", type="l")

par(new="True")

plot(alpha,fpr, xlab="", ylab="", axes=F, xlim=c(0,1), type="l" )

axis(side=4)

mtext(side=4, line=3, "False positive rate")

text(0.18,0.18,"FPR")

text(0.58,0.58,"TPR")

i <- which(round(alpha,2) == .5)

paste("Threshold=" , (alpha[i]) , " TPR=" , tpr[i] , " FPR=" , fpr[i])


i <- which(round(alpha,2) == .15)

paste("Threshold=" , (alpha[i]) , " TPR=" , tpr[i] , " FPR=" , fpr[i])
```

Output:

```
# Deviance and the Log Likelihood Ratio Test
>
> # Using the residual deviances from Churn_logistics2 and Churn_logistic3
> # determine the signficance of the computed test statistic
> summary(Churn_logistic2)

Call:
glm(formula = Churned ~ Age + Married + Churned_contacts, family = binomial(link = "logit"),
    data = churn_input)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4476  -0.5178  -0.1972  -0.0723   3.3776

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       3.472062   0.132107  26.282   <2e-16 ***
Age              -0.156635   0.004088 -38.318   <2e-16 ***
Married           0.066430   0.068299   0.973    0.331
Churned_contacts  0.381909   0.027302  13.988   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8387.3  on 7999  degrees of freedom
Residual deviance: 5358.3  on 7996  degrees of freedom
AIC: 5366.3

Number of Fisher Scoring iterations: 6

> pchisq(.9 , 1, lower=FALSE)
[1] 0.3427817

> # Receiver Operating Characteristic (ROC) Curve
>
> #install.packages("ROCR")      #install, if necessary
> library(ROCR)
>
> pred = predict(Churn_logistic3, type="response")
> predObj = prediction(pred, churn_input$Churned )
>
> rocObj = performance(predObj, measure="tpr", x.measure="fpr")
> aucObj = performance(predObj, measure="auc")
>
> plot(rocObj, main = paste("Area under the curve:", round(aucObj@y.values[[1]] ,4)))
>
> # extract the alpha(threshold), FPR, and TPR values from rocObj
> alpha <- round(as.numeric(unlist(rocObj@alpha.values)),4)
> fpr <- round(as.numeric(unlist(rocObj@x.values)),4)
> tpr <- round(as.numeric(unlist(rocObj@y.values)),4)
>
> # adjust margins and plot TPR and FPR
> par(mar = c(5,5,2,5))
> plot(alpha,tpr, xlab="Threshold", xlim=c(0,1), ylab="True positive rate", type="l")
> par(new="True")
> plot(alpha,fpr, xlab="", ylab="", axes=F, xlim=c(0,1), type="l" )
> axis(side=4)
> mtext(side=4, line=3, "False positive rate")
> text(0.18,0.18,"FPR")
> text(0.58,0.58,"TPR")
>
> i <- which(round(alpha,2) == .5)
> paste("Threshold=" , (alpha[i]) , " TPR=" , tpr[i] , " FPR=" , fpr[i])

[1] "Threshold= 0.5004  TPR= 0.5571  FPR= 0.0793"
>
> i <- which(round(alpha,2) == .15)
> paste("Threshold=" , (alpha[i]) , " TPR=" , tpr[i] , " FPR=" , fpr[i])

[1] "Threshold= 0.1543  TPR= 0.9116  FPR= 0.2869" "Threshold= 0.1518  TPR= 0.9122  FPR= 0.2875"
[3] "Threshold= 0.1479  TPR= 0.9145  FPR= 0.2942" "Threshold= 0.1455  TPR= 0.9174  FPR= 0.2981"
```



Practical 10

Solve the Following:

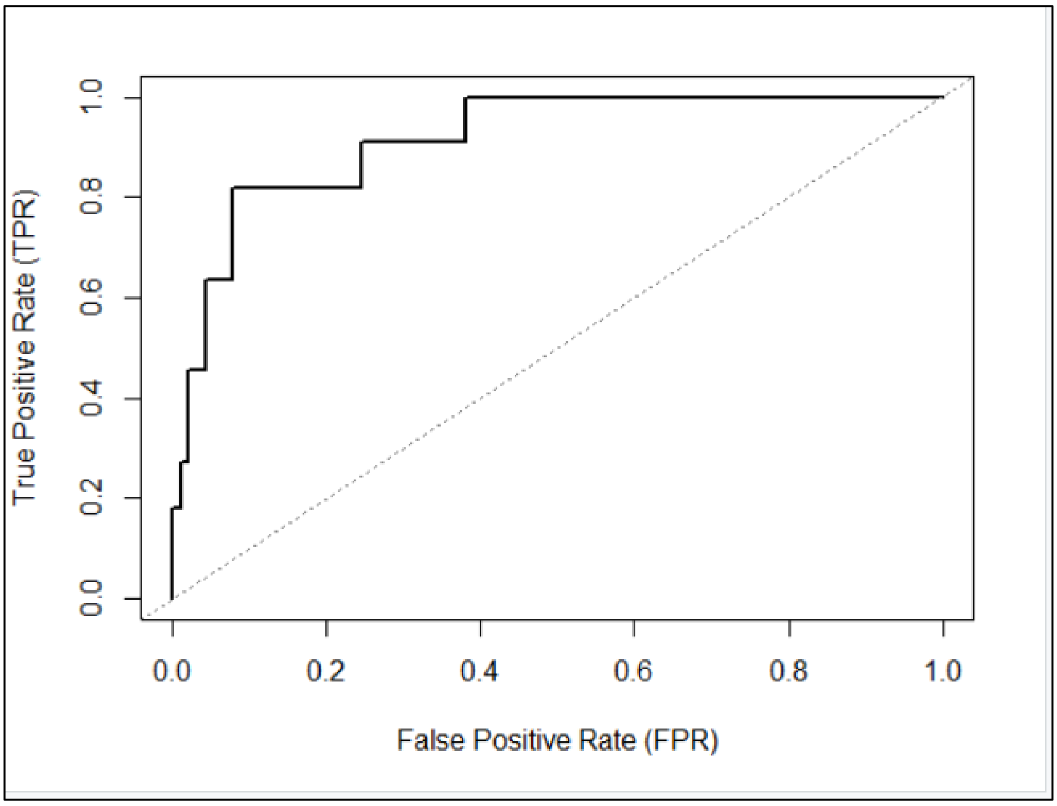
a. CLASSIFICATION MODEL a. Install relevant package for classification. b. Choose classifier for classification problem. c. Evaluate the performance of classifier.

Code:

```
library(ROCR)

library(e1071)

## Read the data

setwd("C:/2020-21/IDOL/Big Data/Practical")

# training set

banktrain <- read.table("bank-sample.csv",header=TRUE,sep=",")

# drop a few columns

drops <- c("balance", "day", "campaign", "pdays", "previous", "month")

banktrain <- banktrain [,!(names(banktrain) %in% drops)]

# testing set

banktest <- read.table("bank-sample-test.csv",header=TRUE,sep=",")

banktest <- banktest [,!(names(banktest) %in% drops)]

# build the naïve Bayes classifier

nb_model <- naiveBayes(subscribed~., data=banktrain)

# perform on the testing set

nb_prediction <- predict(nb_model,

                    # remove column "subscribed"

                     banktest[,-ncol(banktest),  type='raw')

score <- nb_prediction[, c("yes")]

actual_class <- banktest$subscribed == 'yes'

pred <- prediction(score, actual_class)

perf <- performance(pred, "tpr", "fpr")

plot(perf, lwd=2, xlab="False Positive Rate (FPR)",
 ylab="True Positive Rate (TPR)")

abline(a=0, b=1, col="gray50", lty=3)

## corresponding AUC score

auc <- performance(pred, "auc")

auc <- unlist(slot(auc, "y.values"))

auc
```

Output:

```
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Microsoft R Open 4.0.2
The enhanced R distribution from Microsoft
Microsoft packages Copyright (C) 2020 Microsoft Corporation

Using the Intel MKL for parallel mathematical computing (using 4 cores).

Default CRAN mirror snapshot taken on 2020-07-16.
See: https://mran.microsoft.com/.

[Workspace loaded from ~/ClassificaBAyes/.RData]

Loading required package: ROCR
> library(ROCR)
> library(e1071)
> ## Read the data
> setwd("C:/2020-21/IDOL/Big Data/Practical")
> # training set
> banktrain <- read.table("bank-sample.csv",header=TRUE,sep=",")
> # drop a few columns
> drops <- c("balance", "day", "campaign", "pdays", "previous", "month")
> banktrain <- banktrain [,!(names(banktrain) %in% drops)]
> # testing set
> banktest <- read.table("bank-sample-test.csv",header=TRUE,sep=",")
> banktest <- banktest [,!(names(banktest) %in% drops)]
> # build the naïve Bayes classifier
> nb_model <- naiveBayes(subscribed~.,
+                       data=banktrain)
> # perform on the testing set
> nb_prediction <- predict(nb_model,
+                         # remove column "subscribed"
+                         banktest[,-ncol(banktest)],
+                         type='raw')
> score <- nb_prediction[, c("yes")]
> actual_class <- banktest$subscribed == 'yes'
> pred <- prediction(score, actual_class)
> perf <- performance(pred, "tpr", "fpr")
> plot(perf, lwd=2, xlab="False Positive Rate (FPR)",
+      ylab="True Positive Rate (TPR)")
> abline(a=0, b=1, col="gray50", lty=3)
> ## corresponding AUC score
> auc <- performance(pred, "auc")
> auc <- unlist(slot(auc, "y.values"))
> auc
[1] 0.9152196
```



10. b CLUSTERING MODEL

a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.

Code:

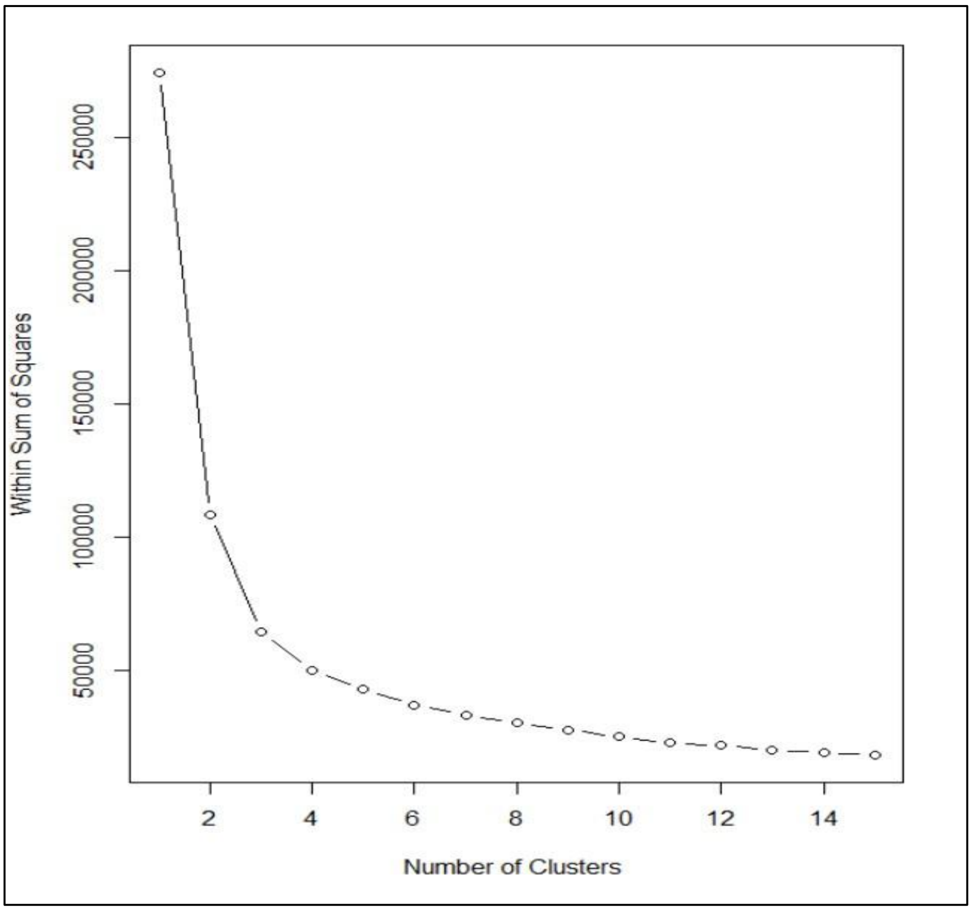
```
library (plyr)

library(ggplot2)

library(cluster)

library(lattice)

library(graphics)

library(grid)

#library(gridExtra)

#library("cowplot")

grade_input = as.data.frame(read.csv("C:/2020-21/IDOL/Big Data/grades_km_input.csv"))

kmdata_orig = as.matrix(grade_input[,c("Student","English","Math","Science")])

kmdata <- kmdata_orig[,2:4]

kmdata[1:10,]

wss <- numeric(15)

for (k in 1:15 ) wss[k] <- sum (kmeans(kmdata, centers=k, nstart=25)$withinss)

plot(1:15, wss, type="b", xlab="Number of Clusters " , ylab="Within Sum of Squares")

km = kmeans(kmdata,3, nstart=25)

km

c( wss[3] , sum(km$withinss) )

df= as.data.frame(kmdata_orig[,2:4])

df$cluster = factor(km$cluster)

centers=as.data.frame(km$centers)

g1= ggplot(data=df, aes(x=English, y=Math, color=cluster )) +

  geom_point() + theme(legend.position="right") +

  geom_point(data=centers,

             aes(x=English,y=Math, color=as.factor(c(1,2,3))), size=10, alpha=.3, show.legend=FALSE)


g2 =ggplot(data=df, aes(x=English, y=Science, color=cluster )) +

  geom_point() +
  geom_point(data=centers,
            aes(x=English,y=Science, color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)


g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
  geom_point() +
  geom_point(data=centers,
            aes(x=Math,y=Science, color=as.factor(c(1,2,3))), size=10, alpha=.3, show.legend=FALSE)
tmp = ggplot_gtable(ggplot_build(g1))
grid.arrange(arrangeGrob(g1 + theme(legend.position="none"), g2 + theme(legend.position="none"), g3 + theme(legend.position="none" ), top ="High School Student Cluster Analysis", ncol=1) )
```

Output:

```
> library (plyr)
> library(ggplot2)
> library(cluster)
> library(lattice)
> library(graphics)
> library(grid)
> #library(gridExtra)
> #library("cowplot")
> grade_input = as.data.frame(read.csv("C:/2020-21/IDOL/Big Data/grades_km_input.csv"))
> kmdata_orig = as.matrix(grade_input[,c("Student","English","Math","Science")])
> kmdata <- kmdata_orig[,2:4]
> kmdata[1:10,]

      English Math Science
 [1,]      99   96      97
 [2,]      99   96      97
 [3,]      98   97      97
 [4,]      95  100      95
 [5,]      95   96      96
 [6,]      96   97      96
 [7,]     100   96      97
 [8,]      95   98      98
 [9,]      98   96      96
[10,]      99   99      95

> wss <- numeric(15)

> for (k in 1:15 ) wss[k] <- sum (kmeans(kmdata, centers=k, nstart=25)$withinss)
> plot(1:15, wss, type="b", xlab="Number of Clusters " , ylab="Within Sum of Squares")
> km = kmeans(kmdata,3, nstart=25)
> km
K-means clustering with 3 clusters of sizes 244, 158, 218
```

```
Cluster means:
   English     Math  Science
1 85.84426 79.68033 81.50820
2 97.21519 93.37342 94.86076
3 73.22018 64.62844 65.84862

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2
 [64] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
[127] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1
2 1 1 1 2 1 1 1
[190] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
[253] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
[316] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 3 3
[379] 3 3 3 3 3 1 3 1 3 1 1 1 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3
[442] 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3
[505] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3
[568] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 1 1 1 1 2 2 1 1 1 3 3 1 3 1 1 1

Within cluster sum of squares by cluster:
[1] 22984.131  6692.589 34806.339
 (between_SS / total_SS =  76.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> c( wss[3] , sum(km$withinss) )

[1] 64483.06 64483.06
>
> df= as.data.frame(kmdata_orig[,2:4])
> df$cluster = factor(km$cluster)
> centers=as.data.frame(km$centers)
>
> g1= ggplot(data=df, aes(x=English, y=Math, color=cluster )) +
+   geom_point() + theme(legend.position="right") +
+   geom_point(data=centers,

+             aes(x=English,y=Math, color=as.factor(c(1,2,3))), size=10, alpha=.3, show.legend=FALSE)
>
> g2 =ggplot(data=df, aes(x=English, y=Science, color=cluster )) +
+   geom_point() +
+   geom_point(data=centers,
+              aes(x=English,y=Science, color=as.factor(c(1,2,3))),size=10, alpha=.3, show.legend=FALSE)
>
> g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
+   geom_point() +
+   geom_point(data=centers,
+              aes(x=Math,y=Science, color=as.factor(c(1,2,3))), size=10, alpha=.3, show.legend=FALSE)
>
> tmp = ggplot_gtable(ggplot_build(g1))
>
> grid.arrange(arrangeGrob(g1 + theme(legend.position="none"), g2 + theme(legend.position="none"), g3 + theme(legend.position="none" ), top ="High School Student Cluster Analysis", ncol=1) )
```

Output


Within Sum of Squares
Number of Clusters