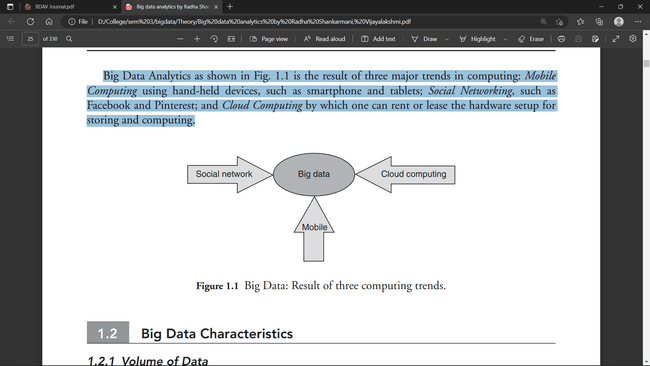
**Chapter No. 1 Introduction of Big Data**

* 1. **Introduction**

Big data analytics describes the process of uncovering trends, patterns, and correlations in large amounts of raw data to help make data-informed decisions. These processes use familiar statistical analysis techniques—like clustering and regression—and apply them to more extensive datasets with the help of newer tools.

Big data is a relative term. If big data is referred by “volume” of transactions and transaction history, then hundreds of terabytes (1012 bytes) may be considered “big data” for a pharmaceutical company and volume of transactions in petabytes (1015 bytes).

Big Data Analytics as shown in Fig. 1.1 is the result of three major trends in computing: Mobile Computing using hand-held devices, such as smartphone and tablets; Social Networking, such as Facebook and Pinterest; and Cloud Computing by which one can rent or lease the hardware setup for storing and computing.



Big data analytics is the use of advanced analytic techniques against very large, diverse big data sets that include structured, semi-structured and unstructured data, from different sources, and in different sizes from terabytes to zettabytes.

It can be defined as data sets whose size or type is beyond the ability of traditional [relational databases](https://www.ibm.com/in-en/analytics/relational-database) to capture, manage and process the data with low latency. Characteristics of big data include high volume, high velocity and high variety. Sources of data are becoming more complex than those for traditional data because they are being driven by [artificial intelligence (AI)](https://www.ibm.com/in-en/cloud/learn/what-is-artificial-intelligence), mobile devices, social media and the Internet of Things (IoT). For example, the different types of data originate from sensors, devices, video/audio, networks, log files, transactional applications, web and social media — much of it generated in real time and at a very large scale.

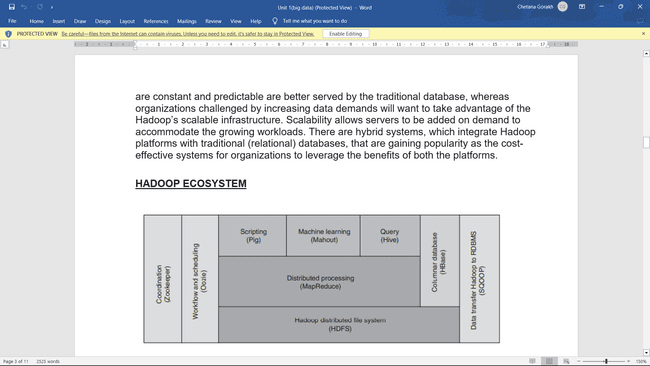
**Chapter No.2 HDFS (Hadoop Distributed File System) and Map Reduce**

**2.1 Introduction**

HDFS is a distributed file system that provides a limited interface for managing the file system to allow it to scale and provide high throughput. HDFS creates multiple replicas of each data block and distributes them on computers throughout a cluster to enable reliable and rapid access. When a file is loaded into HDFS, it is replicated and fragmented into “blocks” of data, which are stored across the cluster nodes; the cluster nodes are also called the DataNodes. The NameNode is responsible for storage and management of metadata, so that when MapReduce or another execution framework calls for the data, the NameNode informs it where the data that is needed resides. Figure 2.1 shows the NameNode and DataNode block replication in HDFS architecture.



**Hadoop Ecosystem**



**2.2 Hardware Required**

* Windows 10,11,etc.
* 8 GB of RAM
* i3 Processor
* 64-bit operating system, x64-based processor

**2.3 Software Required**

* Oracle VM virtualBox
* cloudera-quickstart-vm-5.4.2-0-virtualbox

**2.4 Installation and Configuration**

**Step 1:-** download cloudera image using below link

<https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.4.2-0-virtualbox.zip>

**Step 2:-** Unzip the downloaded Zipped file. After unzipping we will get a folder named cloudera-quickstart-vm-5.4.2-0-virtualbox. Inside this folder two files will be there.

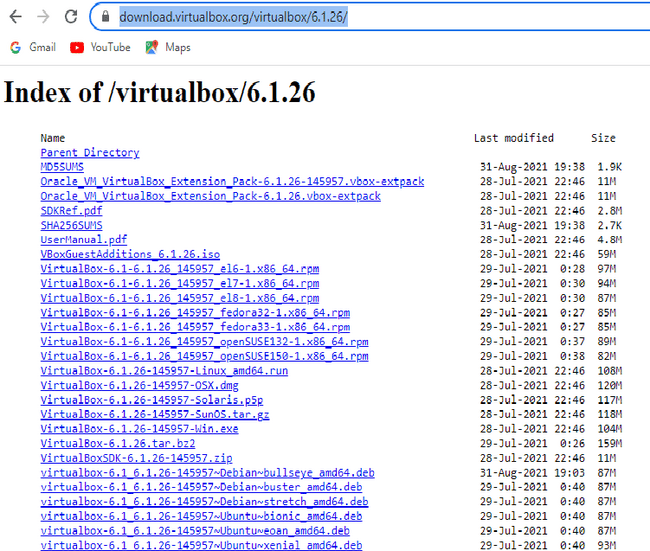
a) cloudera-quickstart-vm-5.4.2-0-virtualbox

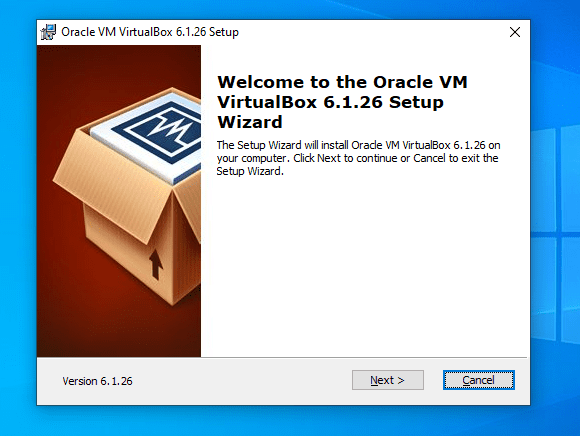
b) cloudera-quickstart-vm-5.4.2-0-virtualbox-disk1

Now next step is to download Oracle VM virtual box by following below steps.

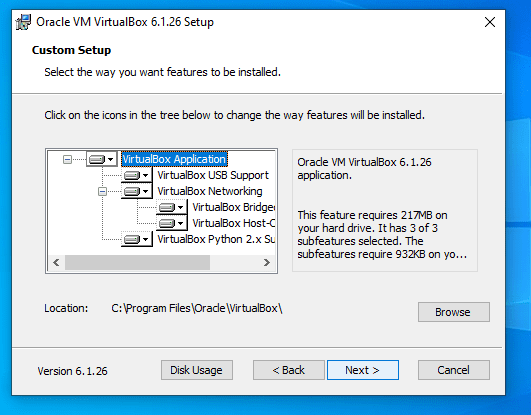
**Step 3:-** Download Oracle VM virtualBox for WINDOWS using this link

<https://download.virtualbox.org/virtualbox/6.1.26/>

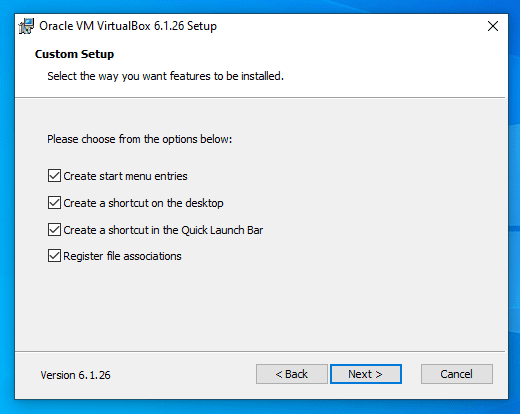


**Step 4:-** After downloading this .exe file simply double click on it and install it.****

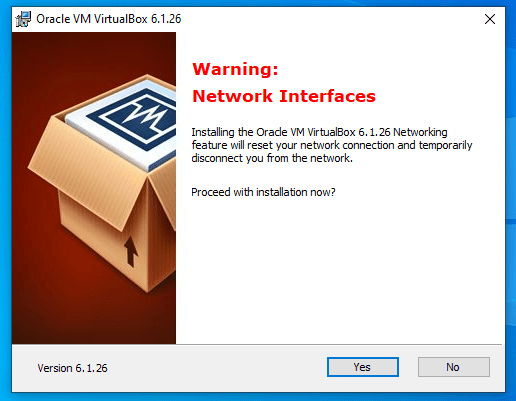
**Step5:- Click on Next**

****

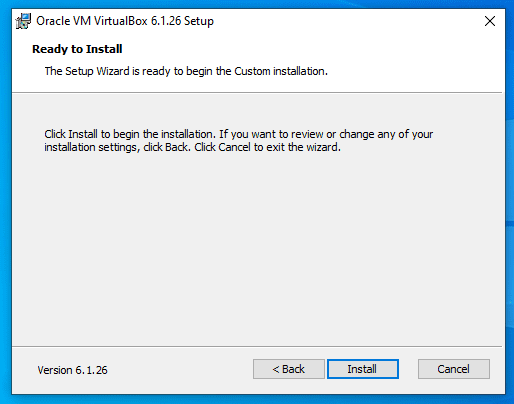
**Step6 :- Click on Next**

****

**Step7 :- Click on Next**

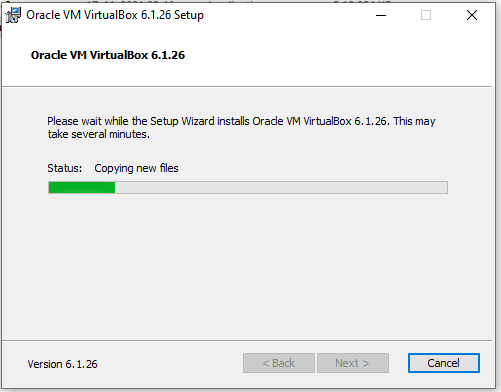
****

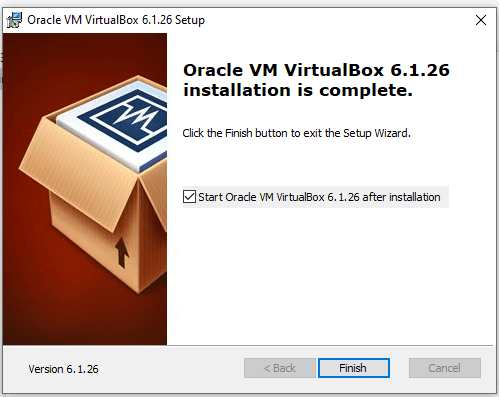
**Step8 :- Click on Yes**

****

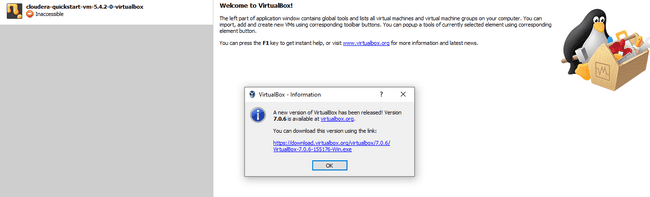
**Step9 :- Click on Install**

**Step 10:-**  After clicking install it will ask to allow changes in device then click on yes. After clicking on Yes it will start installing.

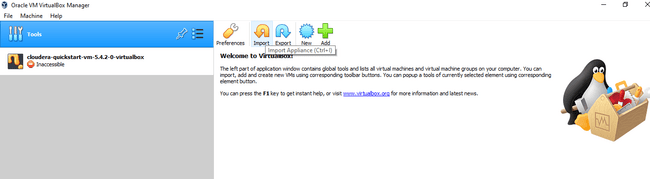


****

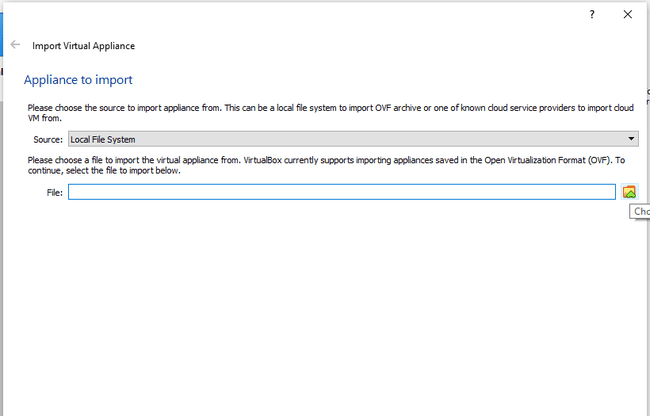
**Step 11:-** Click on Finish. Below screen will appear. Click on OK.

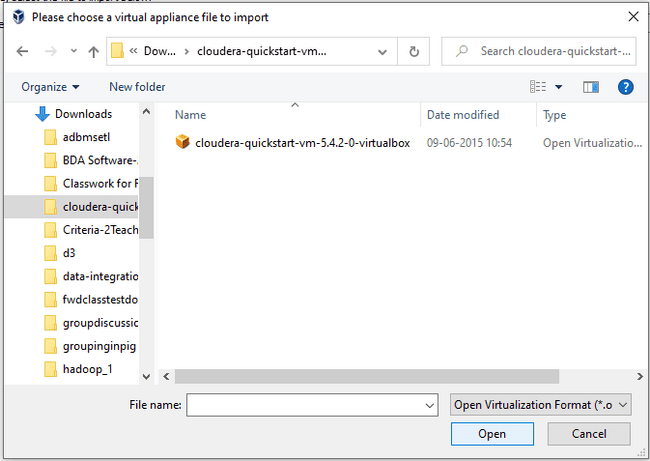
****

**Step 12:-** After clicking OK in step 11 below screen will appear then click on import icon.

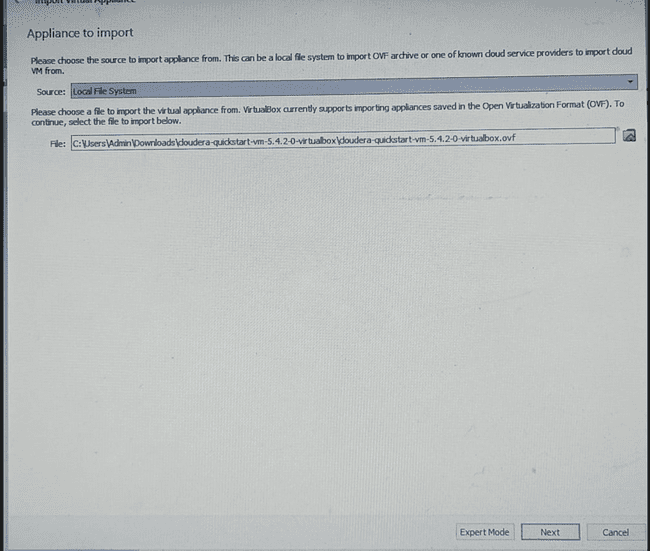
****

**Step 13:- Import the cloudera file mentioned in step 2 (a) and click on Open.**

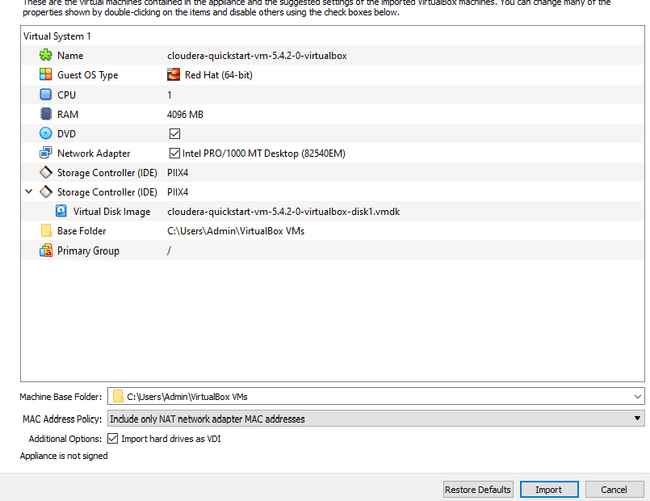
****

****

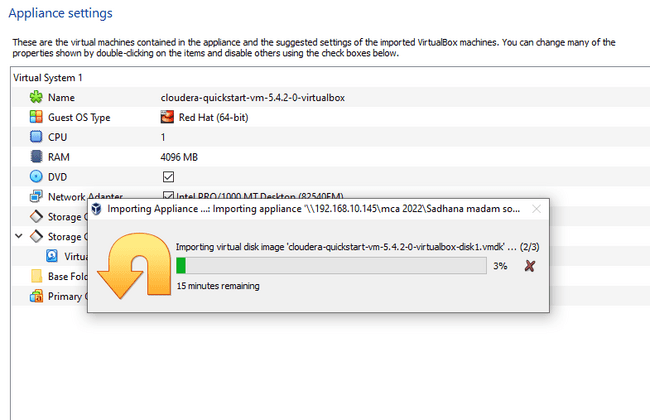
**Step 14:-** Click on Next in below screen**.**

****

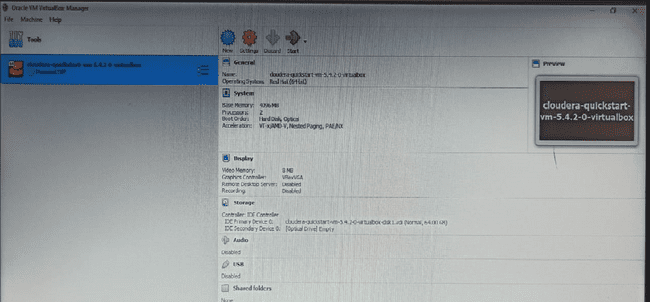
**Step 15:-** Click on import in below screen.

****

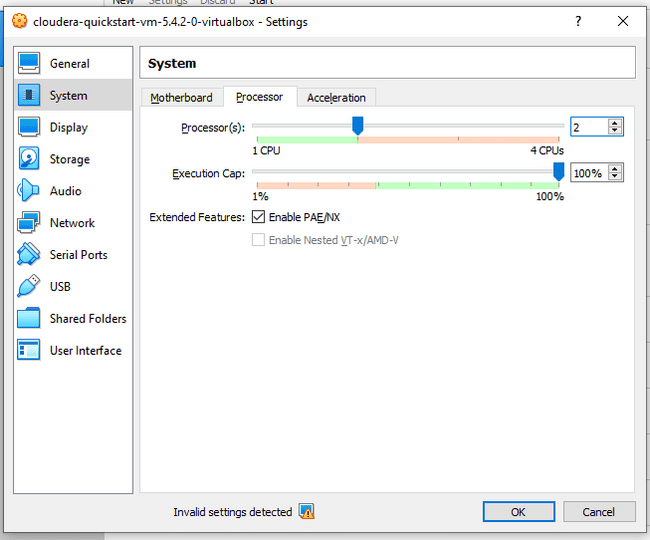
**Step 16:** After clicking on import it will start importing file.



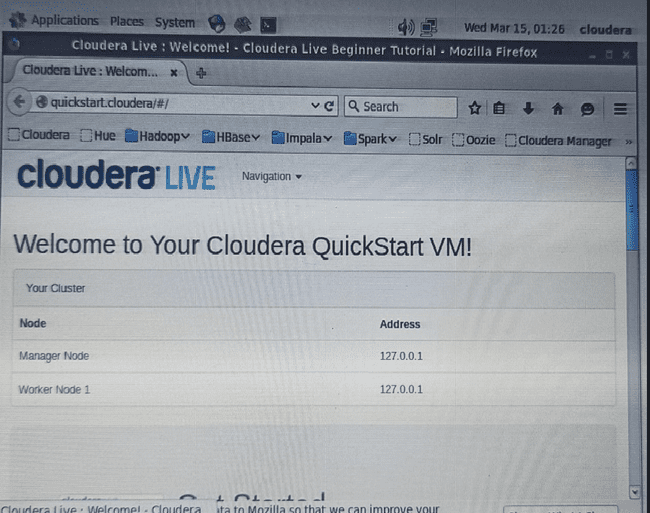
**Step 17:** After importing file below screen will appear

****

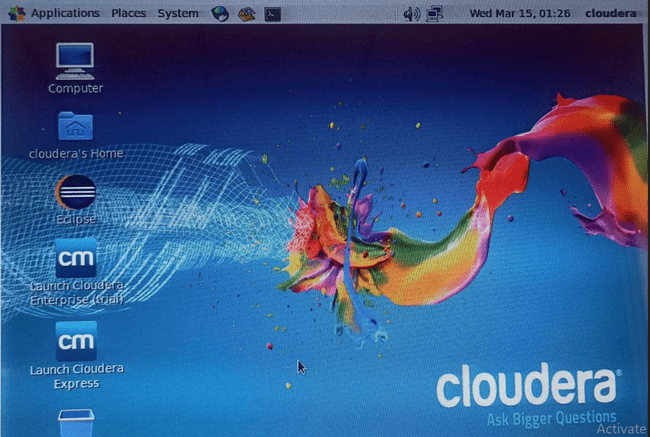
**Step 18**: Click on setting then select system from left pane and click on processor tab and change processor value to 2 from 1.

****

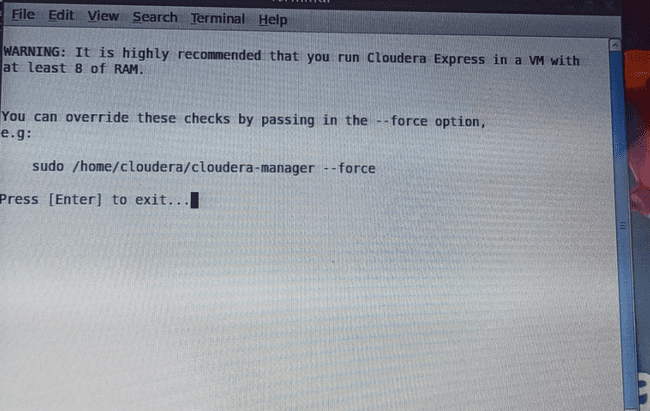
**Step 19:** Click on Start button of the screen shown in step 17. After clicking start button below screen will appear.



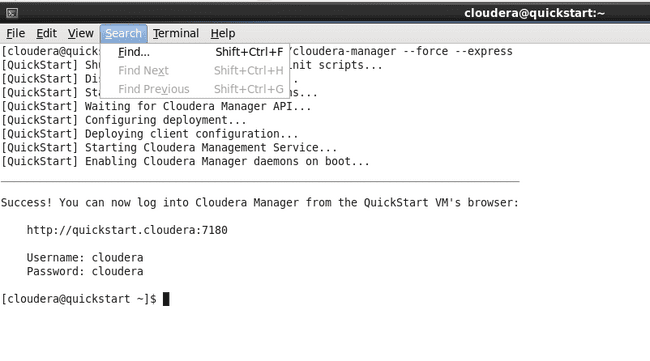
**Step 20:** After closing welcome screen below screen will appear**.**

****

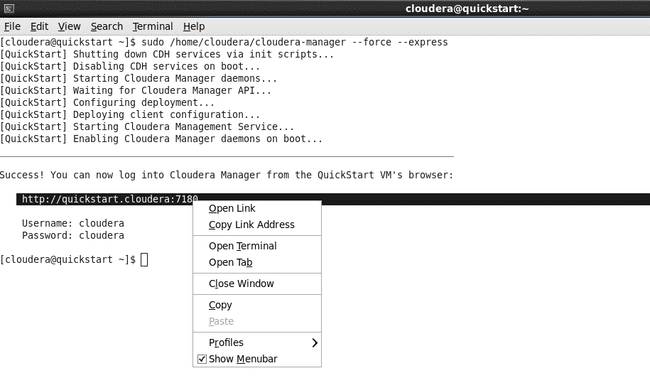
**Step 21:-** Click Launch Cloudera Express icon on screen shown in step 20. Below screen will appear.

****

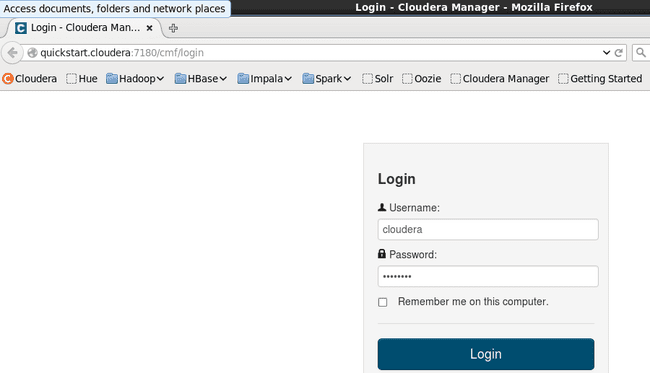
**Step 22:-** Copy the text sudo /home/cloudera/cloudera-manager - - force and click on terminal icon(Black colour) on top and write down – express besides –force and press enter. Below screen will appear.

****

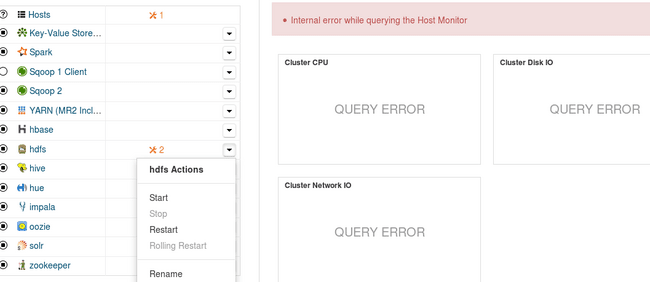
**Step 23:** In step 22 there is a link <http://quickstart.cloudera:7180> . Right click and open in new tab.

****

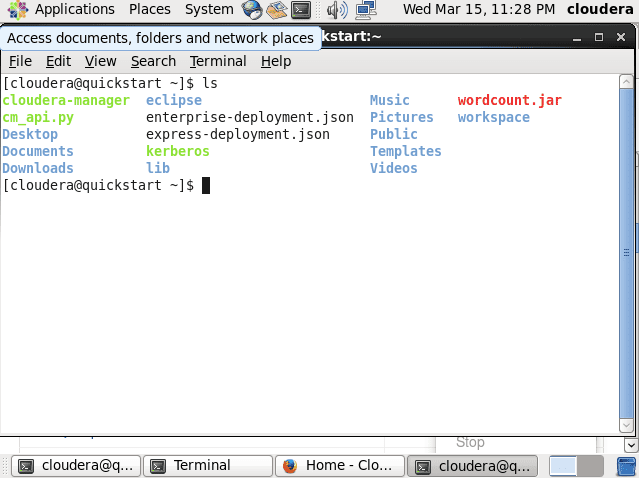
**Step 24: Give user name and password as cloudera.**

****

**Step 25: Start required services. E.g HDFS,HIVE,HBASE,ZOOKEEPER,SPARK etc.**

****

Now we can execute any HDFS command and Linux command . For example after executing ls command below screen will appear.



**2.5 Practice**

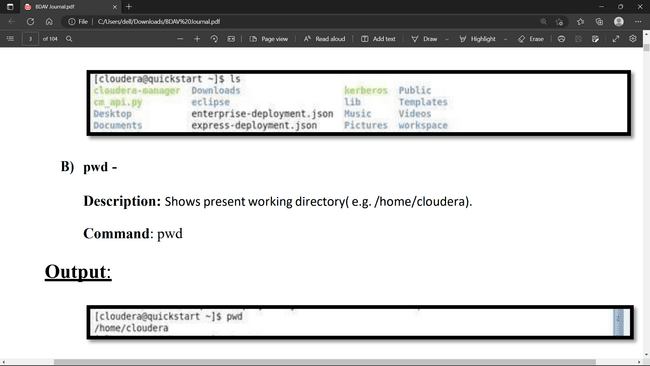
**Practical 1 :** Implementation of HDFS /Linux Commands (mkdir, touchz, copy from local/put, copy to local/get, movefrom local, cp, rmr, du, dus, stat).

1. **ls:**

Description: Shows list of files and directories.

Command: ls

Output:

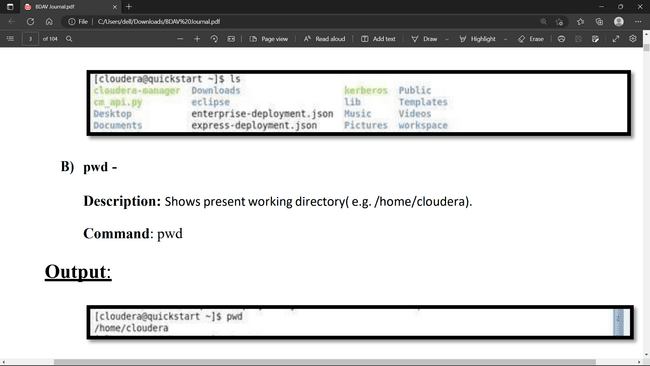


1. **pwd :**

Description: Shows present working directory ( e.g./home/cloudera).

Command: pwd

Output:

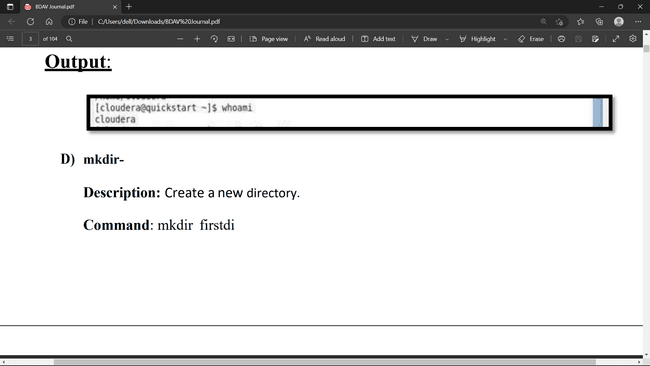


1. **whoami:**

Description: Shows the current user.

Command: whoami

Output:

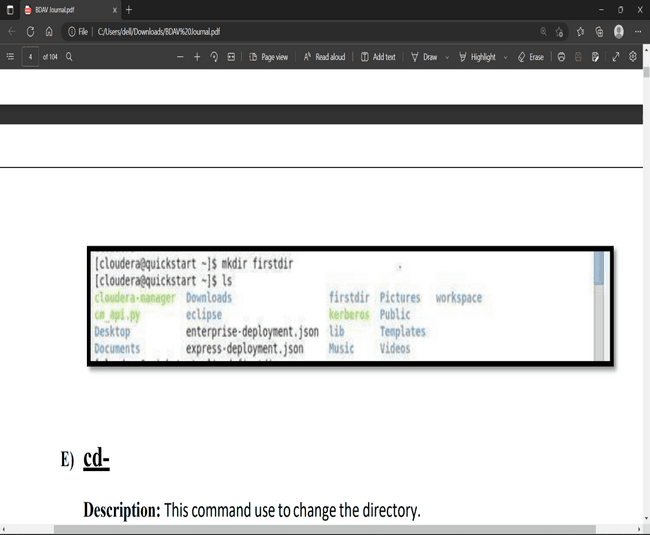


1. **mkdir:**

Description: Create a new directory.

Command: mkdir firstdir

Output:

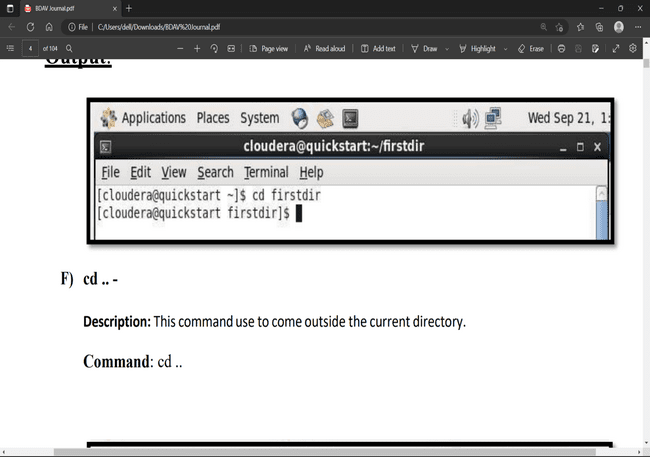


1. **cd:**

Description: This command use to change the directory.

Command: cd directory name (Ex- cd firstdir)

Output:

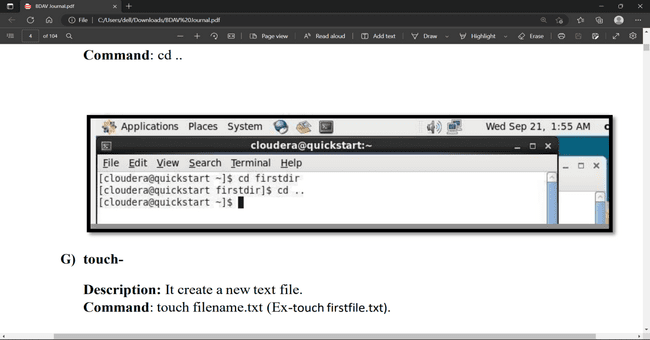


1. **cd .. :**

Description: This command use to come outside the current directory.

Command: cd ..

Output:



1. **touch:**

Description: It create a new text file.

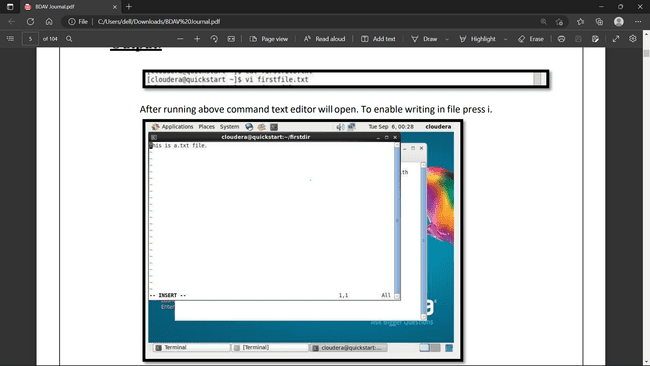
Command: touch filename.txt (Ex-touch firstfile.txt).

Output:

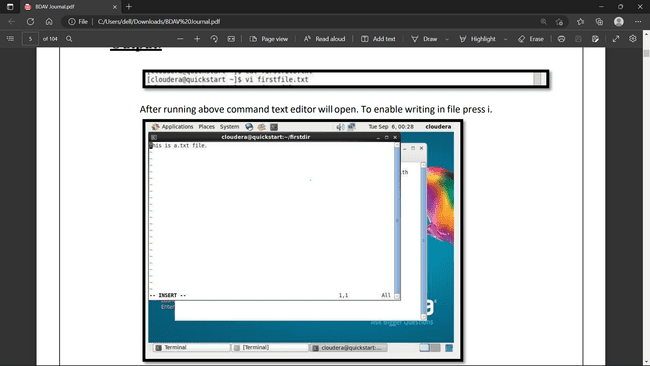
1. **vi:**

Description: It allow user to write something in text file.

Command: vi filename (Ex- vi firstfile.txt)



Output: After running above command text editor will open. To enable writing in file press i.



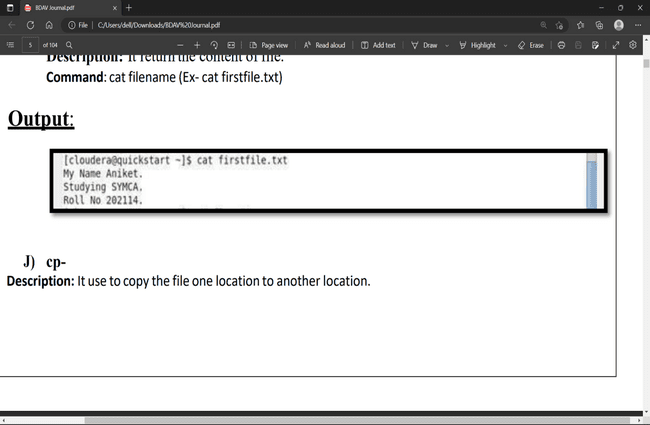
To save the text file use ESC:wq! Press enter.

1. **Cat:**

Description: It return the content of file.

Command: cat filename (Ex- cat firstfile.txt)

Output:

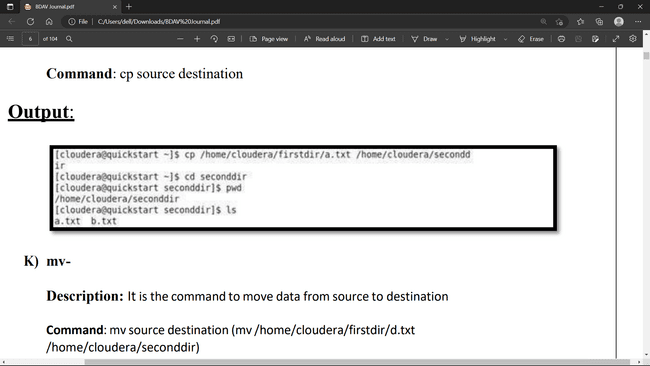


1. **Cp:**

Description: It use to copy the file one location to another location.

Command: cp source destination

Output:

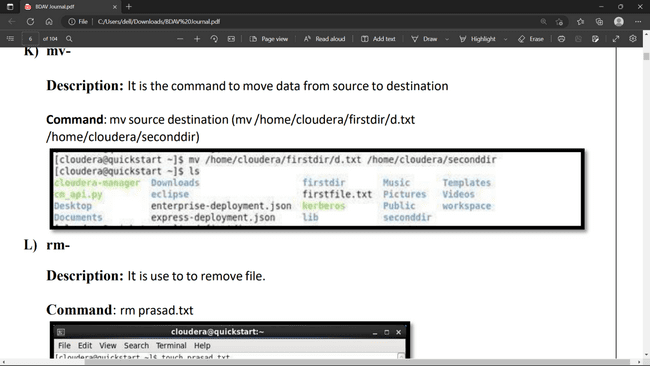


1. **Mv:**

Description: It is the command to move data from source to destination

Command: mv source destination (mv /home/cloudera/firstdir/d.txt /home/cloudera/seconddir)

Output:

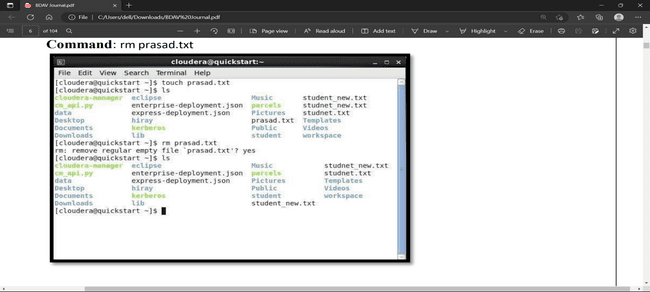


1. **Rm:**

Description: It is use to remove file.

Command: rm prasad.txt

Output:



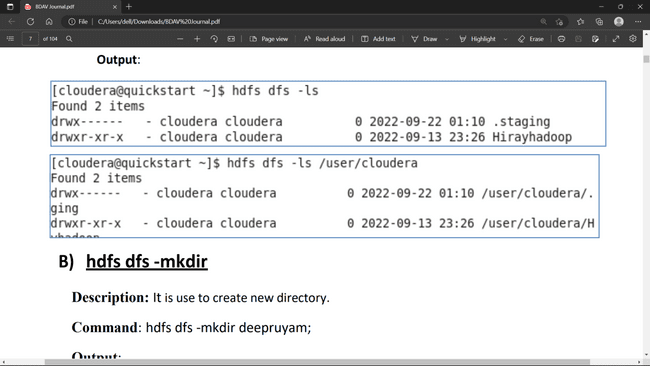
**HDFS Commands**

1. **hdfs dfs-ls**

Description: Displaysthe list of files in hadoop file system.( hdfs dfs-ls /user/cloudera (default directory))

Command: hdfs dfs -ls;

Output:



1. **hdfs dfs -mkdir**

Description: It is use to create new directory.

Command: hdfs dfs -mkdir deepruyam;

Output:



1. **hdfs dfs -put**

Description: It is used copy file from Linux to Hadoop file system.

Command: hdfs dfs –put student.txt/user/cloudera/firsthiray;

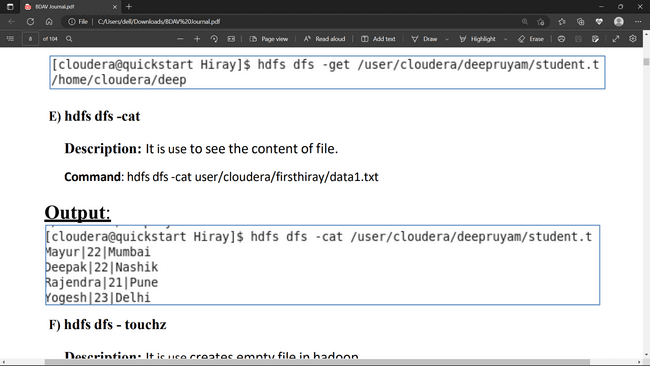
Output:



1. **hdfs dfs -get**

Description: It is use copy file from Hadoop to Linux.

Command: hdfs dfs-get user/cloudera/hiray\_hadoop/student.txt /home/cloudera. Output:

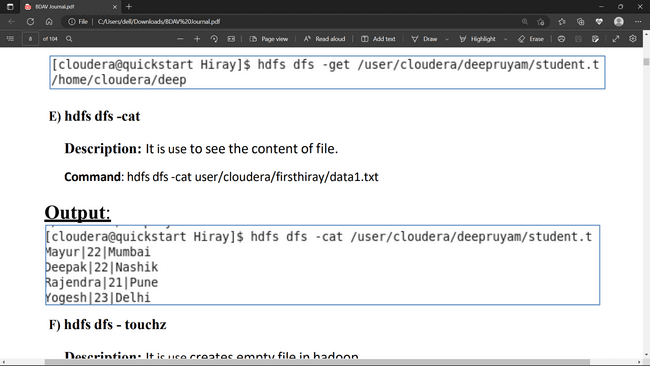


1. **hdfs dfs -cat**

Description: It is use to see the content of file.

Command: hdfs dfs-cat user/cloudera/firsthiray/data1.txt

Output:

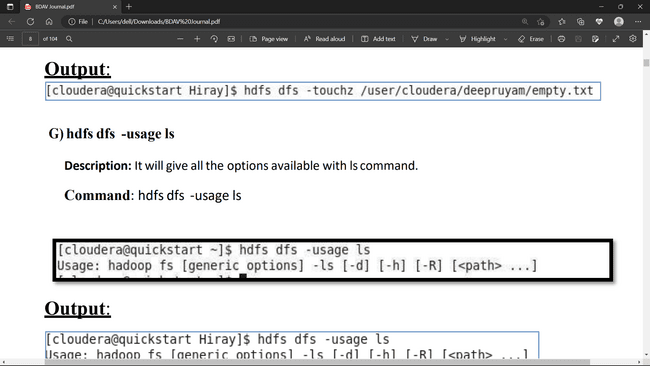


1. **hdfs dfs - touchz**

Description: It is use creates empty file in hadoop.

Command: hdfs dfs -touchz user/cloudera/hiray\_hadoop/student1.txt

Output:

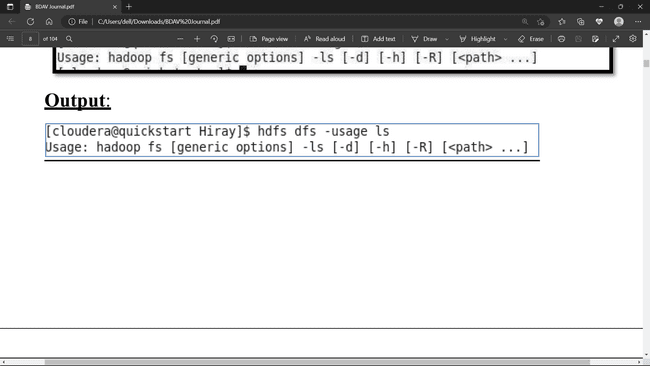


1. **hdfs dfs -usage ls**

Description: It will give all the options available with ls command.

Command: hdfs dfs -usage ls

Output:

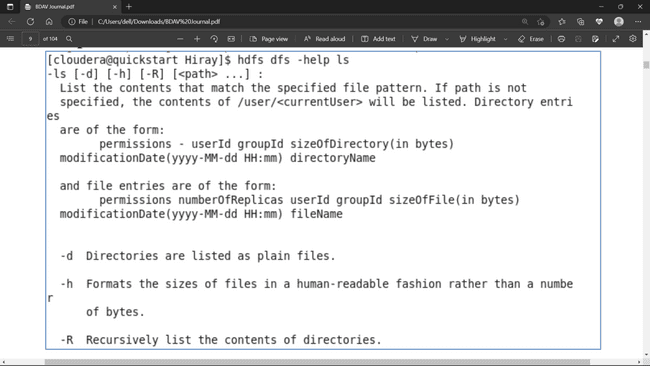


1. **hdfs dfs - help ls**

Description: It gives help of command

Command: hdfs dfs - help ls

Output:

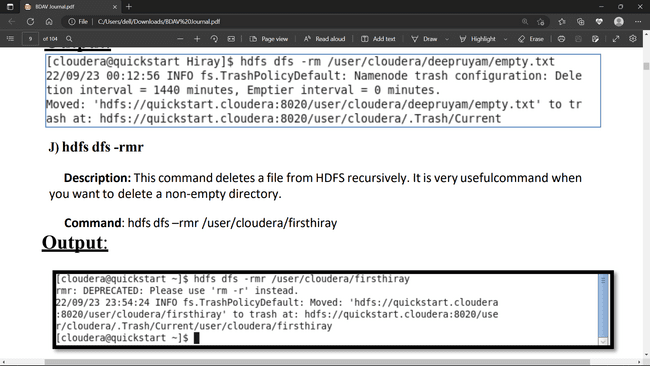


1. **hdfs dfs -rm**

Description: It will move the file in trash folder

Command: hdfs dfs -rm /user/cloudera/hiray\_hadoop/student1.txt

Output:

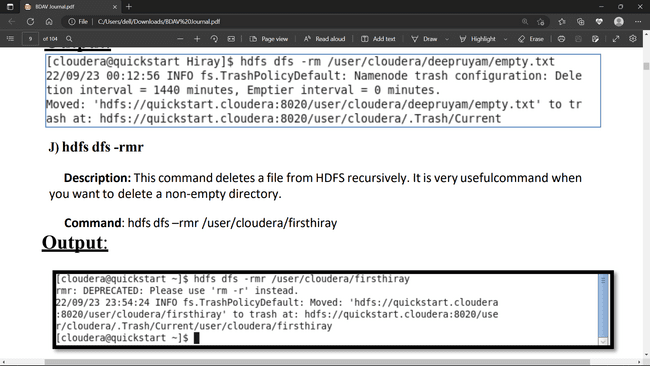


1. **hdfs dfs -rmr**

Description: This command deletes a file from HDFS recursively. It is very useful command when you want to delete a non-empty directory.

Command: hdfs dfs –rmr /user/cloudera/firsthiray

Output:

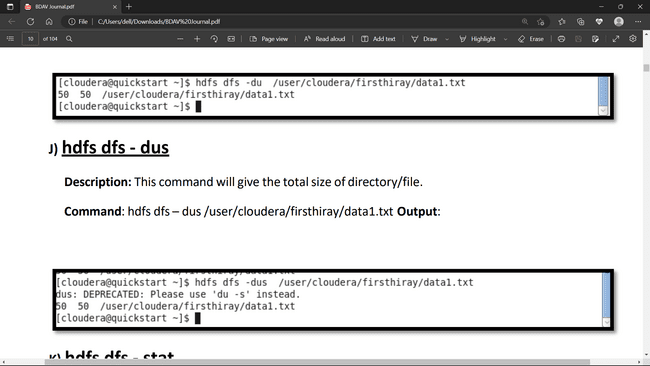


1. **hdfs dfs - du**

Description: It will give the size of each file in directory.

Command: hdfs dfs – du /user/cloudera/firsthiray/data1.txt

Output:

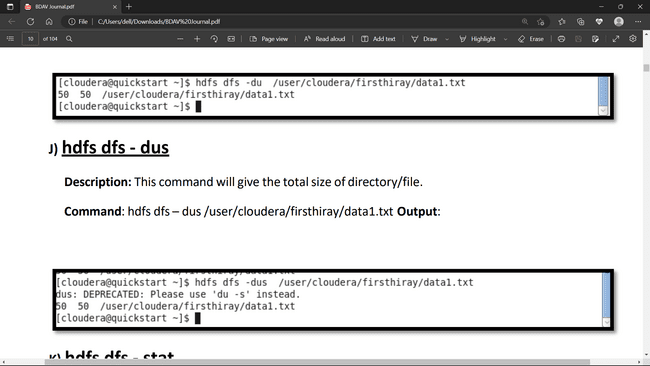


**J) hdfs dfs - dus**

Description: This command will give the total size of directory/file.

Command: hdfs dfs – dus /user/cloudera/firsthiray/data1.txt

Output:

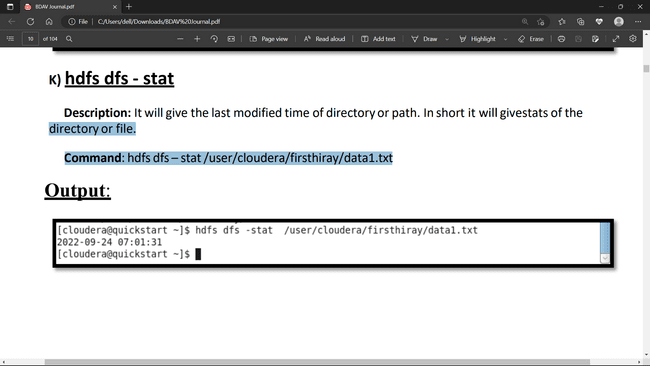


1. **hdfs dfs - stat**

Description: It will give the last modified time of directory or path. In short it will give stats of the directory or file.

Command: hdfs dfs – stat/user/cloudera/firsthiray/data1.txt

Output:



**2.6 MAP REDUCE**

**INTRODUCTION**

MapReduce processes are divided between two applications, JobTracker and TaskTracker at the cluster level. JobTracker is responsible for scheduling job runs and managing computational resources across the cluster; hence it runs on only one node of the cluster. Each MapReduce job is split into a number of tasks which are assigned to the various TaskTrackers depending on which data is stored on that node. So TaskTracker runs on every slave node in the cluster. JobTracker oversees the progress of each TaskTracker as they complete their individual tasks

1 Main Components of MapReduce

The main components of MapReduce are listed below:

1. **JobTrackers:** JobTracker is the master which manages the jobs and resources in the cluster. The JobTracker tries to schedule each map on the TaskTracker which is running on the same DataNode as the underlying block.

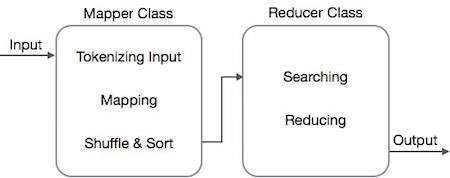
2. **TaskTrackers:** TaskTrackers are slaves which are deployed on each machine in the cluster. They are responsible for running the map and reduce tasks as instructed by the JobTracker.

3. **JobHistoryServer:** JobHistoryServer is a daemon that saves historical information about completed tasks/applications

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

* The map task is done by means of Mapper Class
* The reduce task is done by means of Reducer Class.

Mapper class takes the input, tokenizes it, maps and sorts it. The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.



MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems. In technical terms, MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster.

These mathematical algorithms may include the following −

* Sorting
* Searching
* Indexing
* TF-IDF

## Sorting

Sorting is one of the basic MapReduce algorithms to process and analyze data. MapReduce implements sorting algorithm to automatically sort the output key-value pairs from the mapper by their keys.

* Sorting methods are implemented in the mapper class itself
* In the Shuffle and Sort phase, after tokenizing the values in the mapper class, the **Context** class (user-defined class) collects the matching valued keys as a collection
* To collect similar key-value pairs (intermediate keys), the Mapper class takes the help of **RawComparator** class to sort the key-value pairs.
* The set of intermediate key-value pairs for a given Reducer is automatically sorted by Hadoop to form key-values (K2, {V2, V2, …}) before they are presented to the Reducer.

## Searching

Searching plays an important role in MapReduce algorithm. It helps in the combiner phase (optional) and in the Reducer phase. Let us try to understand how Searching works with the help of an example.

### Example

The following example shows how MapReduce employs Searching algorithm to find out the details of the employee who draws the highest salary in a given employee dataset.

* Let us assume we have employee data in four different files − A, B, C, and D. Let us also assume there are duplicate employee records in all four files because of importing the employee data from all database tables repeatedly. See the following illustration.



* **The Map phase** processes each input file and provides the employee data in key-value pairs (<k, v> : <emp name, salary>). See the following illustration.



* **The combiner phase** (searching technique) will accept the input from the Map phase as a key-value pair with employee name and salary. Using searching technique, the combiner will check all the employee salary to find the highest salaried employee in each file. See the following snippet.

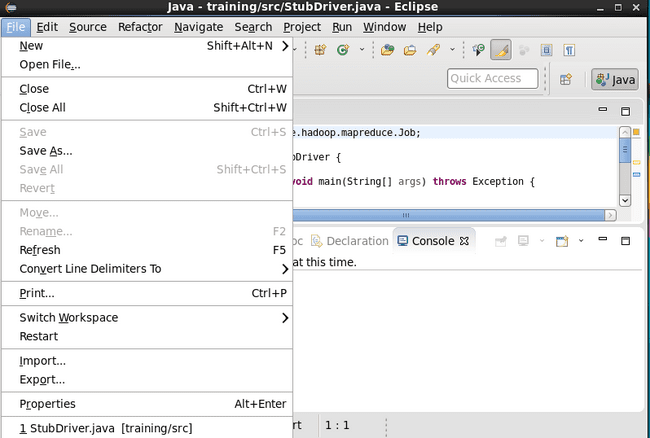
**2.7 PRACTICE**

**Practical 1: Write a program in Map Reduce for Word Count operation.**

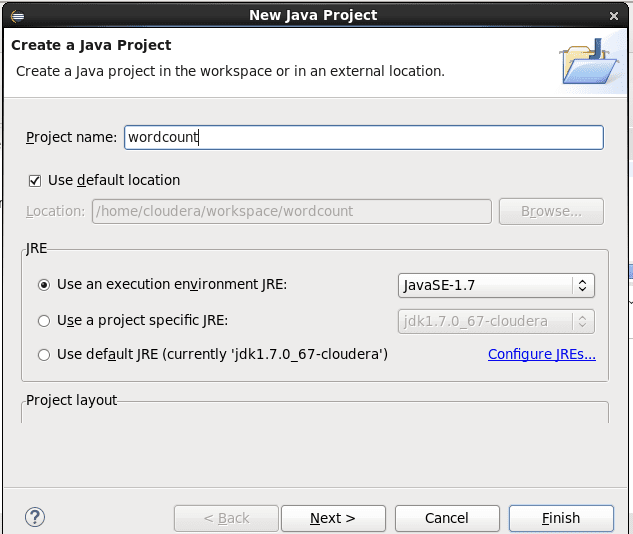
**Step 1 :** Double click on eclipse in cloudera manager screen as shown below.

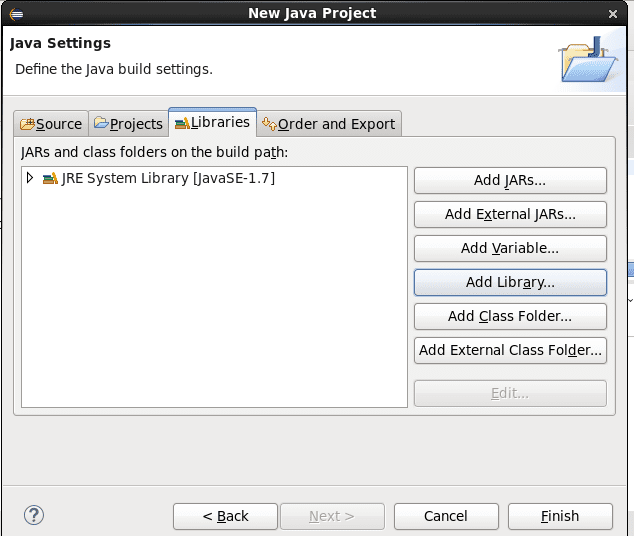
****

**Step2**: Click on File🡪New🡪 Java Project

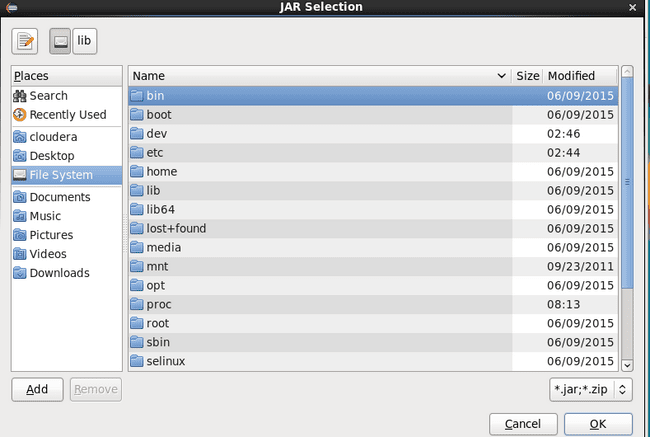
****

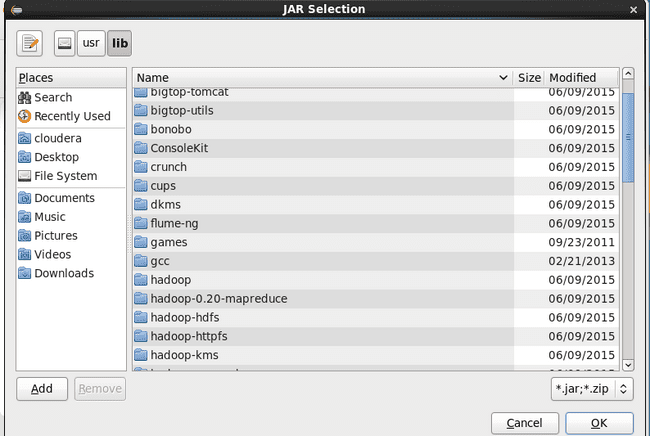
**Step3:-** Give project Name wordcount and click Next.

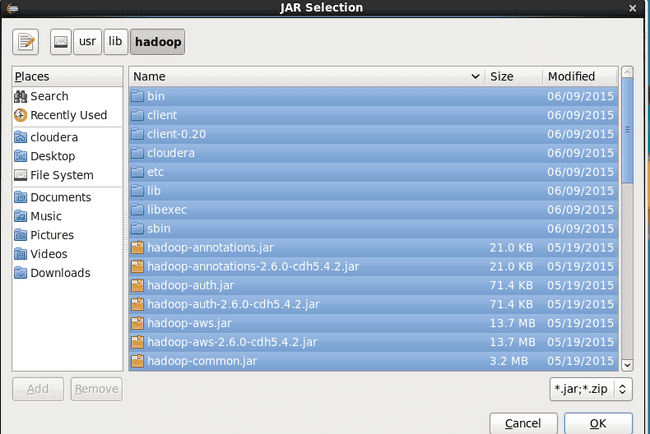
****

**Step 4: After clicking Next in step 3 below screen will appear.After that do not click on Next. Click on Libraries Tab. Then Click on Add External Jars.­­­­**

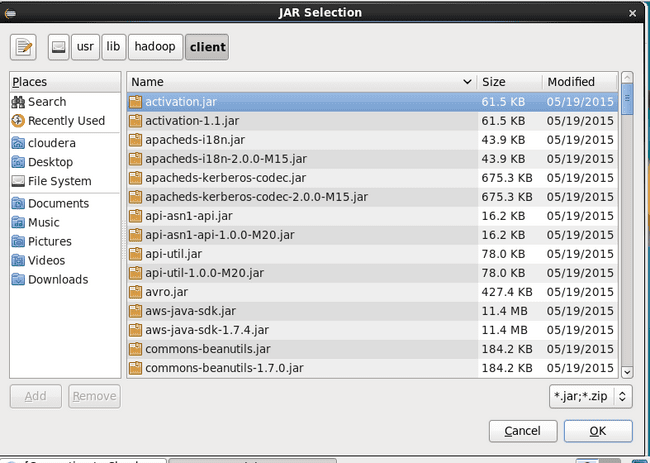
**Step 5:- Click on File System🡪 Usr🡪Lib🡪Haddop then select all Jar files and click OK.**

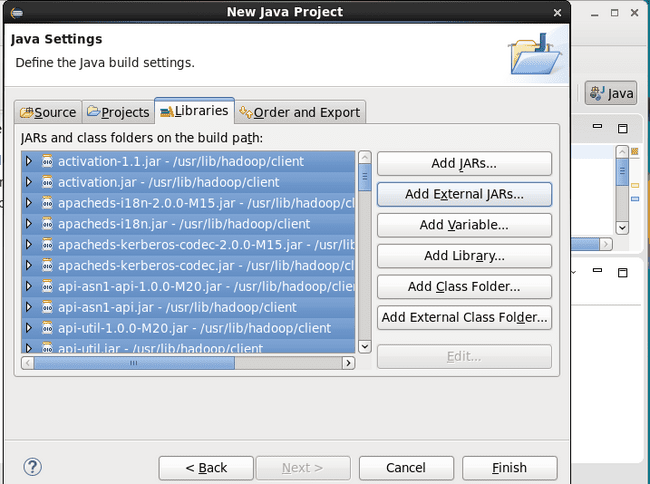
****

****

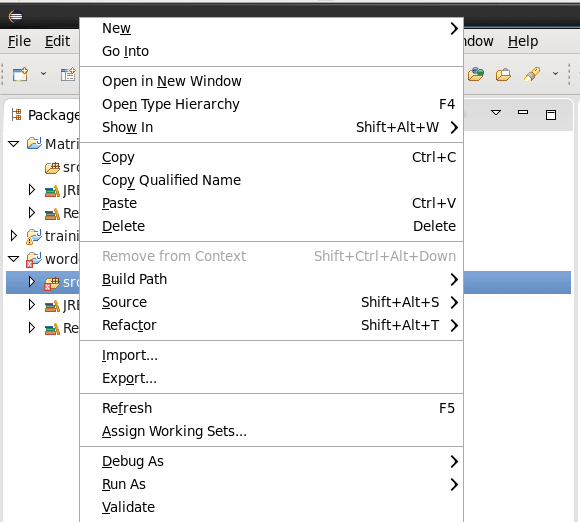
****

**Step 6**:- Click again on Add External Jar then click on client .Select All Jar files and click on finish.

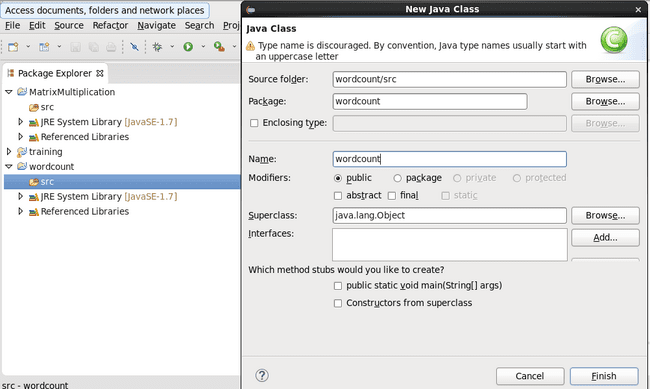
****

****

**Step 7: Click on wordcount🡪src🡪New🡪Class**

****

**Step 8: Give the name as wordcount. Then click on finish.**

****

**Step 9: Write below code of wordcount program in right pane. As shown in below screen shot.**

**Code:**

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

{

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())

{

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer extends Reducer

{

private IntWritable result=new IntWritable();

public void reduce(Text key,Iterablevalues,Context context)throws IOException,InterruptedException

{

int sum=0;

for(IntWritable val:values)

{

sum=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.setReducerClass(IntSumReducer.class);

job.setCombinerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

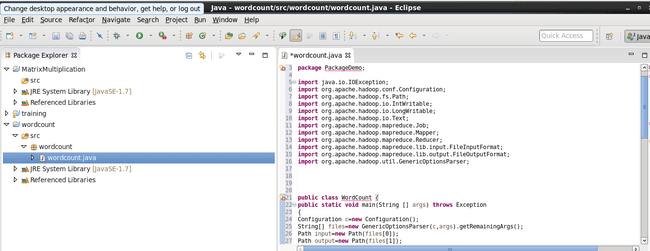
job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job,new Path(args[0]));

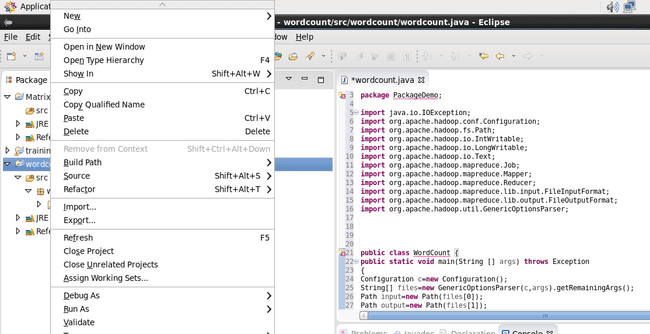
FileOutputFormat.setOutputPath(job, new Path(args[1])); System.exit(job.waitForCompletion(true)?0:1);

}

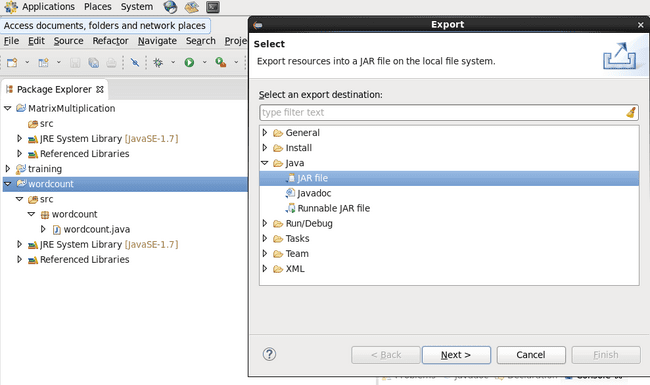
}

****

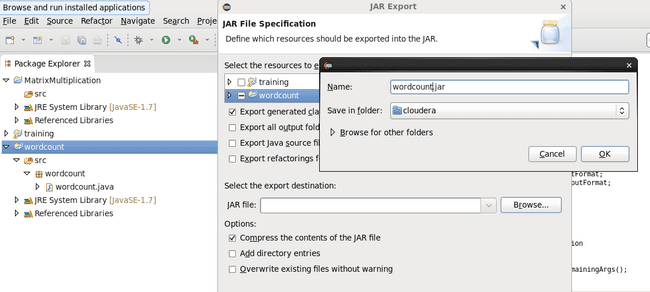
**Step 10: Right click on the wordcount package and click on export.**

****

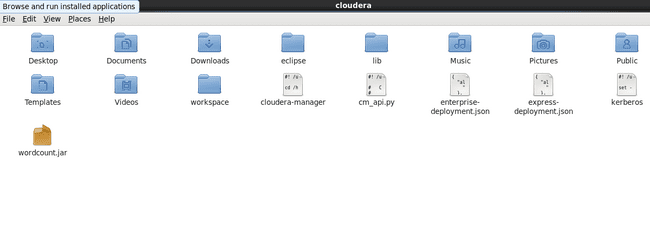
**Step 10: Click on Java🡪JAR File🡪 Next**

****

**Step 11: Browse the Jar file and give the name as wordcount and save in cloudera folder. As shown in below screen shot. Click on OK🡪Finish🡪 OK.**

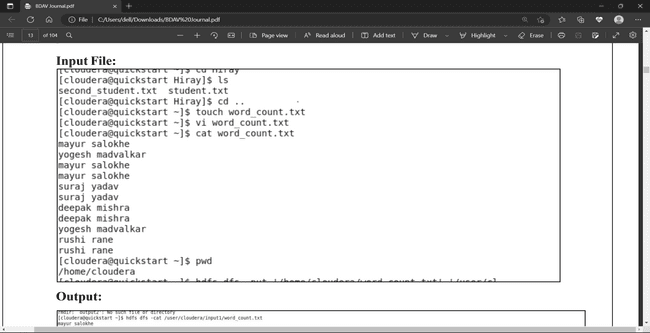
****

**Step 12: Click on computer🡪Filesystem🡪Home🡪 Cloudera. Check the wordcount.jar file in filesystem. It should be there.**

****

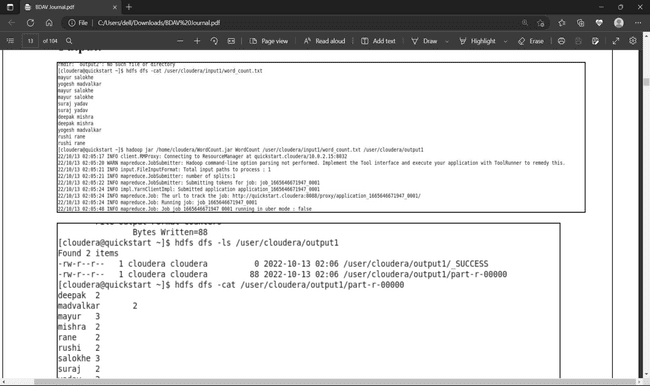
**Step 13: Create input1 and output1 directory in Hadoop file system and Create wordcount.txt file using touch command in Linux. Import wordcound.txt file from linux to Hadoop system using Put command. As shown below.**

**INPUT FILE:**

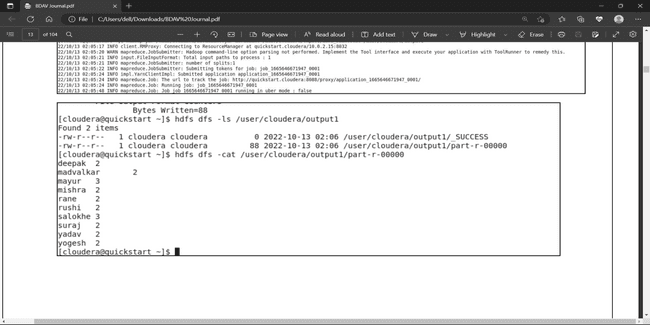


**Step 14: write below command**

**Hadoop jar /home/cloudera/wordcount.jar wordcount /user/cloudera/input1/wordcount.txt /user/cloudera/output1**



**Step 15:** Check the output by the commands given below.



**Practical 2 : Write a program in Map Reduce for Union operation.**

**Repeat Steps 1 to 15 for below program and do required changes as per program.**

**Code:**

import java.io.\*;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.io.LongWritable;

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.MultipleInputs;

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

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

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

import org.apache.hadoop.util.GenericOptionsParser;

public class Union

{

public static class MultipleMaps extends Mapper

{

private final static IntWritable one=new IntWritable(1);

private Text keyEmit=new Text();

public void map(LongWritable k,Text value,Context context)throws IOException, InterruptedException

{

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

while(itr.hasMoreElements())

{

keyEmit.set(itr.nextToken());

context.write(keyEmit, one);

}

}

}

public static class MultipleReducer extends Reducer

{

private IntWritable result=new IntWritable();

public void reduce(Text key,Iterablevalues,Context context)throws IOException, InterruptedException

{

int sum=0;

for(IntWritable val:values)

{

sum +=val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static voidmain(String[] args)throws Exception

{

if(args.length!=3)

{

System.err.println("No of arguments should be 3");

System.exit(0);

}

Configuration c=new Configuration(); String[] files=new GenericOptionsParser(c,args).getRemainingArgs();

Path p1=new Path(files[0]);

Path p2=new Path(files[1]);

Path p3=new Path(files[2]);

FileSystem fs=FileSystem.get(c);

if(fs.exists(p3))

{

fs.delete(p3,true);

}

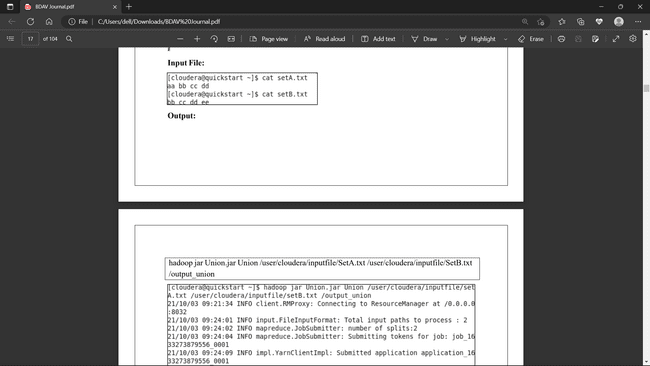
Job job=Job.getInstance(c,"Union");

job.setJarByClass(Union.class); MultipleInputs.addInputPath(job,p1,TextInputFormat.class,MultipleMaps.class); MultipleInputs.addInputPath(job,p2,TextInputFormat.class,MultipleMaps.class); job.setReducerClass(MultipleReducer.class); job.setCombinerClass(MultipleReducer.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class); FileOutputFormat.setOutputPath(job, p3); boolean success=job.waitForCompletion(true); System.exit(success?0:1);

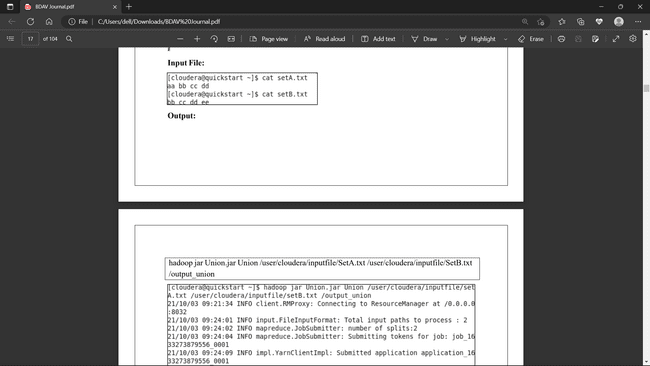
}

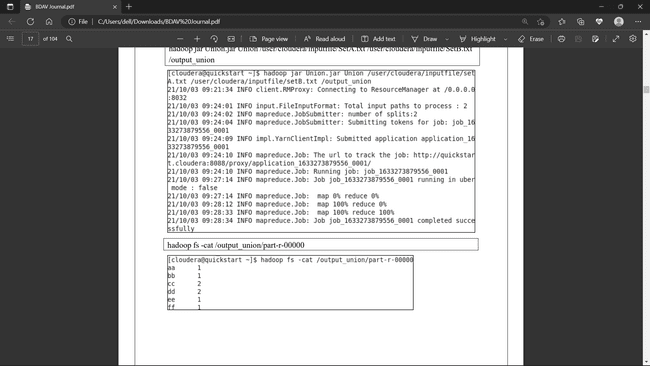
}

**INPUT FILE**



**OUTPUT FILE**





**Practical 3 : Write a program in Map Reduce for Intersection operation.**

**Repeat Steps 1 to 15 for below program and do required changes as per program.**

**Coding:**

import java.io.\*;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.TextInputFormat;

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.MultipleInputs;

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

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

import org.apache.hadoop.util.GenericOptionsParser;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class Difference1

{

public static class MultipleMapA extends Mapper

{

private Text keyEmit=new Text();

private Text valEmit=new Text("A");

public void map(LongWritable k,Text value,Context context)throws IOException, InterruptedException

{

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

while(itr.hasMoreTokens())

{

keyEmit.set(itr.nextToken());

context.write(keyEmit, valEmit);

}

}

}

public static class MultipleMapB extends Mapper

{

private Text keyEmit=new Text();

private Text valEmit=new Text("B");

public void map(LongWritable k,Text value,Context context)throws IOException, InterruptedException

{

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

while(itr.hasMoreTokens())

{

keyEmit.set(itr.nextToken());

context.write(keyEmit, valEmit);

}

}

}

public static class MultipleReducer extends Reducer

{

private Text valEmit=new Text("");

public void Reduce(Text key,Iterable values,Context context)throws IOException, InterruptedException

{

boolean flag=true;

for(Text val:values)

{

if(val.toString().contains("B"))

{

flag=false;

break;

}

}

if(flag)

{

context.write(key, valEmit);

}

}

}

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

{

if(args.length!=3)

{

System.err.println("No of arguments should be 3");

System.exit(0);

}

Configuration c=new Configuration();

String[] files=new GenericOptionsParser(c,args).getRemainingArgs();

Path p1=new Path(files[0]);

Path p2=new Path(files[1]);

Path p3=new Path(files[2]);

FileSystem fs=FileSystem.get(c);

if(fs.exists(p3))

{

fs.delete(p3,true);

}

Job job=Job.getInstance(c,"Difference1");

job.setJarByClass(Difference1.class);

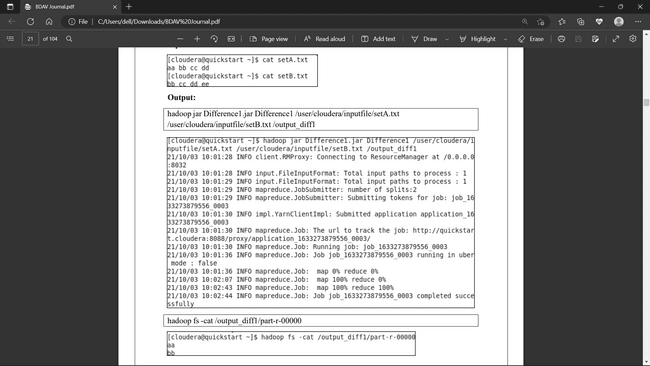
MultipleInputs.addInputPath(job, p1, TextInputFormat.class,MultipleMapA.class); MultipleInputs.addInputPath(job, p2, TextInputFormat.class,MultipleMapB.class); job.setReducerClass(MultipleReducer.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(Text.class); FileOutputFormat.setOutputPath(job, p3); if(!job.waitForCompletion(true)) { System.exit(1);

}

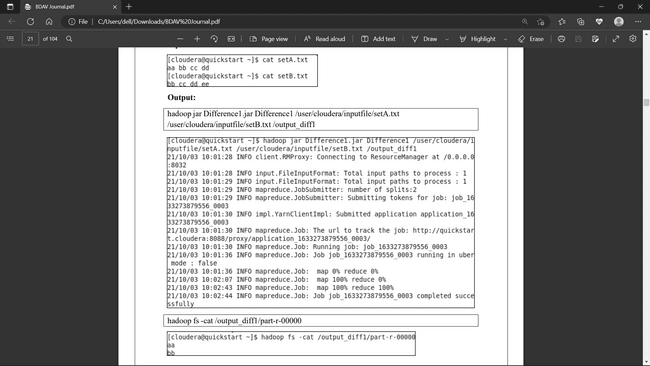
}

}

**INPUT FILE**



**OUTPUT FILE**



**Practical 4: Write a program in Map Reduce for Grouping and Aggregation.**

**Repeat Steps 1 to 15 for below program and do required changes as per program.**

**Coding:**

import java.io.IOException;

import java.io.IOException.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.mapreduce.\*;

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

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

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

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

public class AvgGroup

{

public static class Mapavg extends Mapper

{

private Text gender=new Text();

private IntWritable age=new IntWritable();

public void map(LongWritable key,Text value,Context context)throws IOException, InterruptedException

{

String line=value.toString();

String str[]=line.split(",");

if(str.length>6)

{

gender.set(str[4]);

if(str[1].equals("0"))

{

if(str[5].matches("\\d+"))

{

int i=Integer.parseInt(str[5]);

age.set(i);

}

}

}

context.write(gender,age);

}

}

public static class Reduceavg extends Reducer

{

public void reduce(Text key, Iterablevalues, Context context)throws IOException, InterruptedException

{

int sum=0;

int l=0;

for(IntWritable val:values)

{

l+=1;

sum+=val.get();

}

sum=sum/l;

context.write(key, new IntWritable(sum));

}

}

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

{

Configuration conf=new Configuration();

Job job=Job.getInstance(conf,"AvgGroup");

job.setJarByClass(AvgGroup.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setMapperClass(Mapavg.class);

job.setReducerClass(Reduceavg.class);

job.setCombinerClass(Reduceavg.class);

job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.addInputPath(job,new Path(args[0]));

FileOutputFormat.setOutputPath(job,new Path(args[1]));

Path out=new Path(args[1]);

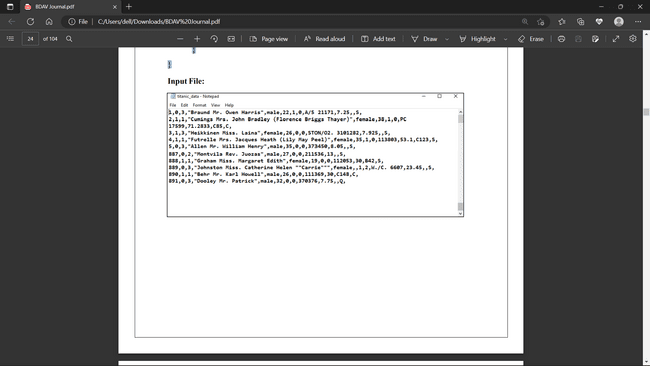
out.getFileSystem(conf).delete(out);

job.waitForCompletion(true);

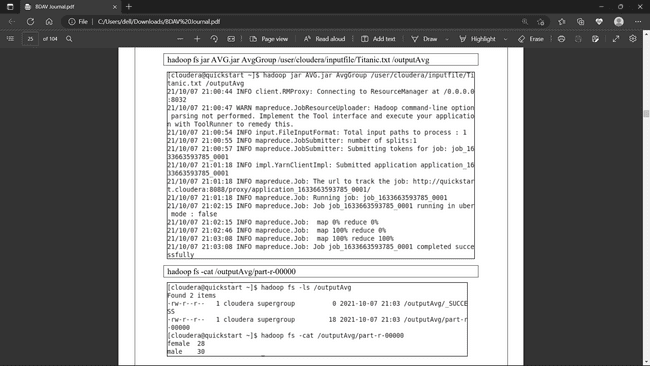
}

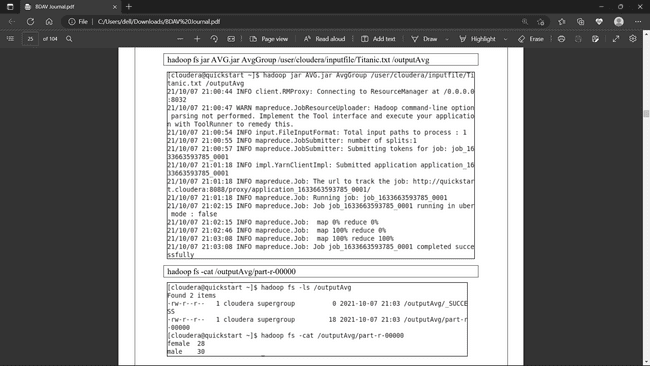
}

**INPUT FILE**



**OUTPUT FILE**





**Practical 5 : Write a program in Map Reduce for Matrix Multiplication.**

**Repeat Steps 1 to 15 for below program and do required changes as per program.**

**Code:**

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.Reducer.Context;

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

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

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

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

public class Matrix

{

public static class MapM extends Mapper

{

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

{

Configuration conf = context.getConfiguration();

int m = Integer.parseInt(conf.get("m"));

int p = Integer.parseInt(conf.get("p"));

String line = value.toString();

String[] indicesAndValue = line.split(",");

Text outputKey = new Text();

Text outputValue = new Text();

if(indicesAndValue[0].equals("M"))

{

for(int k = 0 ; k< p ; k++)

{

outputKey.set(indicesAndValue[1] + "," + k);

outputValue.set(indicesAndValue[0] + "," + indicesAndValue[2] + "," + indicesAndValue[3] ); context.write(outputKey, outputValue);

}

else

{

}

for(int i=0 ; i

{

public void reduce(Text key, Iterable values, Context context) throws IOException, InterruptedException

{

String[] value;

HashMap hashA = new HashMap();

HashMap hashB = new HashMap();

for(Text val : values)

{

value =val.toString().split(",");

if(value[0].equals("M"))

{

}

else

{

}

hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2])); hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));

}

int n = Integer.parseInt(context.getConfiguration().get("n"));

float result = 0.0f;

float m\_ij;

float n\_jk;

for(int j=0; j<n;j++)

{

m\_ij = hashA.containsKey(j) ? hashA.get(j) : 0.0f;

n\_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;

result += m\_ij \* n\_jk;

}

if(result != 0.0f)

{

context.write(null, new Text( key.toString() + " " + Float.toString(result)));

}

}

}

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

{

if(args.length != 2)

{

System.err.println("use 2 arguments");

System.exit(2);

}

Configuration conf = new Configuration();

conf.set("m", "2");

conf.set("n", "3");

conf.set("p", "2");

@SuppressWarnings("depcretion") Job job = new Job(conf,"Matrix");

job.setJarByClass(Matrix.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(MapM.class);

job.setReducerClass(ReduceM.class);

job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

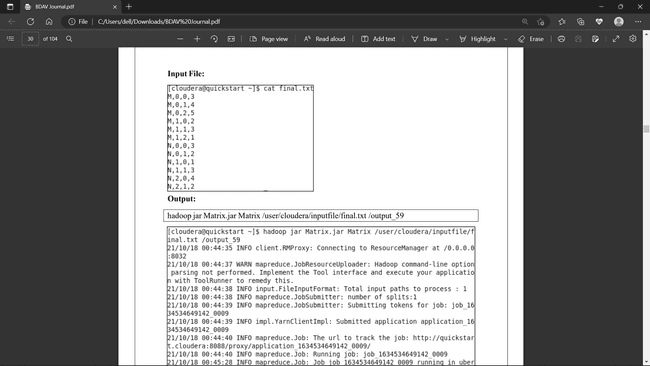
FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.waitForCompletion(true);

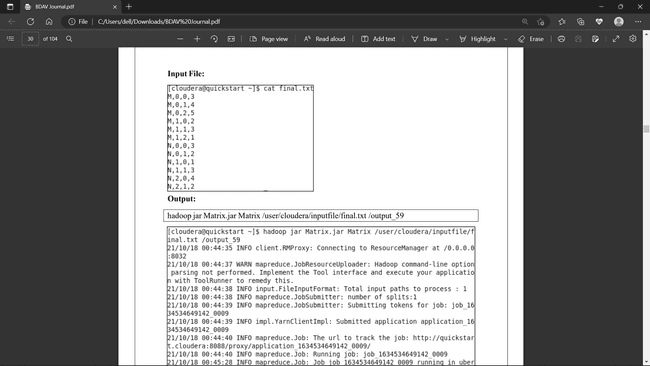
}

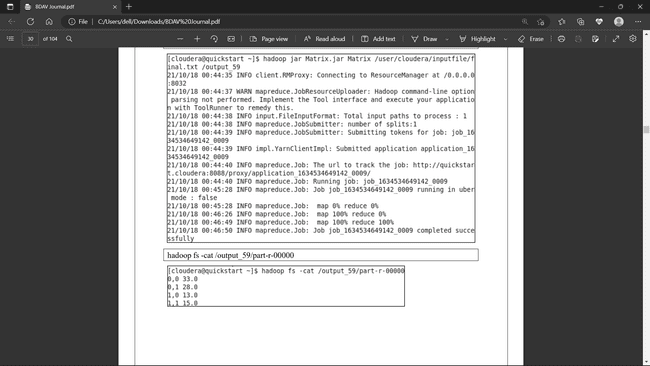
}

**INPUT FILE**



**OUTPUT FILE**





**Chapter No.3 No SQL**

**3.1 Introduction**

NoSQL is database management system that provides mechanism for storage and retrieval of massive amount of unstructured data in a distributed environment on virtual servers with the focus to provide high scalability, performance, availability and agility.

NoSQL database is also referred as Not only SQL. Most NoSQL systems are entirely non-relational; they do not have fixed schemas or JOIN operations. Instead they use objects, key-value pairs, or tuples.

**3.2 Hardware Required**

* At least 10 GB of free disk space plus whatever space is necessary to hold your MongoDB data.
* At least 4 GB of RAM.
* If you use AWS EC2 instances, you should use a minimum of an m3.medium instance.
* The MongoDB Agent must be installed only on 64-bit architectures.

**3.3 Software Required**

MongoDB Community edition

**3.4 Installation and Configuration**

Download the mongodb from <https://fastdl.mongodb.org/windows/mongodb-windows-x86_64-6.0.4-signed.msi> this link and install it.

**3.5 Practice**

**Practical 1:** Basic MongoDB Querying commands.

**Step 1 :-** Go to Windows Powershell and type mongodb at prompt.

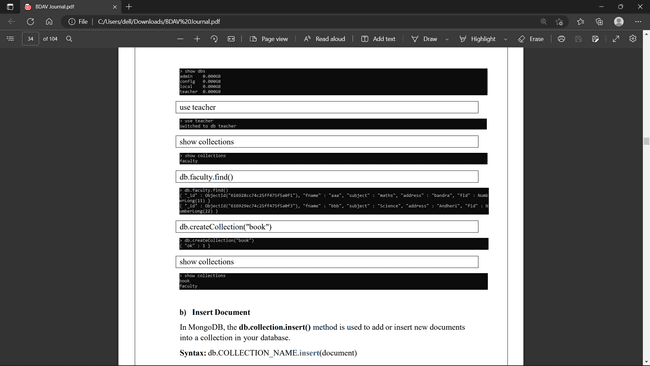
**Step 2:** Execute all the commands given below at prompt one by one.

**a) Create Collection**

In MongoDB, db.createCollection(name, options) is used to create collection. But usually you don’t need to create collection. MongoDB creates collection automatically when you insert some documents. Syntax: db.createCollection(name, options)

Example:

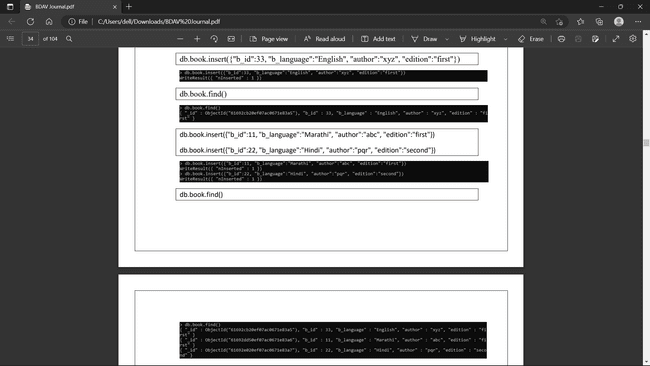


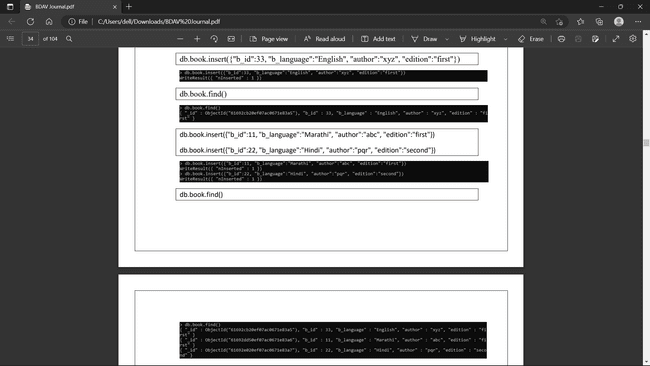


**b) Insert Document**

In MongoDB, the db.collection.insert() method is used to add or insert new documents into a collection in your database. Syntax: db.COLLECTION\_NAME.insert(document)

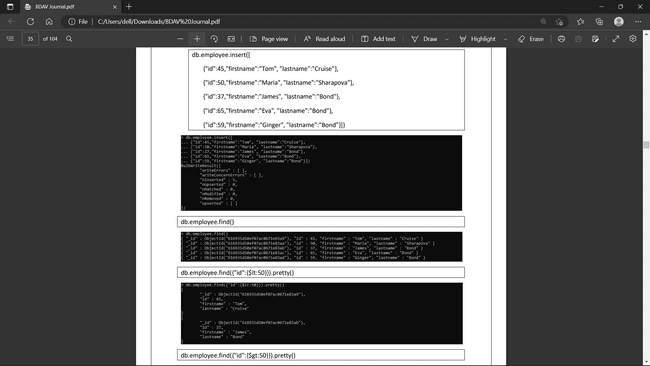
Example:



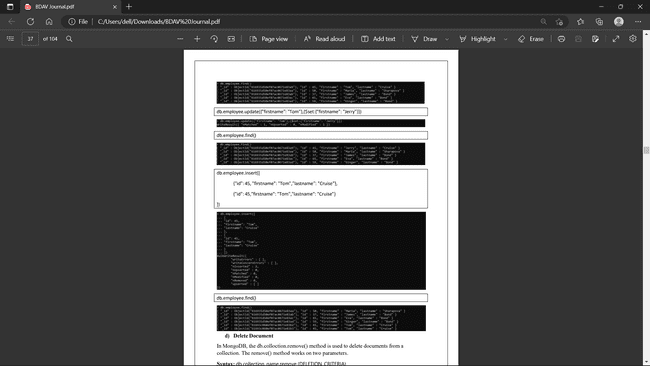


**c) Query Document**

In MongoDB, the db.collection.find() method is used to retrieve documents from a collection. This method returns a cursor to the retrieved documents



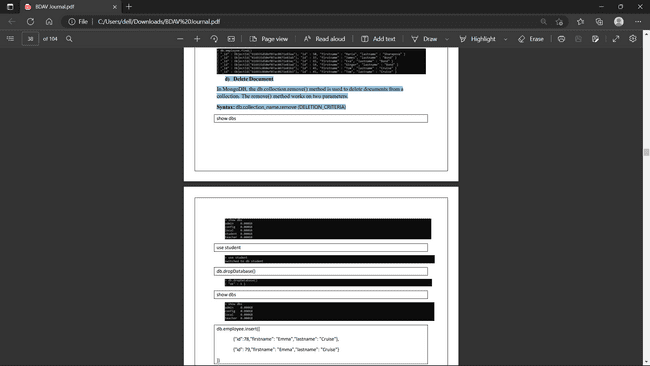


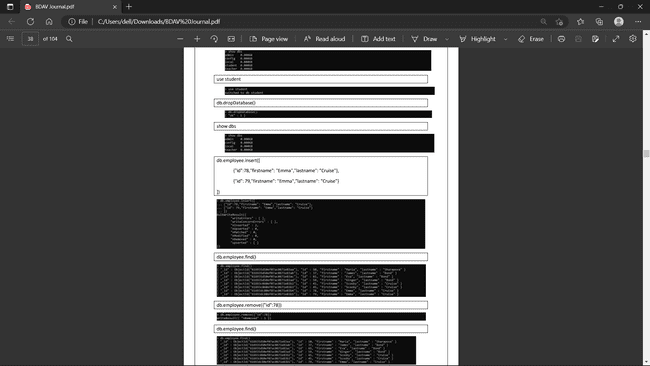


**d) Delete Document**

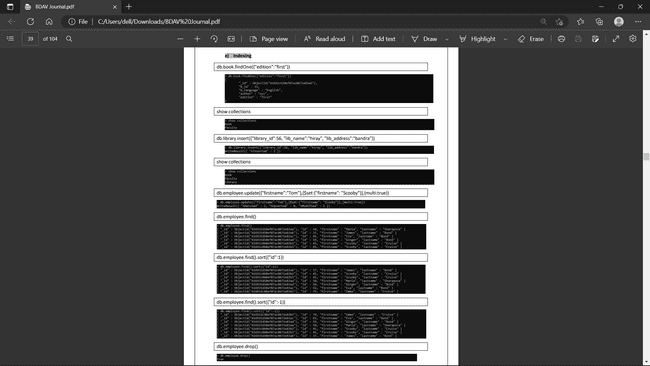
In MongoDB, the db.colloction.remove() method is used to delete documents from a collection. The remove() method works on two parameters.

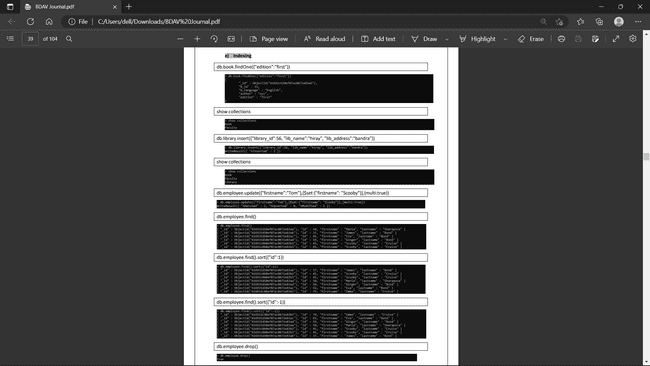
Syntax: db.collection\_name.remove (DELETION\_CRITERIA)





**e) Indexing**





db.employee.insert([

{

"userId":"rirani", "jobTitleName":"Developer", "firstName":"Romin","lastName":"Irani", "preferredFullName":"Romin Irani","employeeCode":"E1","region":"CA", "phoneNumber":"408-1234567","emailAddress":"romin.k.irani@gmail.com","age":42

},

{

"userId":"nirani","jobTitleName":"Developer","firstName":"Neil","lastName":"Irani", "preferredFullName":"Neil Irani","employeeCode":"E2","region":"CA", "phoneNumber":"408-1111111","emailAddress":"neilrirani@gmail.com","age":36

},

{

"userId":"thanks","jobTitleName":"Program Directory","firstName":"Tom","lastName":"Hanks", "preferredFullName":"Tom Hanks","employeeCode":"E3","region":"CA", "phoneNumber":"408-2222222","emailAddress":"tomhanks@gmail.com","age":34

},

{

"userId":"rirani","jobTitleName":"Developer","firstName":"Romin","lastName":"Irani", "preferredFullName":"Romin Irani","employeeCode":"E1","region":"CB", "phoneNumber":"408-1234567","emailAddress":"romin.k.irani@gmail.com","age":45

},

{

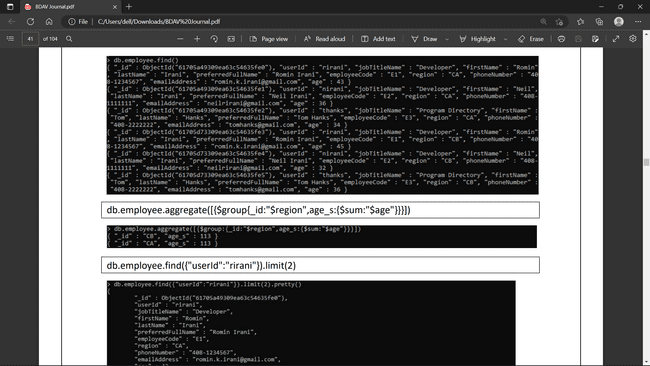
"userId":"nirani","jobTitleName":"Developer","firstName":"Neil","lastName":"Irani", "preferredFullName":"Neil Irani","employeeCode":"E2","region":"CB", "phoneNumber":"408-1111111","emailAddress":"neilrirani@gmail.com","age":32

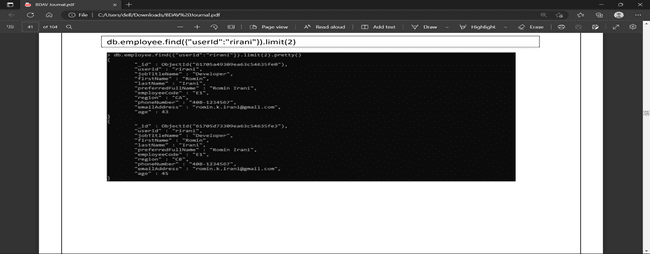
},

{

"userId":"thanks","jobTitleName":"Program Directory","firstName":"Tom","lastName":"Hanks", "preferredFullName":"Tom Hanks","employeeCode":"E3","region":"CB", "phoneNumber":"408-2222222","emailAddress":"tomhanks@gmail.com","age":36

}])





**Chapter No.4 Hadoop Eco-system (HIVE and PIG)**

**4.1 INTRODUCTION**

**HIVE**

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

**Features of Hive**

These are the following features of Hive:

* Hive is fast and scalable.
* It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
* It is capable of analyzing large datasets stored in HDFS.
* It allows different storage types such as plain text, RCFile, and HBase.
* It uses indexing to accelerate queries.
* It can operate on compressed data stored in the Hadoop ecosystem.
* It supports user-defined functions (UDFs) where user can provide its functionality.

**Limitations of Hive**

* Hive is not capable of handling real-time data.
* It is not designed for online transaction processing.
* Hive queries contain high latency.

**PIG**

Apache Pig is a high-level data flow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using PIG can also be achieved using java used in MapReduce.

## Advantages of Apache Pig

* Less code - The Pig consumes less line of code to perform any operation.
* Reusability - The Pig code is flexible enough to reuse again.
* Nested data types - The Pig provides a useful concept of nested data types like tuple, bag, and map.

**4.2 Hardware Required**

* Windows 10,11,etc.
* 8 GB of RAM
* i3 Processor
* 64-bit operating system, x64-based processor

**4.3 Software Required**

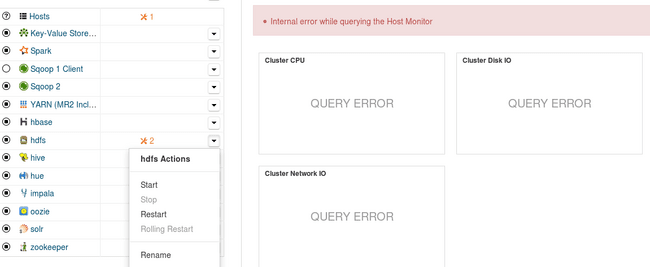
* Refer Page No. 5

**4.4 Installation and Configuration**

* Refer Page No. 5

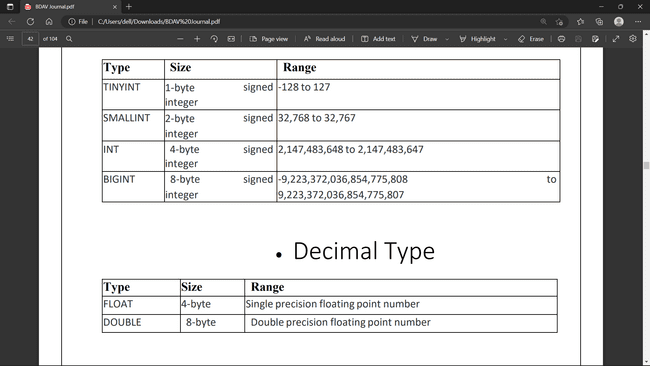
**4.5 PRACTICE**

To perform practical of Hive repeat steps mentioned in page no. 15 to 20 and start below services as shown in figure below. For Hive need to start Hive service.

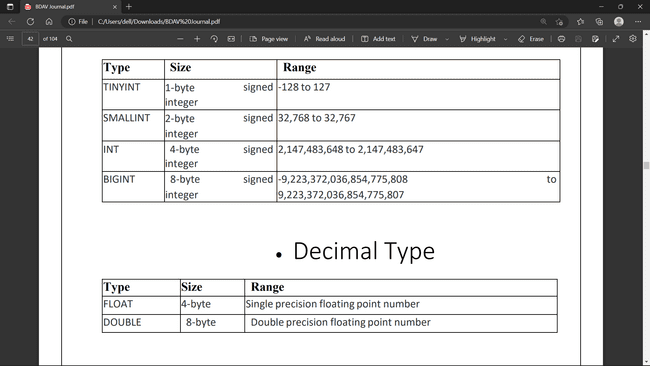


**Introduction of HIVE data types**

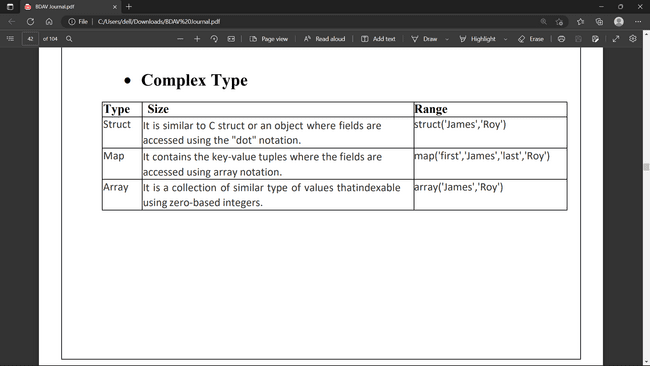
**• Integer Types**



**• Decimal Type**



**• Complex**



**• String Types**

**STRING** The string is a sequence of characters. It values can be enclosed within single quotes (') or double quotes (").

**Varchar** The varchar is a variable length type whose range lies between 1 and 65535, which specifies that the maximum number of characters allowed in the character string.

**CHAR** The charis a fixed-length type whose maximum length is fixed at 255.

**• Date/Time Types**

**TIMESTAMP**

¬ It supports traditional UNIX timestamp with optional nanosecond precision.

¬ As Integer numeric type, it is interpreted as UNIX timestamp in seconds.

¬ As Floating point numeric type, it is interpreted as UNIX timestamp in seconds with decimal precision.

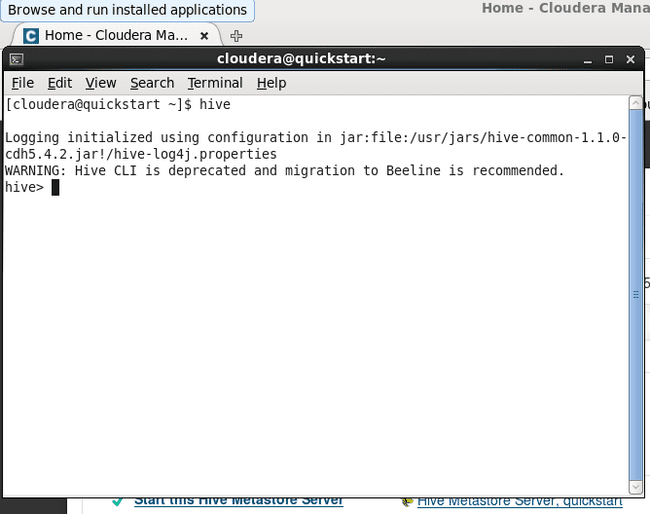
¬ As string, it follows java.sql.Timestamp format "YYYY-MM-DD HH:MM:SS.fffffffff" (9 decimal place precision)

**DATES**

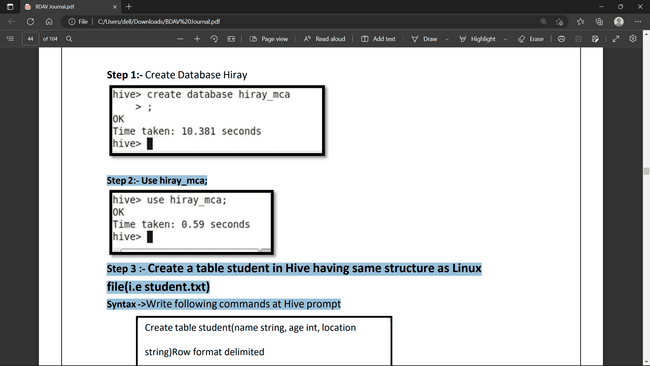
The Date value is used to specify a particular year, month and day, in the form YYYY--MM--DD. However, it didn't provide the time of the day. The range of Date type lies between 0000--01--01 to 9999--12--31.

**Practical 1 :- Implementation of Create Database & Table in HIVE**

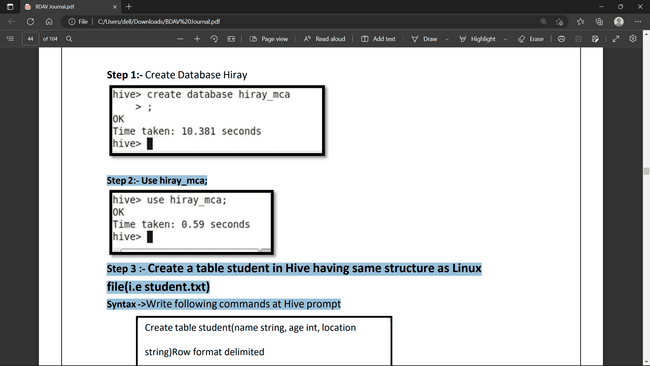
**Step1:-** Type Hive at Linux shell and press enter.



**Step 2 :-** Create Database Hiray

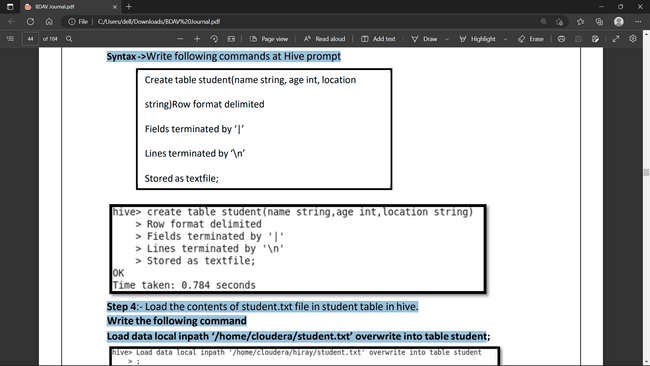


Step 2:- Use hiray\_mca;



Step 3 :- Create a table student in Hive having same structure as Linux file(i.e student.txt)

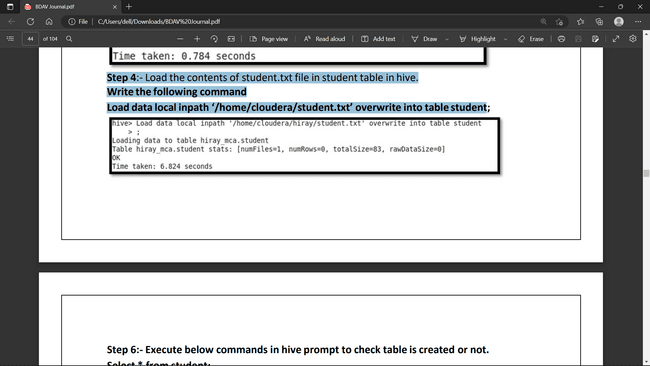
Syntax ->Write following commands at Hive prompt



Step 4:- Load the contents of student.txt file in student table in hive.

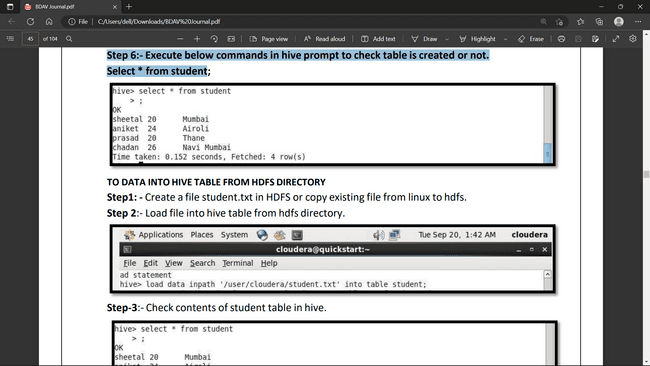
Write the following command

Load data local inpath ‘/home/cloudera/student.txt’ overwrite into table student



Step 6:- Execute below commands in hive prompt to check table is created or not.

Select \* from student;



**LOAD DATA INTO HIVE TABLE FROM HDFS DIRECTORY**

Step1: - Create a file student.txt in HDFS or copy existing file from linux to hdfs.

Step 2:- Load file into hive table from hdfs directory.



Step-3:- Check contents of student table in hive.

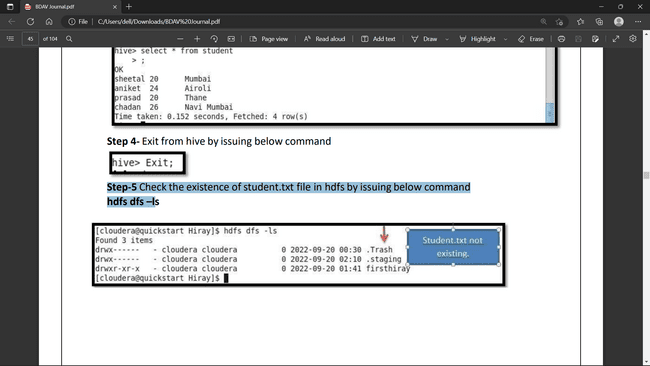


Step 4- Exit from hive by issuing below command



Step-5 Check the existence ofstudent.txt file in hdfs by issuing below

command hdfs dfs –ls



**Practical 2: Implementation of HIVE Partitioning**

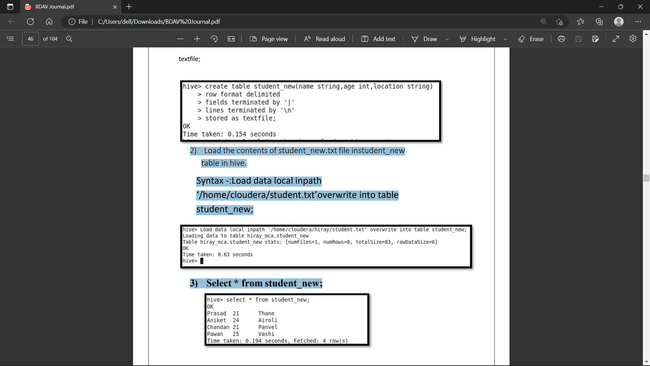
Create table student\_new(name string, age int, location string)Row format

delimited

Fields terminated by ‘|’ Lines

terminated by ‘\n’Stored as

textfile;

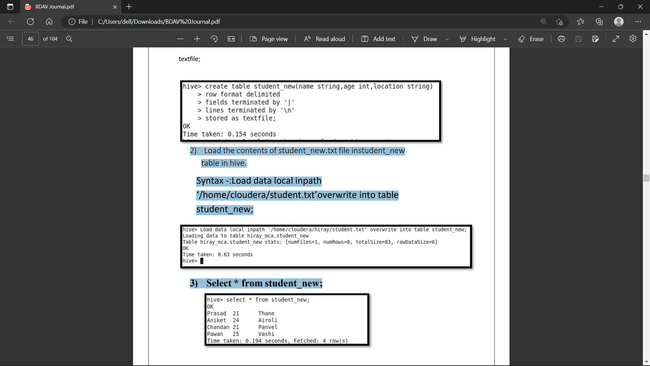


2) Load the contents of student\_new.txt file instudent\_new table in hive.

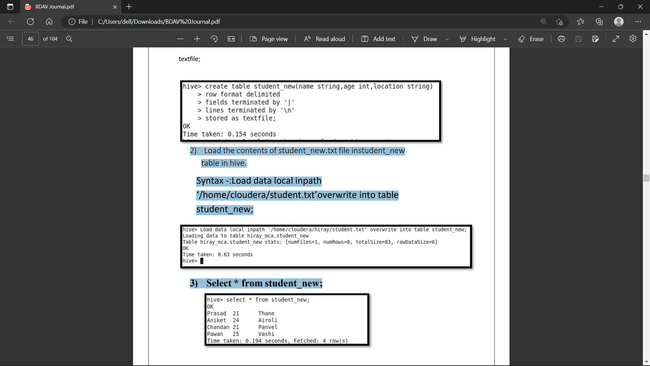
Syntax -:Load data local inpath

‘/home/cloudera/student.txt’overwrite into table

student\_new;

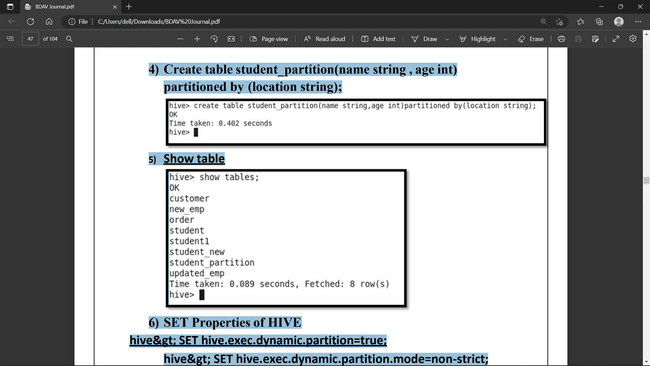


3) Select \* from student\_new;

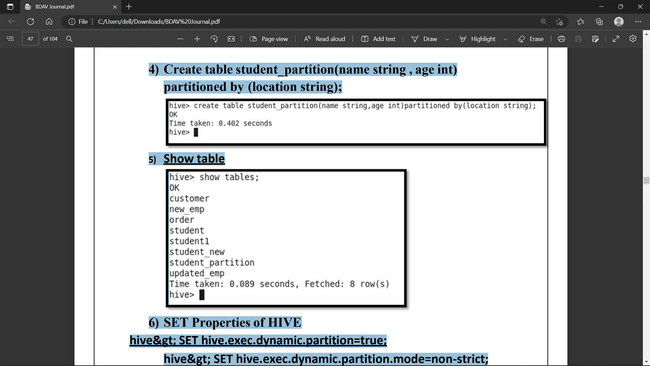


4) Create table student\_partition(name string , age int)

partitioned by (location string);



5) Show table

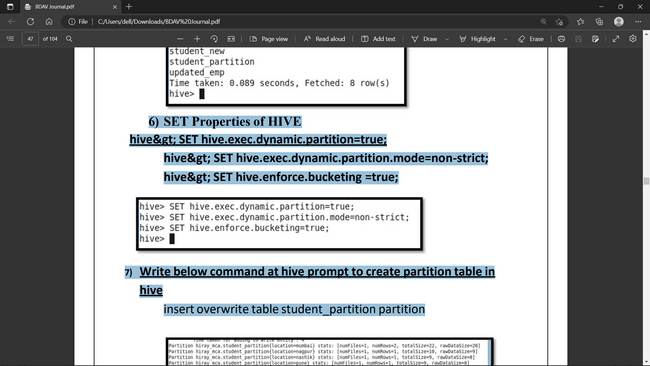


6) SET Properties of HIVE

hive> SET hive.exec.dynamic.partition=true;

Hive> SET hive.exec.dynamic.partition.mode=non-strict;

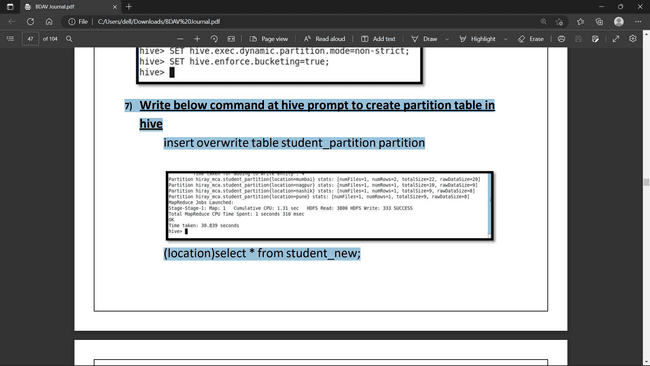
Hive> SET hive.enforce.bucketing =true;



7) Write below command at hive prompt to create partition table in hive

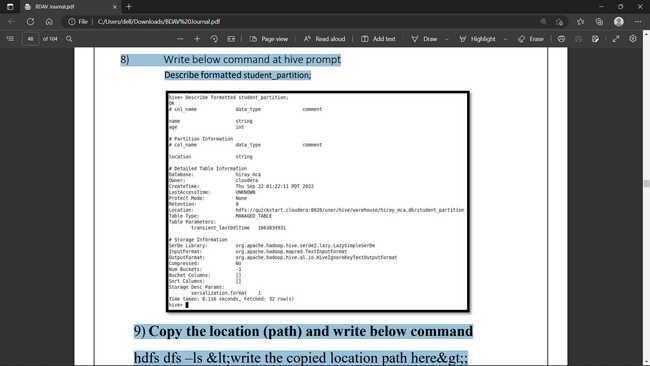
insert overwrite table student\_partition

partition (location)select \* from student\_new;



8) Write below command at hive prompt

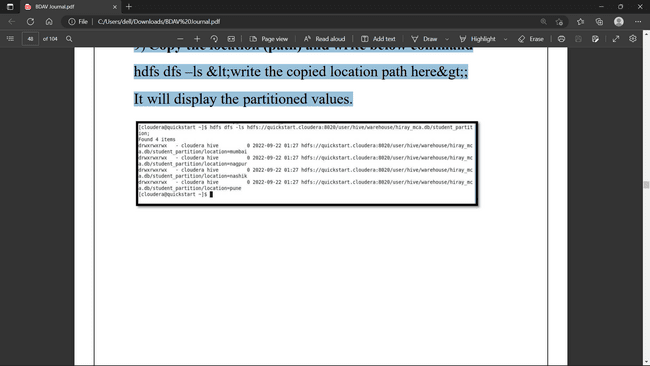
Describe formatted student\_partition;



9) Copy the location (path) and write below command

hdfs dfs –ls&lt; write the copied location path here&gt;

It will display the partitioned values.



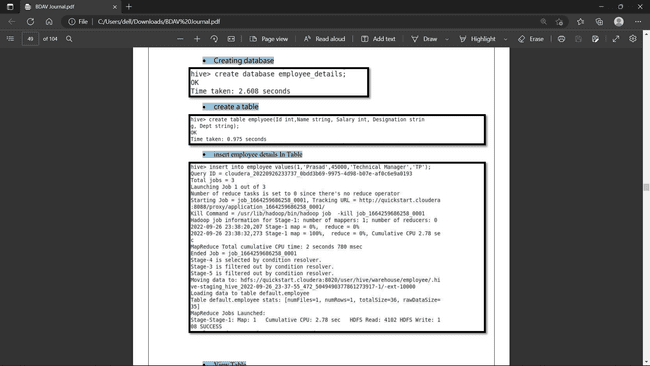
**Practical 3 : Implementation of HIVE Built –in Operators**

• Relational Operators

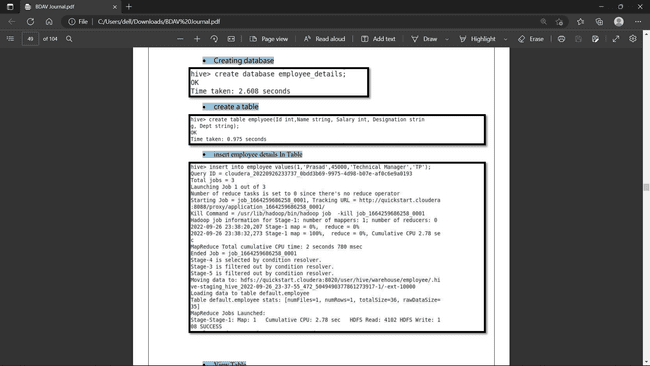
• Arithmetic Operators

Relational Operators

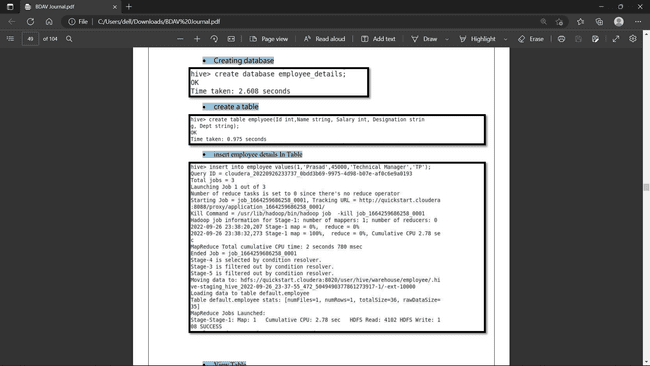
• Creating database



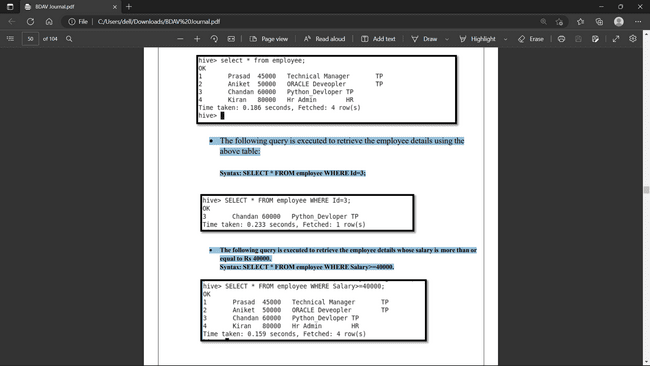
• create a table



• insert employee details In Table

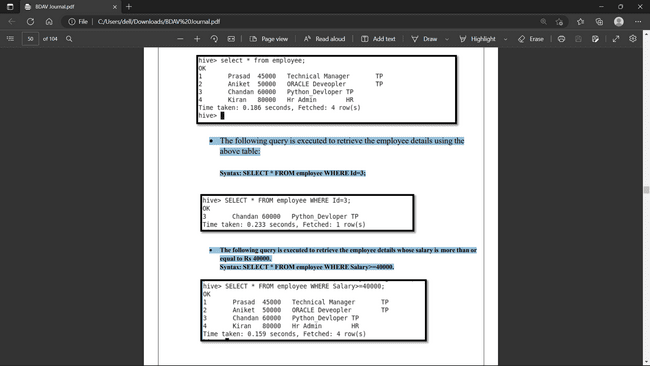


• View Table



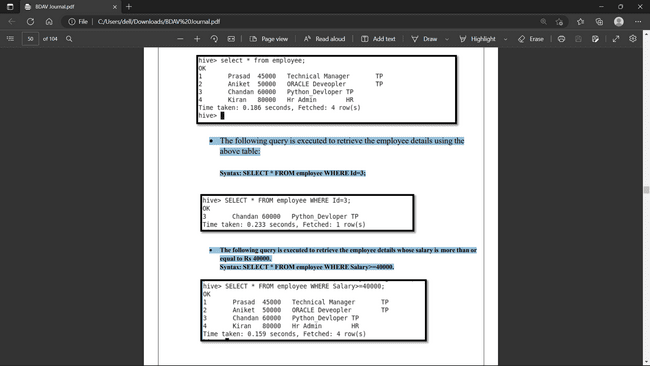
• The following query is executed to retrieve the employee details using the above table:

Syntax: SELECT \* FROM employee WHERE Id=3;



• The following query is executed to retrieve the employee details whose salary is more than or equal to Rs 40000.

Syntax: SELECT \* FROM employee WHERE Salary>=40000.

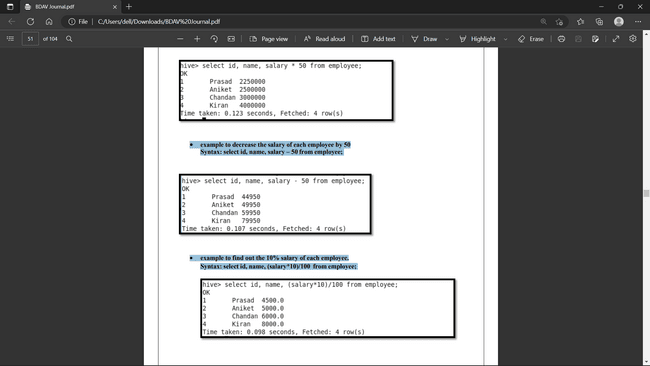


**Arithmetic Operators**

**Examples of Arithmetic Operator in Hive**

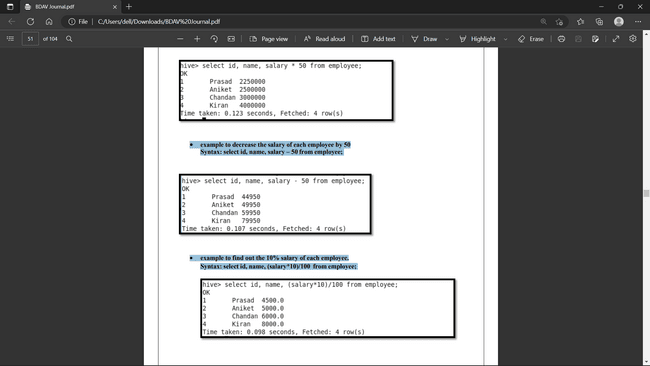
Increase the salary of each employee by 50.

Syntax: select id, name, salary \* 50 from employee;



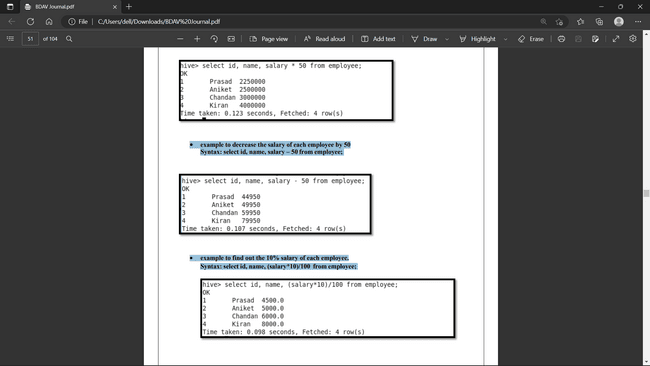
Decrease the salary of each employee by 50

Syntax: select id, name, salary – 50 from employee;



Find out the 10% salary of each employee.

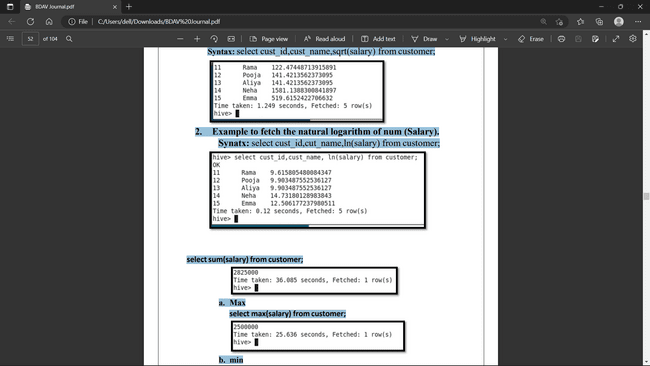
Syntax: select id, name, (salary\*10)/100 from employee;



**Practical 4 : Implementation of HIVE Built –in Functions**

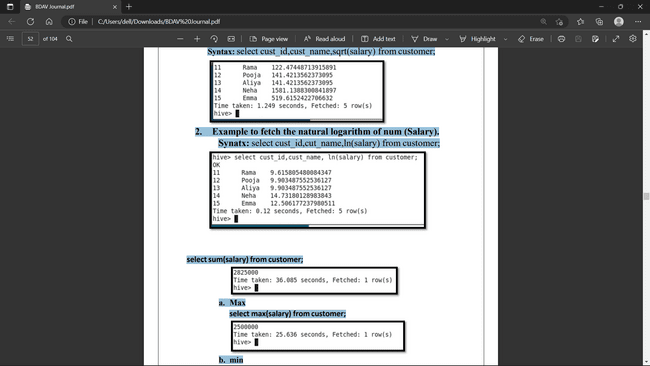
1. Fetch the square root of each customer salary.

Syntax:select cust\_id,cust\_name,sqrt(salary) from customer;

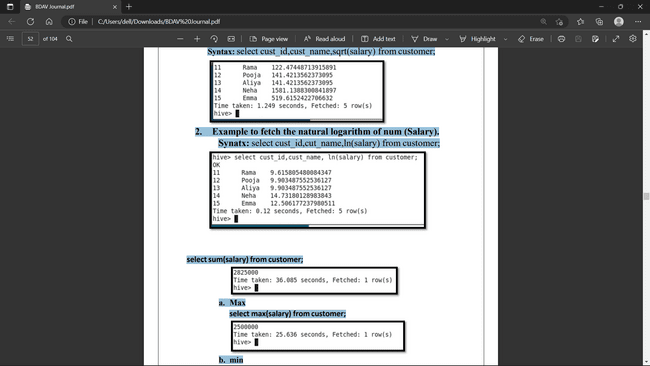


1. Fetch the natural logarithm of num (Salary).

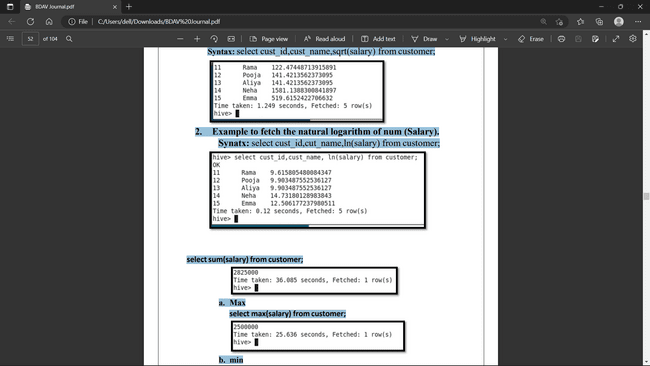
Synatx: select cust\_id,cut\_name,ln(salary) from customer;



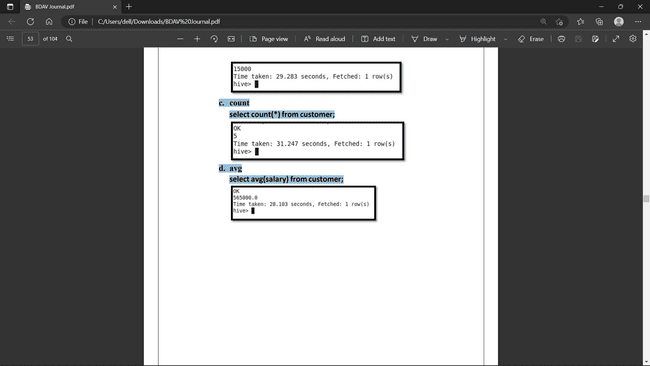
selectsum(salary) from customer;



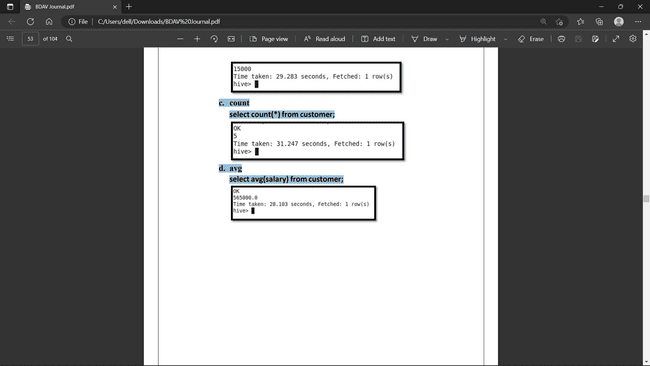
1. Max select max(salary) from customer;



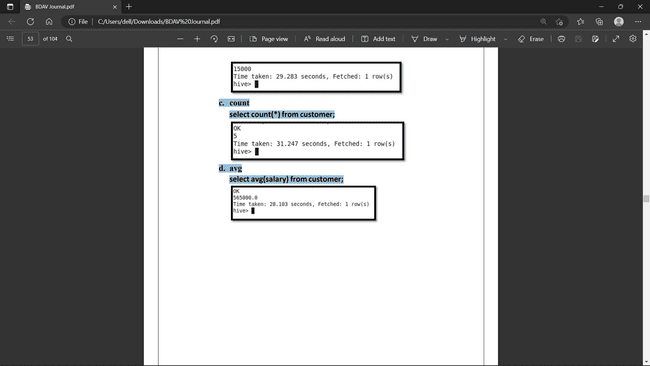
1. min select min(salary) from customer;



1. count select count(\*) from customer;



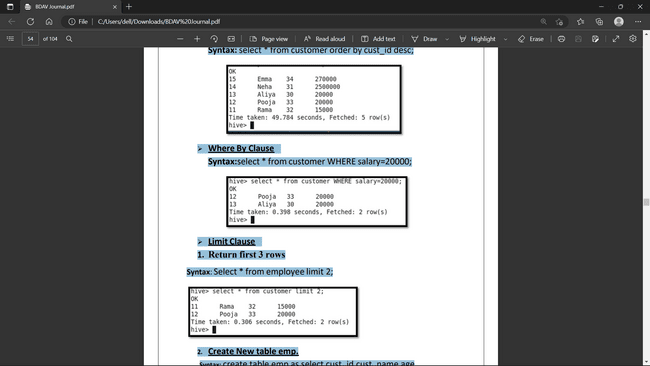
d. avg select avg(salary) from customer;



**Practical 5: Implementation of HIVE QL**

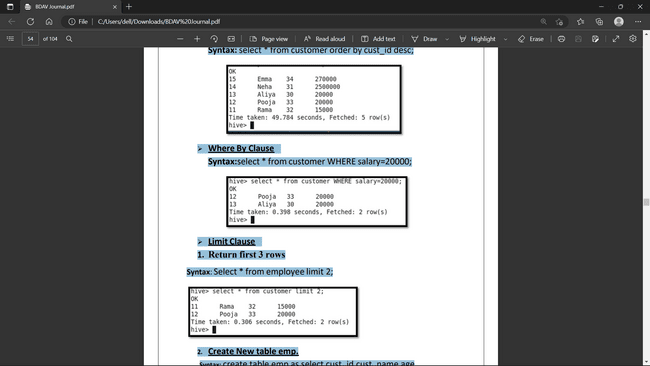
¬ Order by clause

Syntax: select \* from customer order by cust\_id desc;



¬ Where Clause

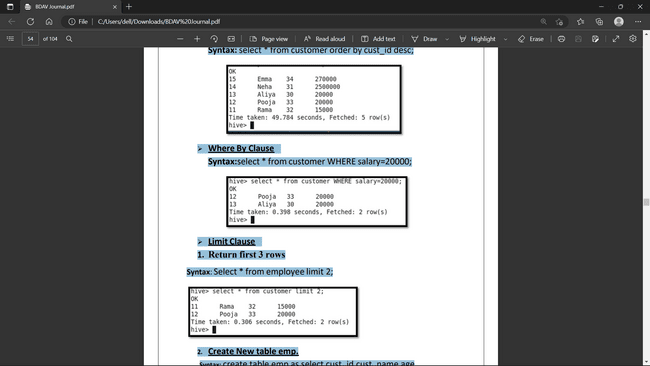
Syntax:select \* from customer WHERE salary=20000;



¬ Limit Clause

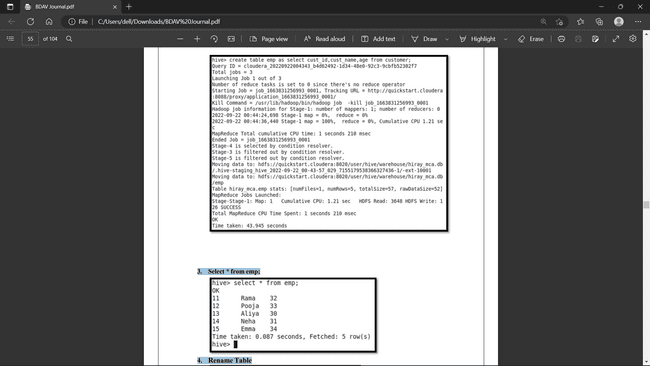
1. Return first 3 rows

Syntax: Select \* from employee limit 2;

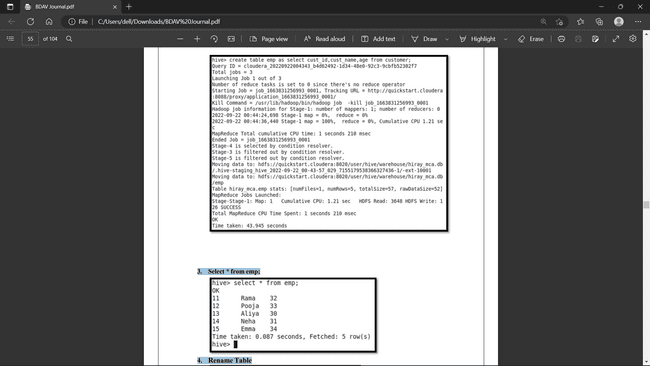


2. Create New table emp.

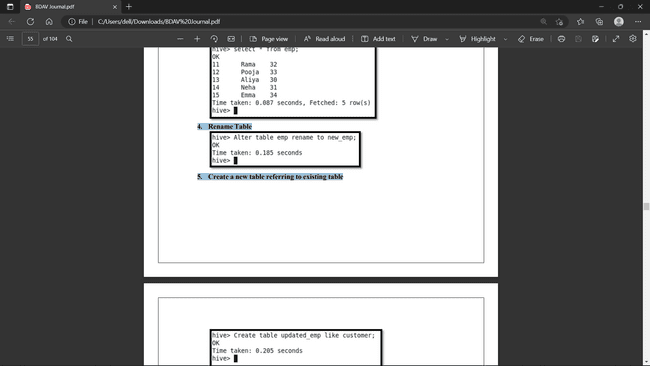
Syntax: create table emp as select cust\_id,cust\_name,age fromcustomer;



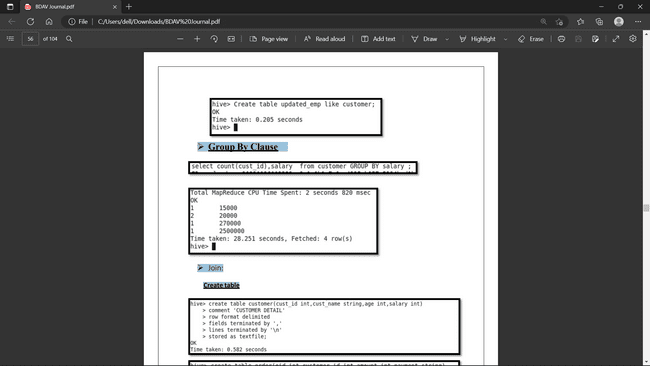
1. Select \* from emp;



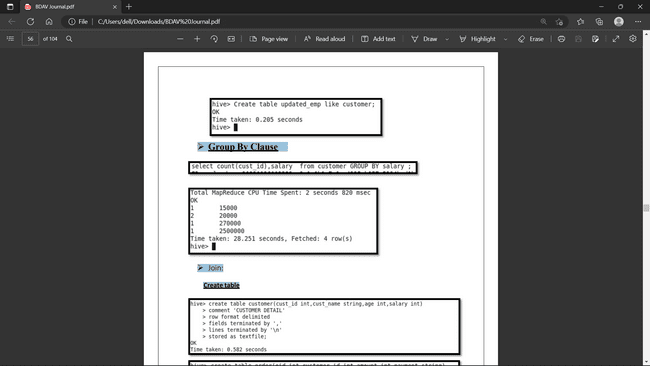
1. Rename Table



1. Create a new table referring to existing table

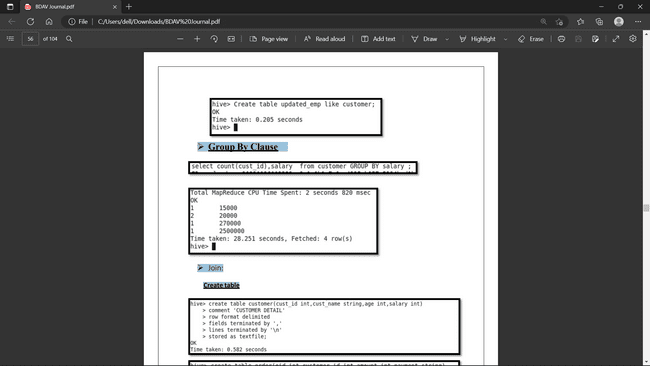


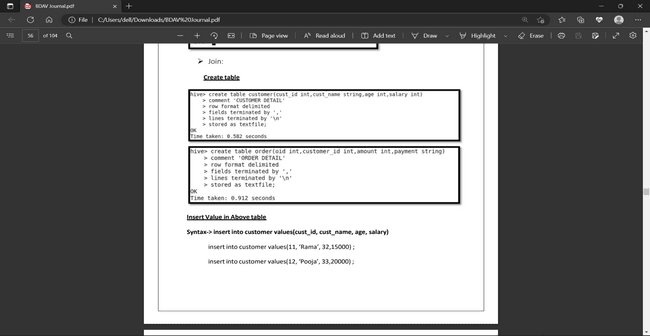
¬ Group By Clause



¬ Join:

Create table Insert Value in Above table





Syntax-> insert into customer values(cust\_id, cust\_name, age, salary)

insert into customer values(11, ‘Rama’, 32,15000) ;

insert into customer values(12, ‘Pooja’, 33,20000) ;

insert into customer values(13, ‘Aliya’, 30,20000) ;

insert into customer values(14, ‘Neha’, 31,2500000) ;

insert into customer values(15, ‘Emma’, 33,27000) ;

Syntax-> insert into order values(oid, customer\_id, amount,payment)

insert into order values(101, 11, 2345,’Cash’) ;

insert into order values(102, 13, 1150,’Cash’) ;

insert into order values(103, 14, 2410,’Card’) ;

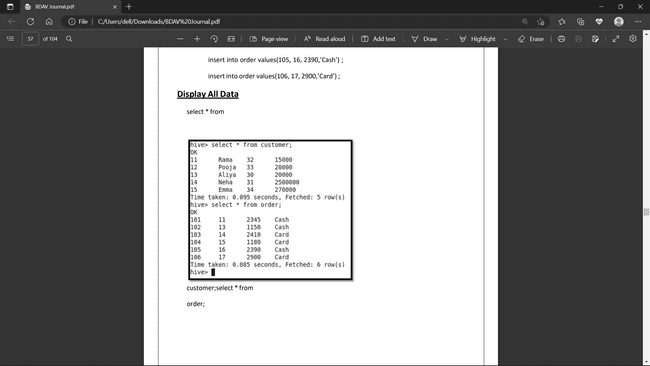
insert into order values(104, 15, 1180,’Card’) ;

insert into order values(105, 16, 2390,’Cash’) ;

insert into order values(106, 17, 2900,’Card’) ;

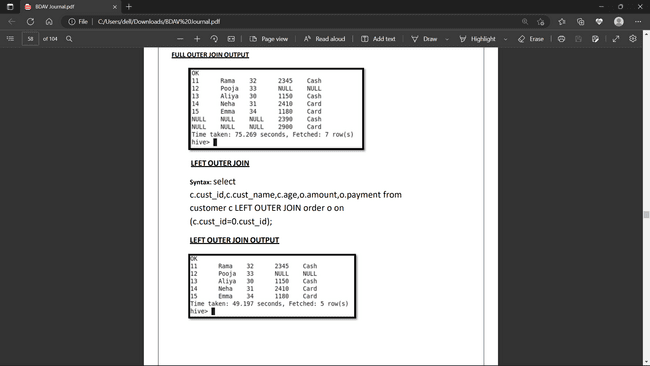
**Display All Data**

select \* from customer;select \* from order;



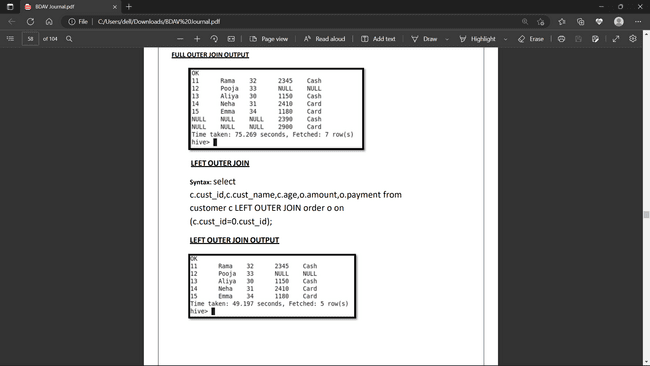
**IMPLEMENTATION OF FULL OUTER JOIN**

select c.cust\_id,c.cust\_name,c.age,o.amount,o.amount fromcustomer c FULL OUTER JOIN order o on(c.cust\_id=0.customer\_id);



**IMPLEMENTATION OF LFETOUTER JOIN**

select c.cust\_id,c.cust\_name,c.age,o.amount,o.payment from customer c LEFT OUTER JOIN order o on (c.cust\_id=0.cust\_id);



**IMPLEMENTATION OF RIGHT OUTER JOIN**

select c.cust\_id,c.cust\_name,c.age,o.amount,o.payment from customer c LEFT OUTER JOIN order o on (c.cust\_id=0.cust\_id);

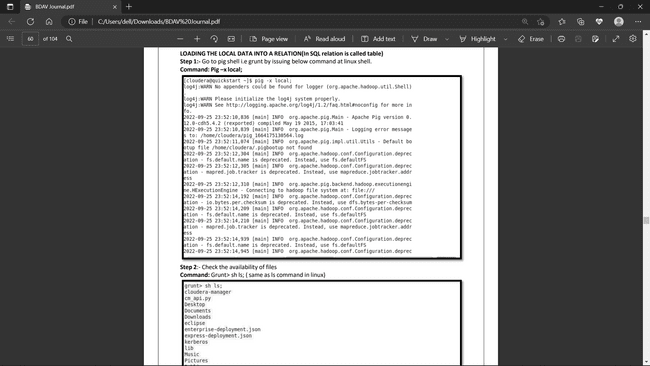


**PIG**

**LOADING THE LOCAL DATA INTO A RELATION(In SQL relation is called table)**

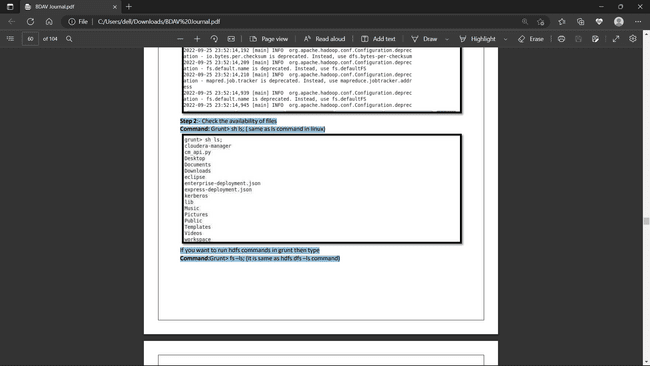
Step 1:- Go to pig shell i.e grunt by issuing below command at linux shell.

Command: Pig –x local;



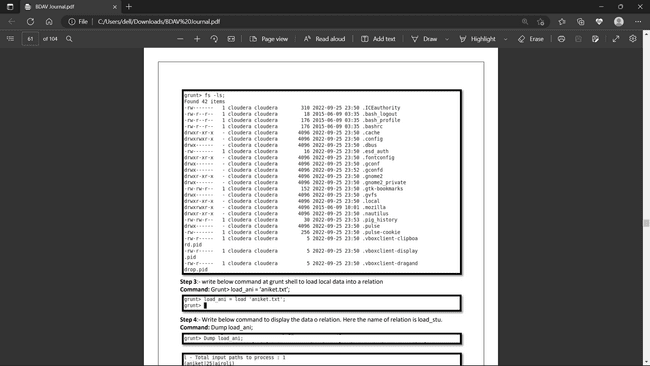
Step 2:- Check the availability of files

Command: Grunt> sh ls; (same as ls command in linux)



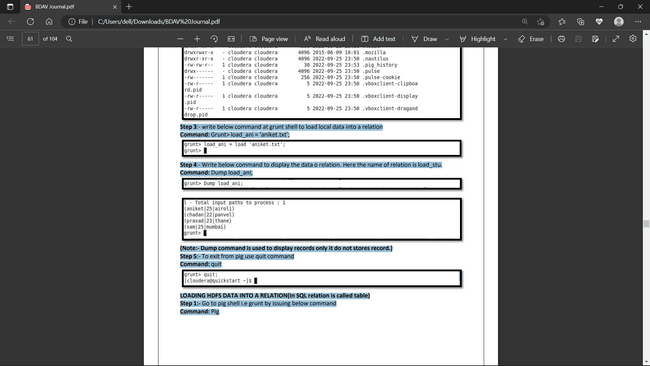
To run hdfs commands in grunt then type

Grunt> fs –ls; (it is same as hdfs dfs –ls command)

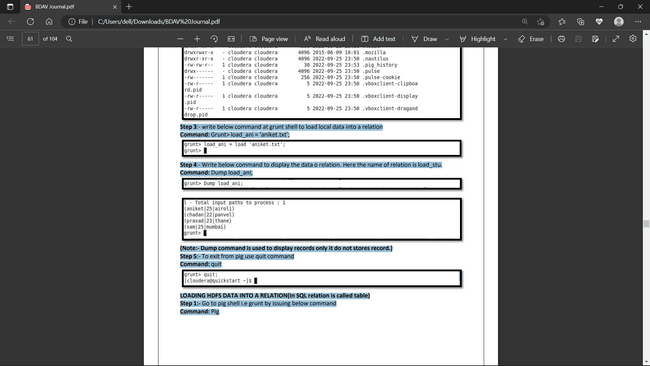


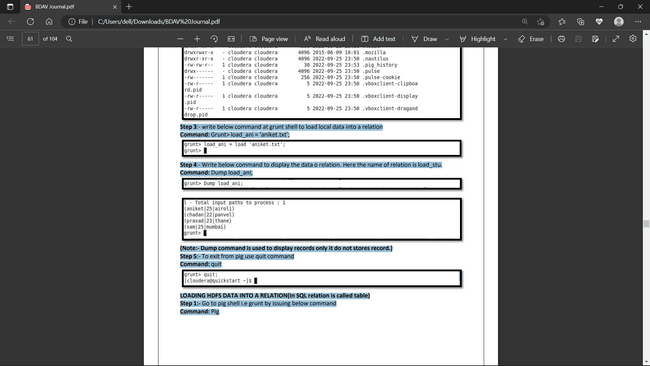
Step 3:- write below command at grunt shell to load local data into a relation

Grunt> load\_ani = ‘aniket.txt’;



Step 4:- Write below command to display the data o relation. Here the name of relation isload\_stu. Grunt> Dump load\_ani;

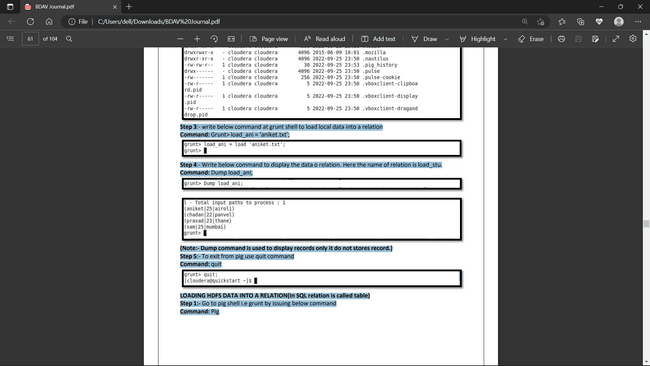




(Note:- Dump command is used to display records only it do not stores record.)

Step 5:- To exit from pig use quit command

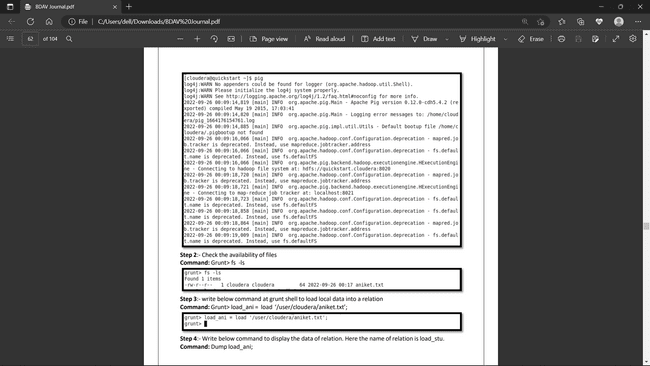
Grunt> quit



**LOADING HDFS DATA INTO A RELATION(In SQL relation is called table)**

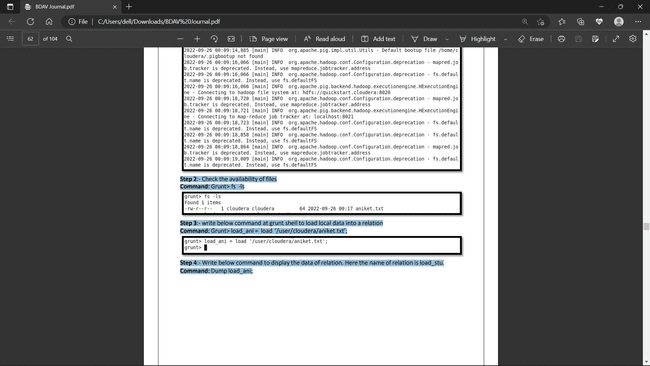
Step 1:- Go to pig shell i.e grunt by issuing below command

Command: Pig



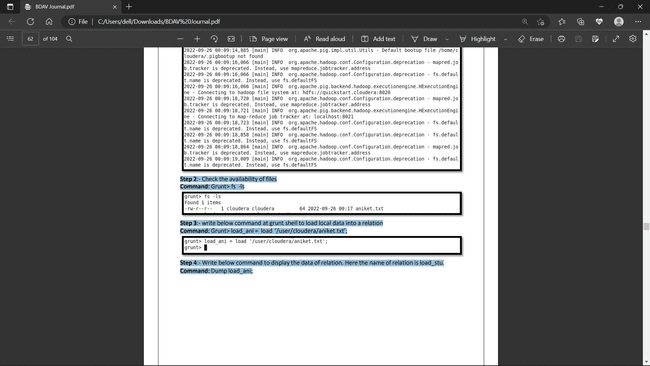
Step 2:- Check the availability of files

Grunt> fs -ls

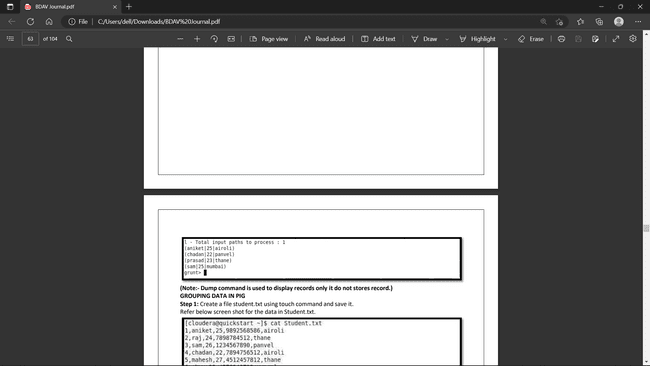


Step 3:- write below command at grunt shell to load local data into a relation

Grunt> load\_ani = load ‘/user/cloudera/aniket.txt’;



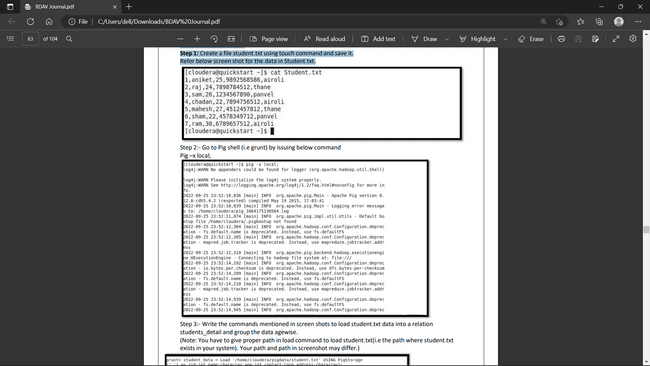
Step 4:- Write below command to display the data of relation. Here the name ofrelation is load\_stu. Command: Dump load\_ani;



**(Note:- Dump command is used to display records only it do not stores record.)**

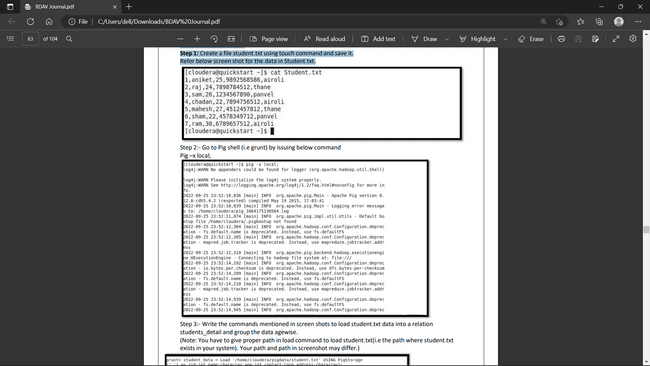
**GROUPING DATA IN PIG**

Step 1: Create a file student.txt using touch command and save it. Refer below screen shot for the data in Student.txt.



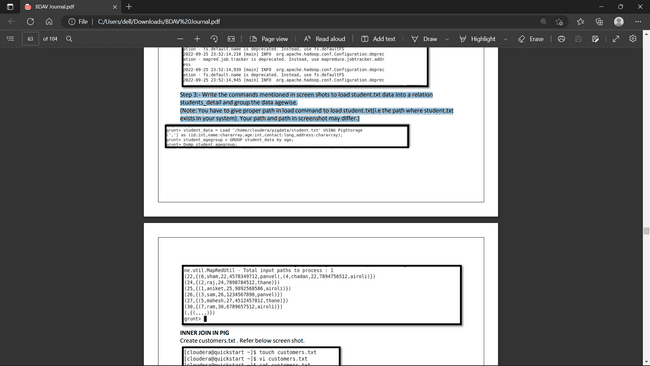
Step 2:- Go to Pig shell (i.e grunt) by issuing below command

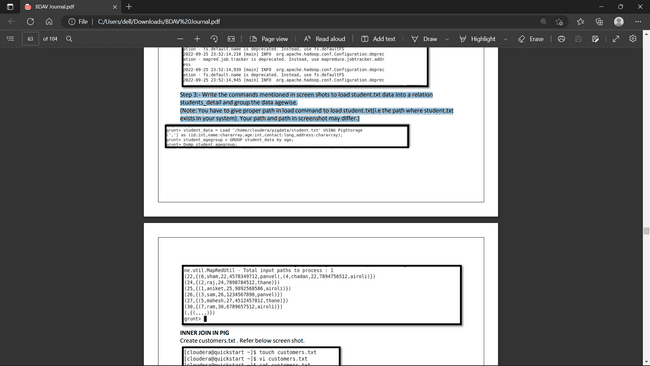
Pig –x local;



Step 3:- Write the commands mentioned in screen shots to load student.txt data into a relation students\_detail and group the data agewise.

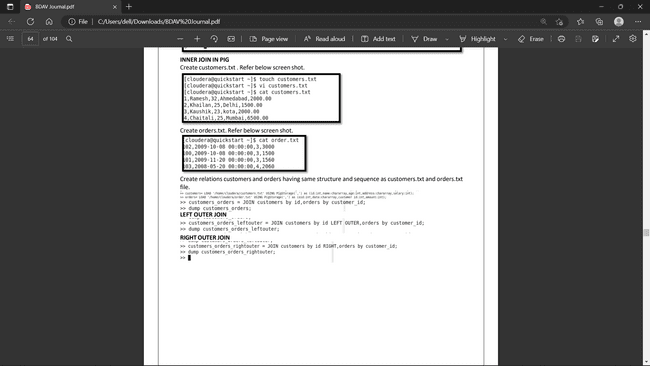
(Note: You have to give proper path in load command to load student.txt(i.e the path where student.txt existsin your system). Your path and path in screenshot may differ.



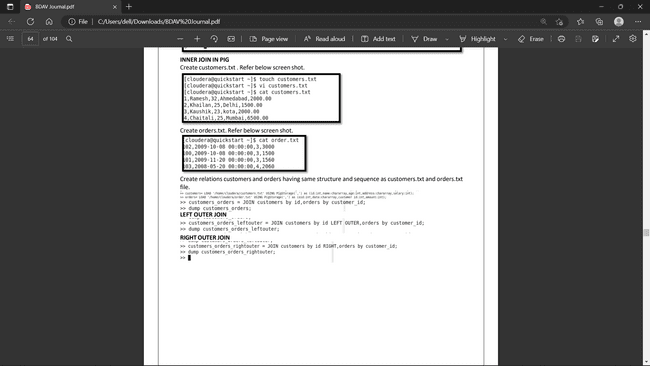


**INNER JOIN IN PIG**

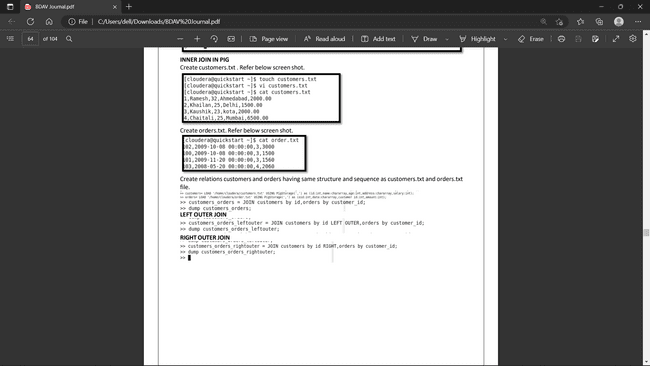
Create customers.txt . Refer below screen shot.



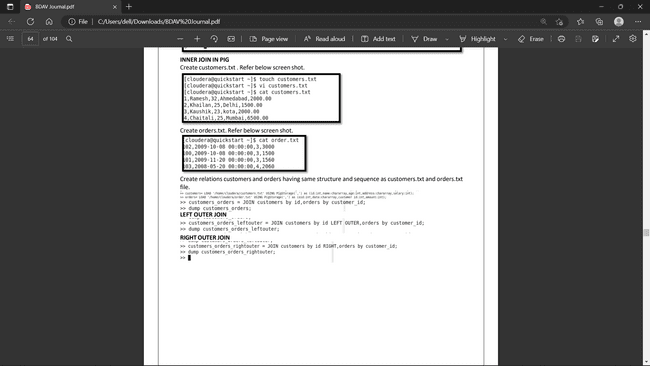
Create orders.txt. Refer below screen shot.



Create relations customers and orders having same structure and sequence as customers.txt and orders.txt file.



**LEFT OUTER JOIN**



**PIG FILTERING COMMAND**

STEP 1:- Create a file in linux Emp.txt The contents should be

Manish,Singh,1

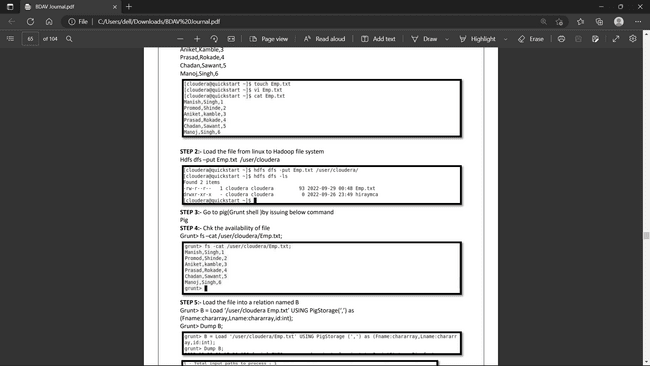
Pramod,Shinde,2

Aniket,Kamble,3

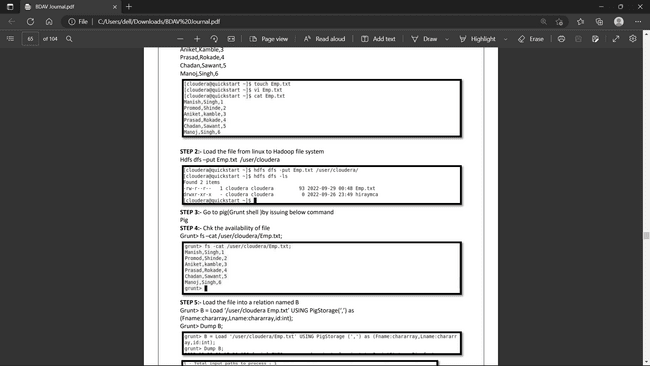
Prasad,Rokade,4

Chadan,Sawant,5

Manoj,Singh,6

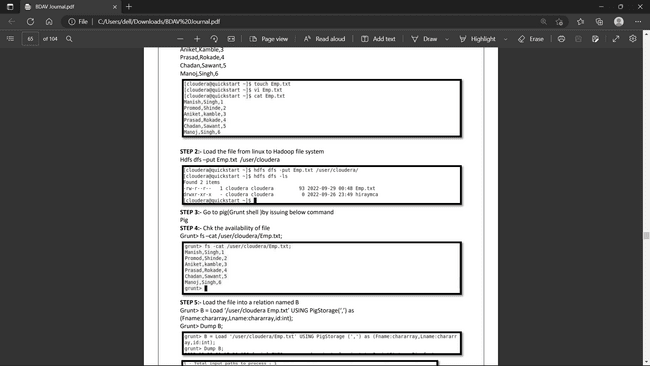


STEP 2:- Load the file from linux to Hadoop file system Hdfs dfs –put Emp.txt /user/cloudera

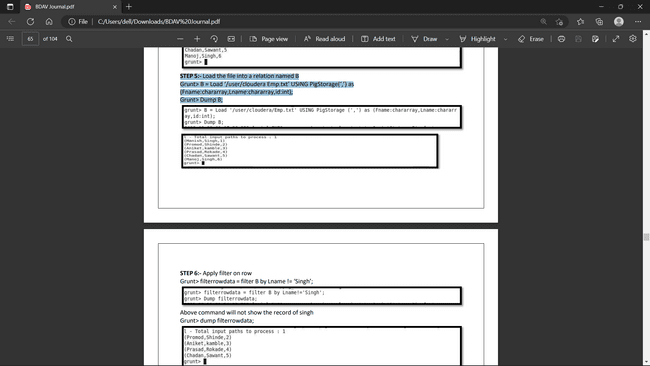


STEP 3:- Go to pig(Grunt shell )by issuing below command Pig

STEP 4:- Chk the availability of file Grunt> fs –cat/user/cloudera/Emp.txt

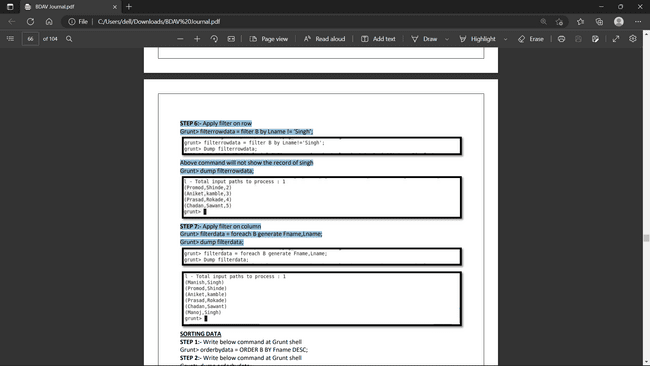


STEP 5:- Load the file into a relation named B

Grunt> B = Load ‘/user/cloudera Emp.txt’ USING PigStorage(‘,’) as (Fname:chararray,Lname:chararray,id:int); Grunt> Dump B;

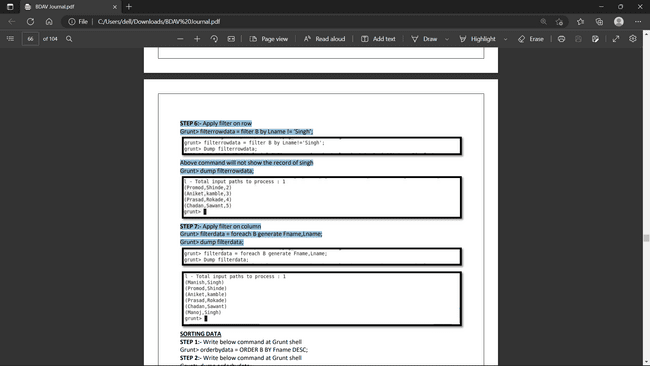
STEP 6:- Apply filter on row

Grunt> filterrowdata = filter B by Lname != ‘Singh’;



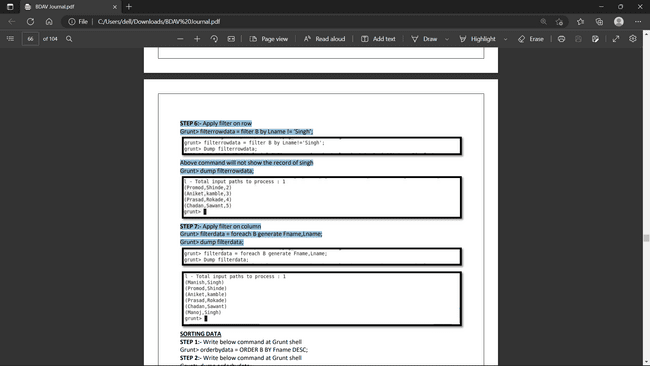
Above command will not show the record of singh

Grunt> dump filterrowdata;

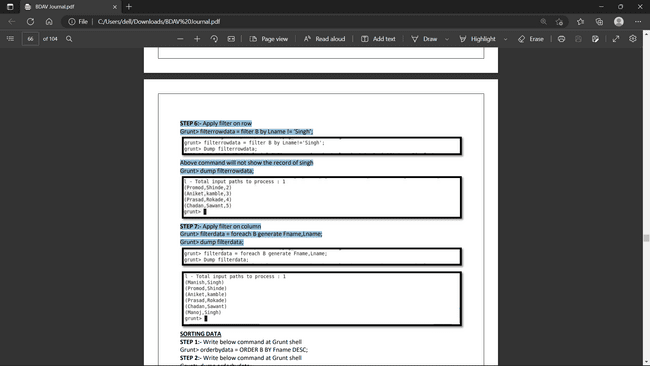


STEP 7:- Apply filter on column

Grunt> filterdata = foreach B generate Fname,Lname;



Grunt> dump filterdata;



**SORTING DATA STEP**

1:- Write below command at Grunt shell

Grunt> orderbydata = ORDER B BY Fname DESC;

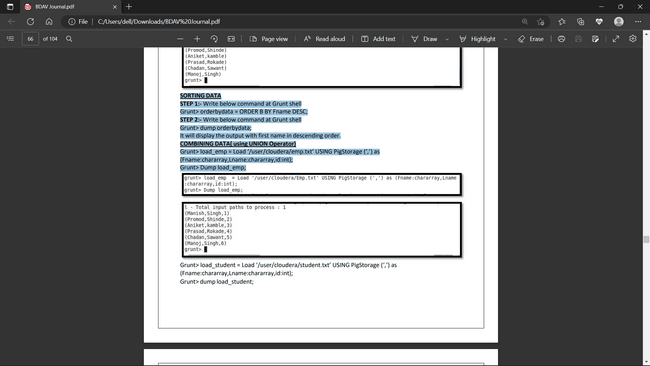
STEP 2:- Write below command at Grunt shell

Grunt> dump orderbydata;

It will display the output with first name in descending order. COMBINING DATA( using UNION Operator)

Grunt> load\_emp = Load ‘/user/cloudera/emp.txt’ USING PigStorage (‘,’) as (Fname:chararray,Lname:chararray,id:int);

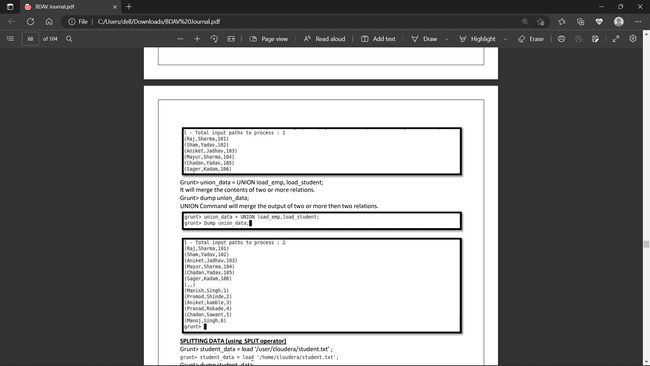
Grunt> Dump load\_emp;



Grunt> load\_student = Load ‘/user/cloudera/student.txt’ USING PigStorage (‘,’) as (Fname:chararray,Lname:chararray,id:int);

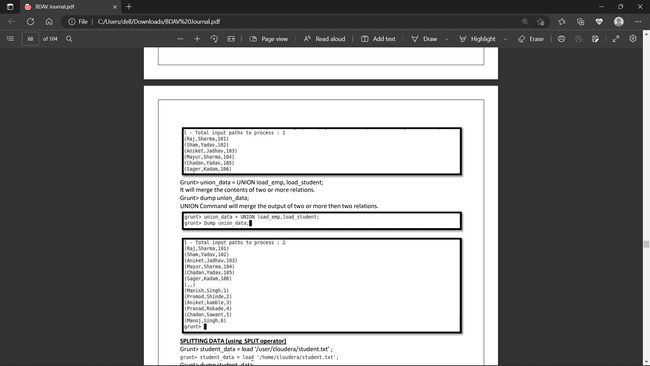
Grunt> dump load\_student;





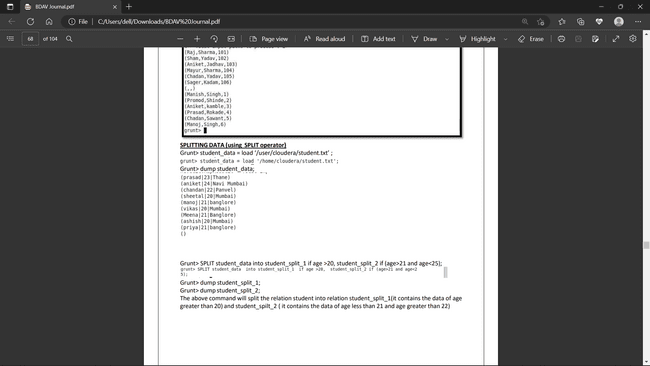
Grunt> union\_data = UNION load\_emp, load\_student; It will merge the contents of two or more relations.

Grunt> dump union\_data; UNION Command will merge the output of two or more then two relations.

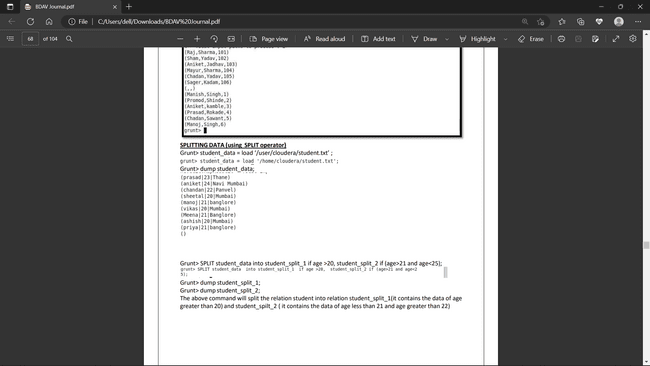


SPLITTING DATA (using SPLIT operator)

Grunt> student\_data = load ‘/user/cloudera/student.txt’;



Grunt> dump student\_data;



Grunt> SPLIT student\_data into student\_split\_1 if age >20, student\_split\_2 if (age>21 and age>25);

Grunt>dump student\_split\_1;

Grunt> dump student\_split\_2;

The above command will split the relation student into relation student\_split\_1(it contains the data of age greater than 20) and student\_spilt\_2 ( it contains the data of age less than 21 and age greater than 22)

**Chapter No.5 Data Visualization**

**5.1 Introduction**

Data visualization is the graphical representation of information and data. By using v[isual elements like charts, graphs, and maps](https://www.tableau.com/data-insights/reference-library/visual-analytics), data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Additionally, it provides an excellent way for employees or business owners to present data to non-technical audiences without confusion.

**5.2 Hardware Required**

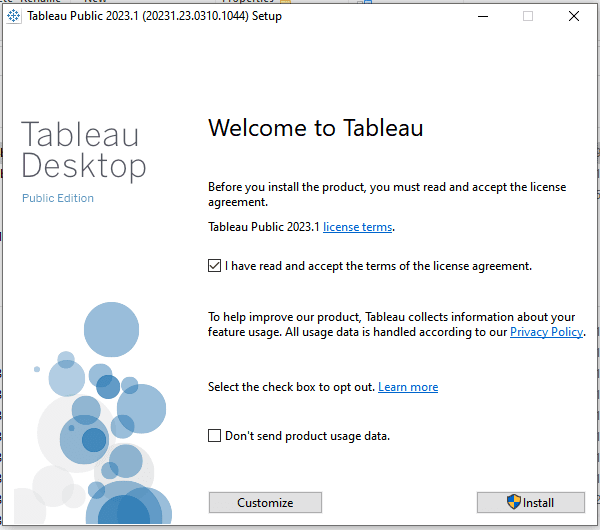
* Windows 10,11,etc.
* 8 GB of RAM
* i3 Processor
* 64-bit operating system, x64-based processor

**5.3 Software Required**

TableauPublicDesktop-64bit-2023-1-0

**5.4 Installation and Configuration**

Download tableau using link <https://www.softpedia.com/get/Office-tools/Other-Office-Tools/Tableau-Public.shtml#download> and install it.



**5.5 Practice**

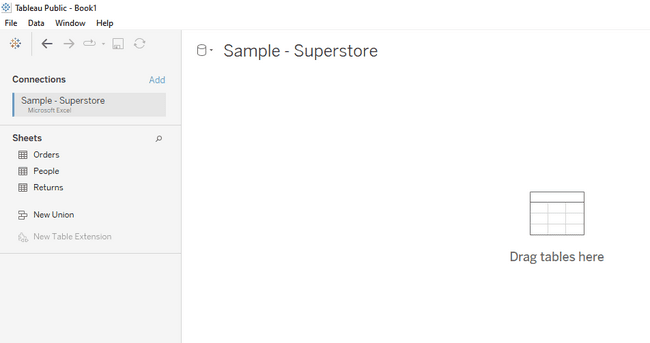
**Step 1:**  After installing Tableau double click on Tableau icon. Below screen will appear.



**Step 2: Click on Open from Tableau Public. It will ask for Sign in, So create user account and then sign up.**

**Step 3: Download sample superstore.xls file from Tableau Community.**

**Step 4: Click on Microsoft Excel on left pane and import sample superstore.xls file. Below screen will appear.**

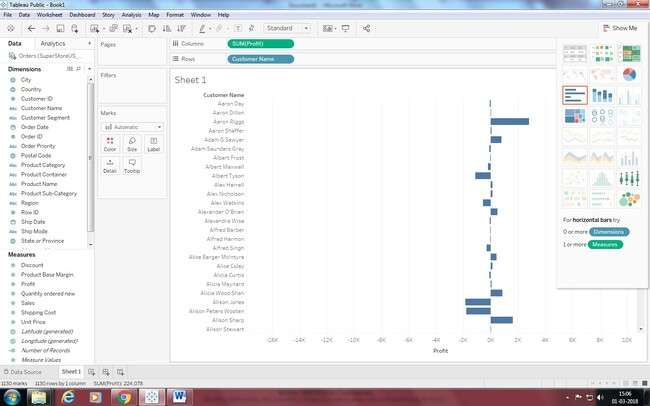
****

Assignment 1: Analysis operations

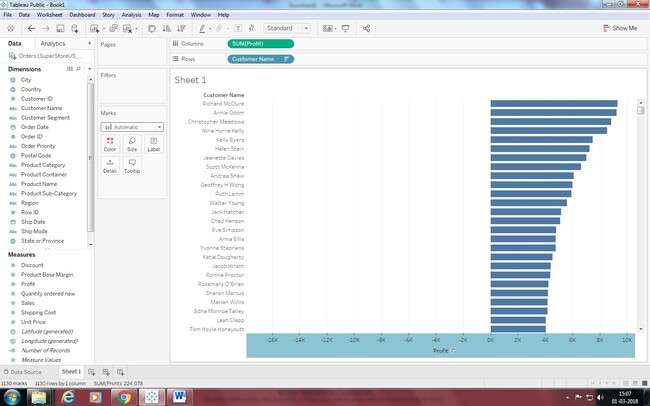
Practical 1: Find the customer with the highest overall profit. What is his/her profit ratio? Ans:

Step 1: Open the superstoreus excel data set Step 2: Drag Orders sheet to sheet area

Step 3: Go to sheet 1 and add Customer name as rows and profit as column



Step 4: Sort the data by clicking on Profit label on bottom

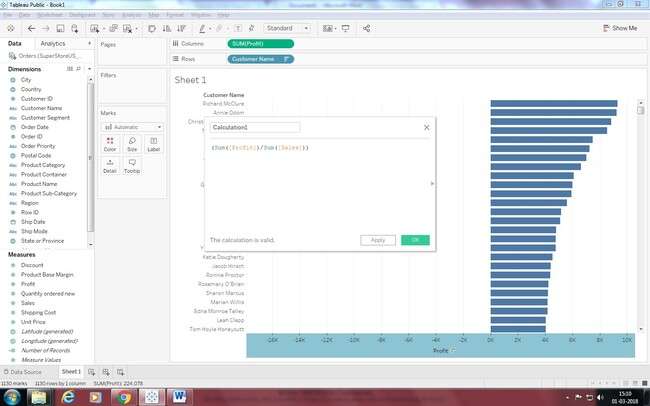


Step 5: To calculate profit Ratio

Profit Ratio= (Sum([Profit])/Sum([Sales]))

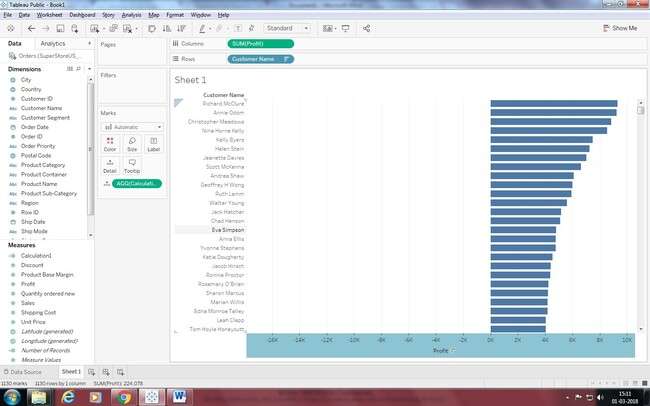
This formula needs to be entered as tooltip or label

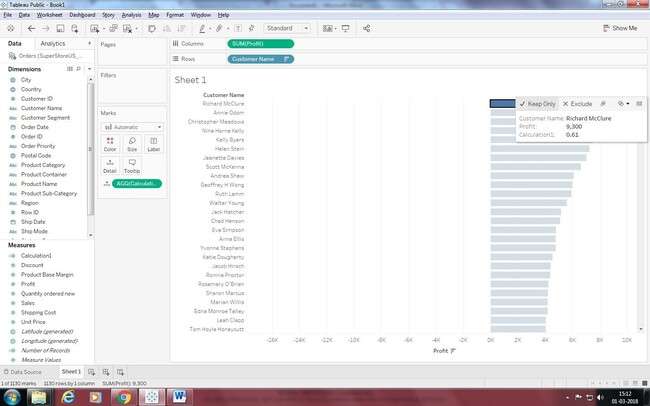
Click on Analysis>Create Calculated Field and enter the formula



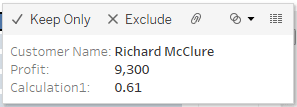


You can see Calculation1 in measures. Drag is to Marks area.

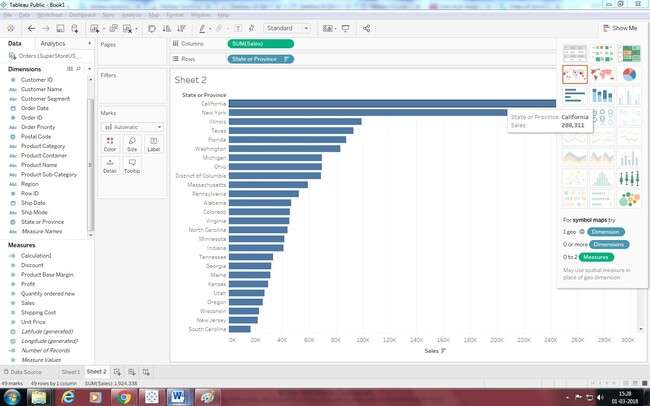




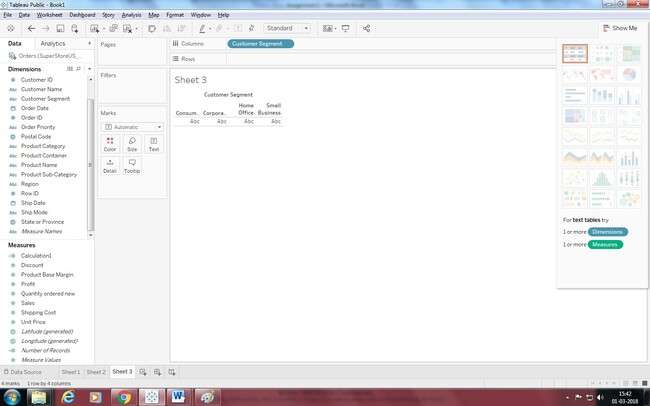
Final answer is:

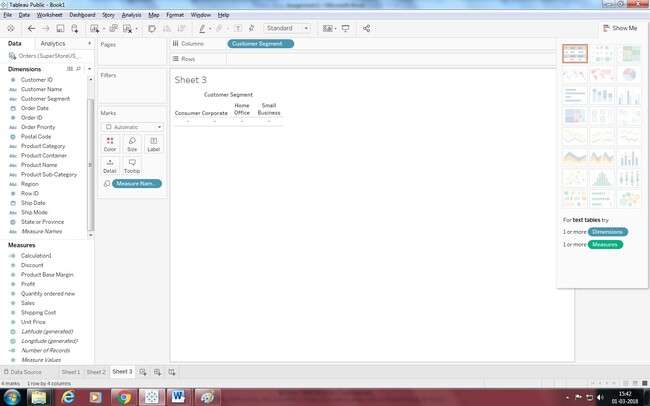


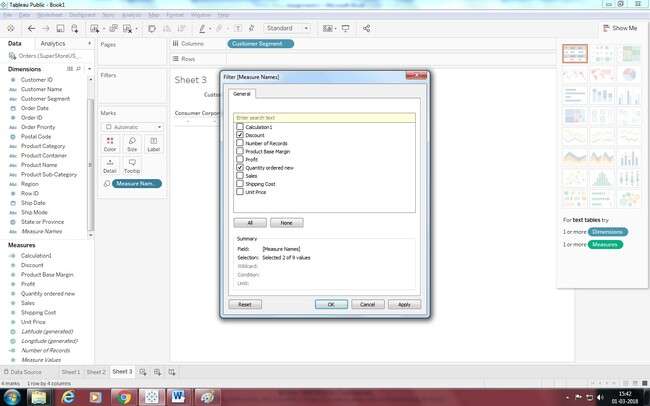
Practical 2: Which state has the highest Sales (Sum)? What is the total Sales for that state? Ans:

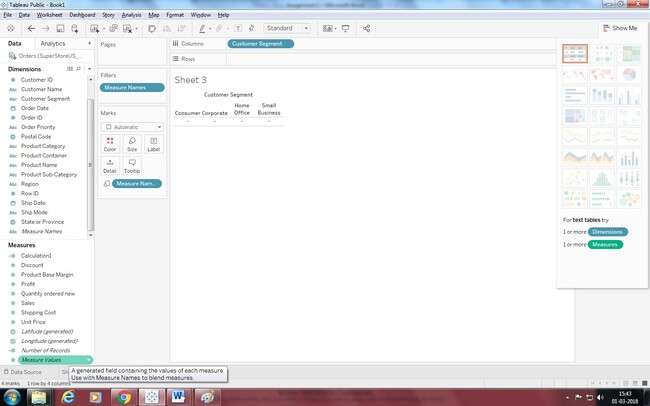


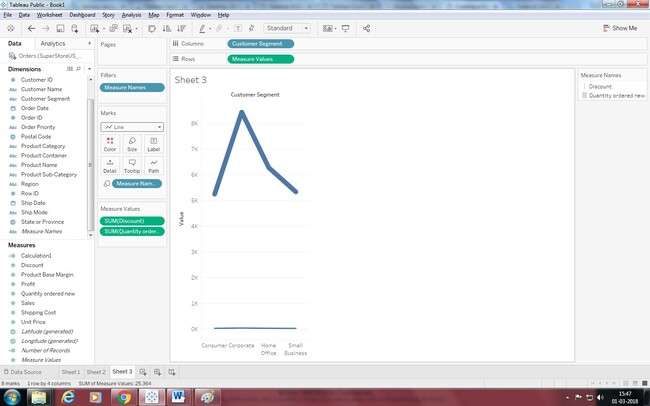
**Practical 3: Which customer segment has both the highest order quantity and average discount rate? What is the order quantity and average discount rate for that state?**







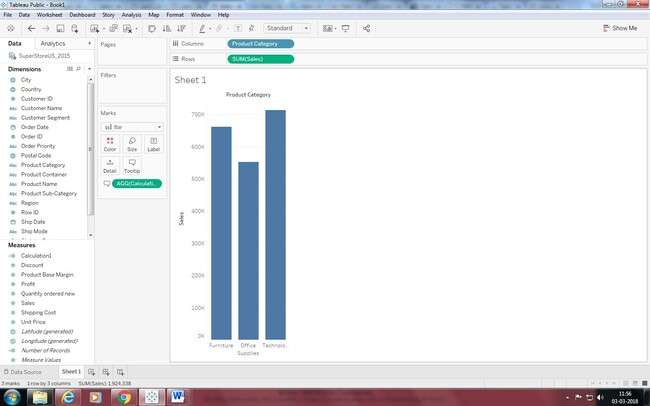


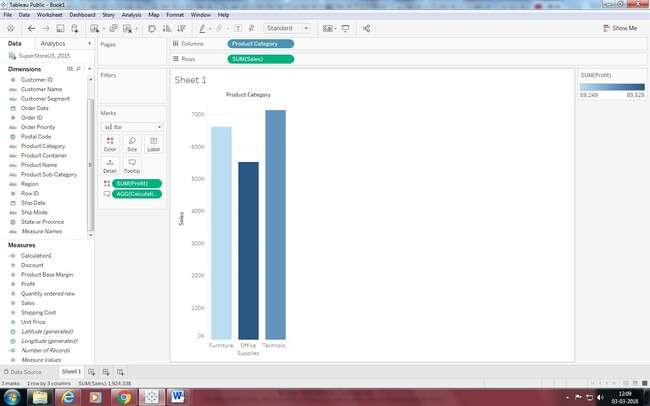


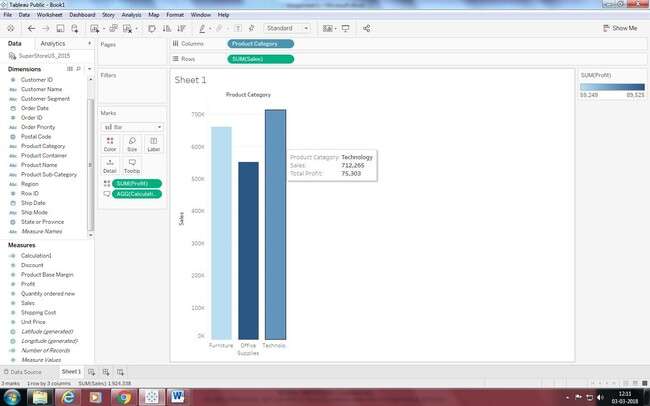
**Practical 4: Which Product Category has the highest total Sales? Which Product Category has the worst Profit? Name the Product Category and $ amount for each.**

Ans:

1. Bar Chart displaying total Sales for each Product Category

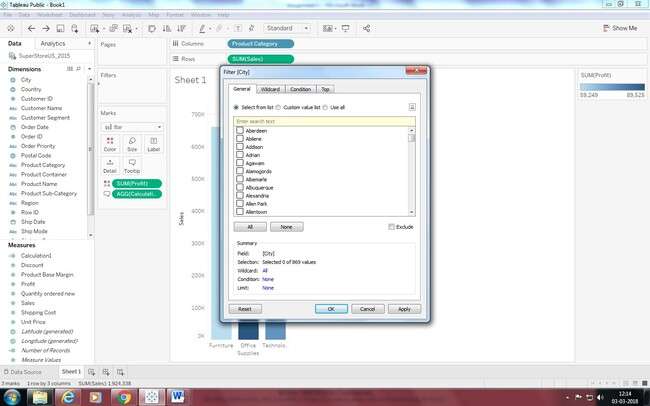


1. Add a color scale indicating Profit
2. Each Product Category labeled with total Sales and Each Product Category labeled with Profit

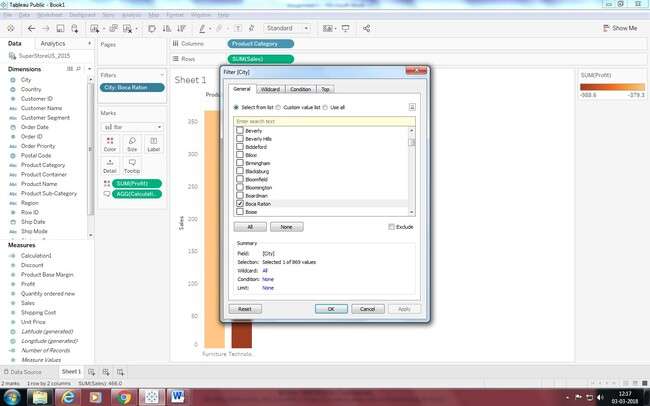


Practical 5: Use the same visualization created for Question #4.What was the Profit on Technology (Product Category) in Boca Raton (City) ?

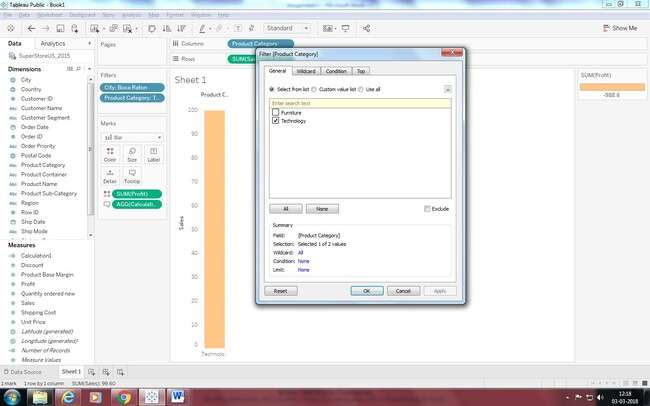
Add Filter City

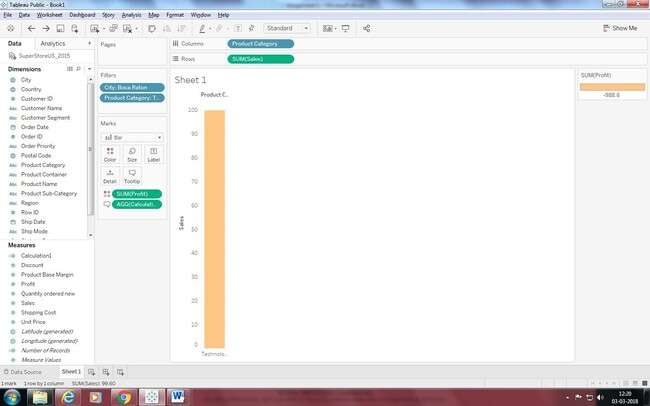


Select the city to show a single bar for the city



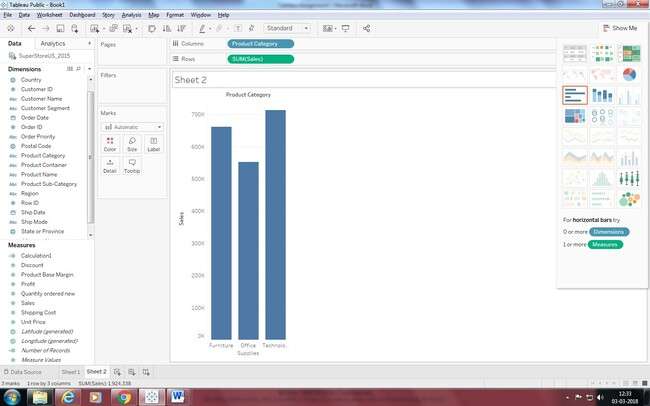
Apply a filter for Technology



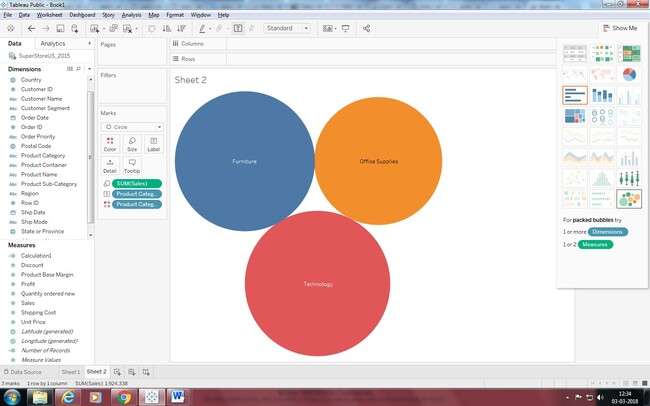


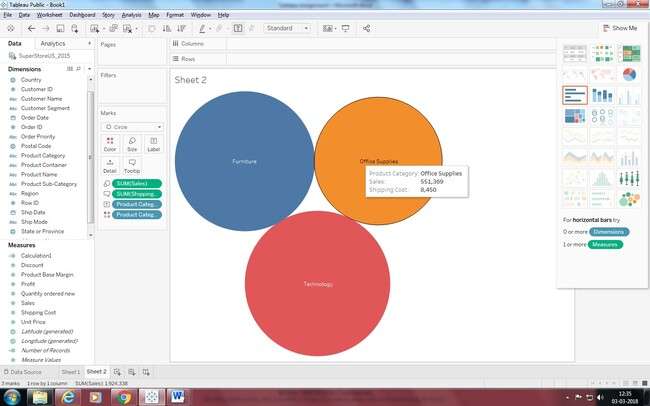
Practical 6: Which Product Department has the highest Shipping Costs? Name the Department and cost.

1. Packed bubble chart showing each Product Department as a colored bubble



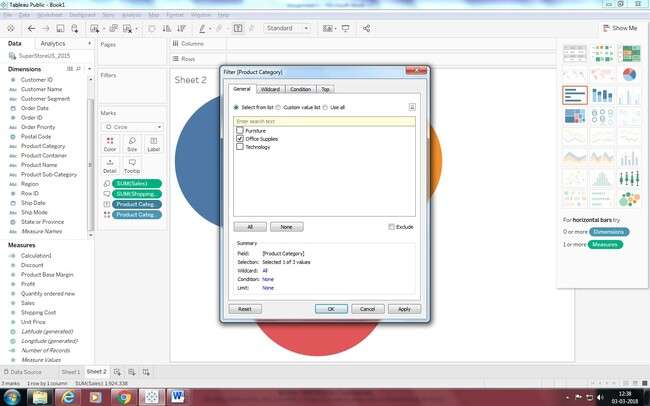
1. **Use Shipping Cost to display the size of each bubble**



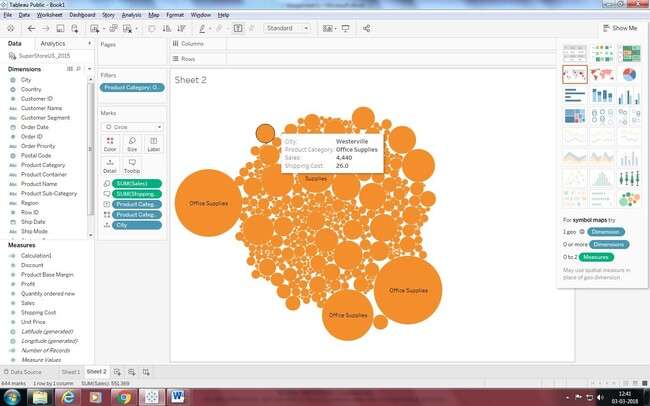


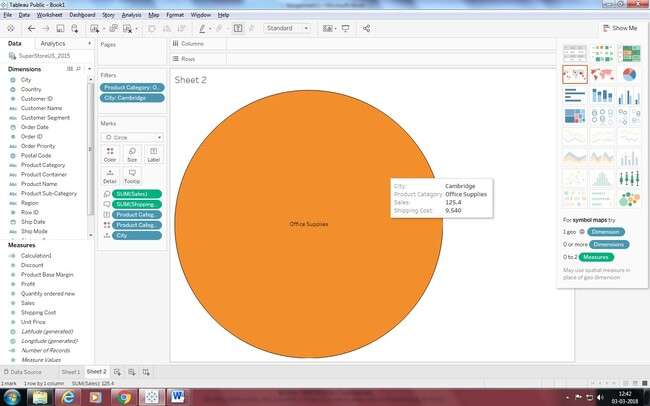
Practical 7: Use the same visualization created for Question #6. What was the shipping cost of Office Supplies for Xerox 1905 in the Home Customer Segment in Cambridge?

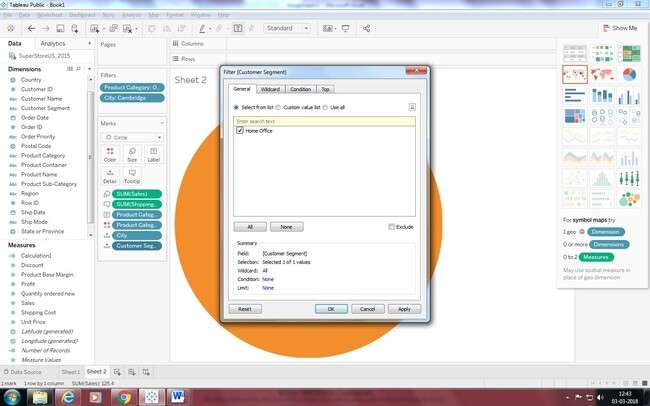
**Ans:** Apply filters as per requirement

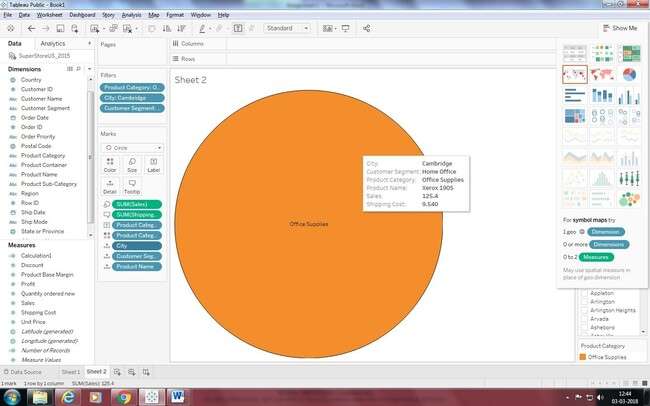


Apply city Filter







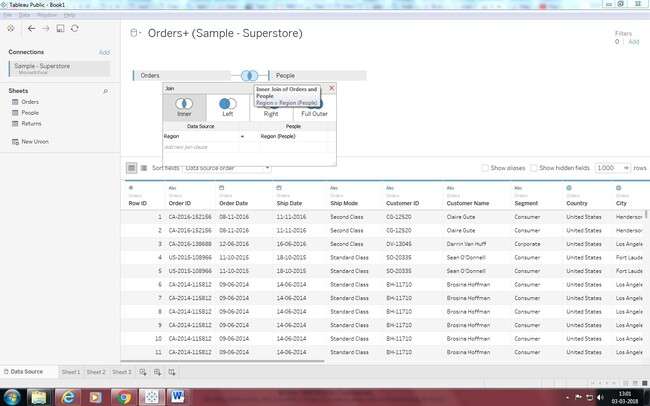


# Assignment 2: Preparing Maps

**Data Set for this Lab Sample- Superstore**

**Practical 1: Prepare a Geographic map to show sales in each state. Ans:**

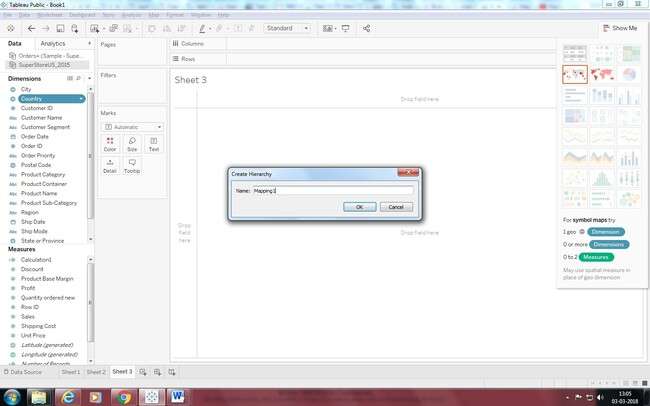
1. Connect to dataset
2. Join sheets

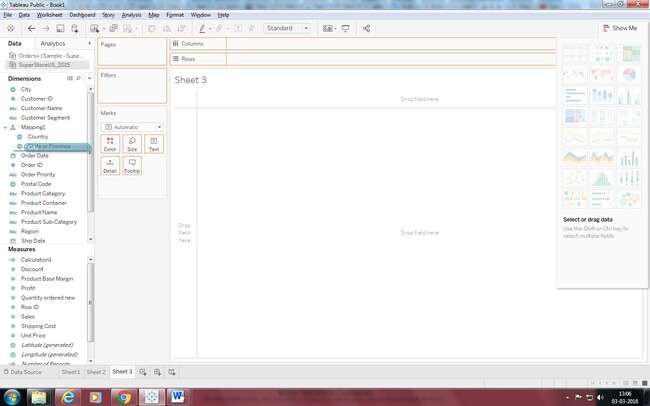


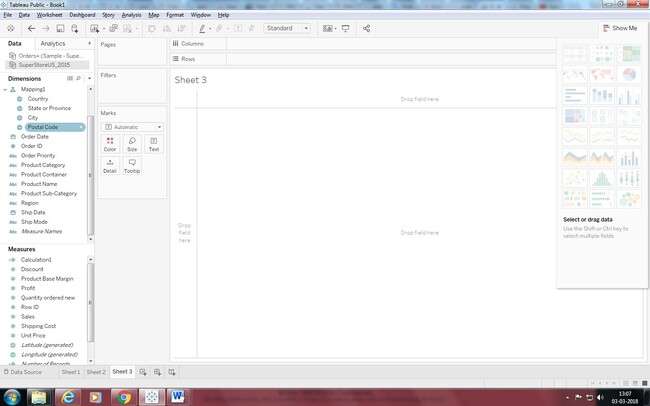
1. Create a Geographic Hierarchy
   1. In the Data pane, right-click the geographic field, **Country**, and then select **Hierarchy** > **Create Hierarchy**.
   2. In the Create Hierarchy dialog box that opens, give the hierarchy a name, such as Mapping Items, and then click **OK**.

At the bottom of the Dimensions section, the Mapping Items hierarchy is created with the Country field.

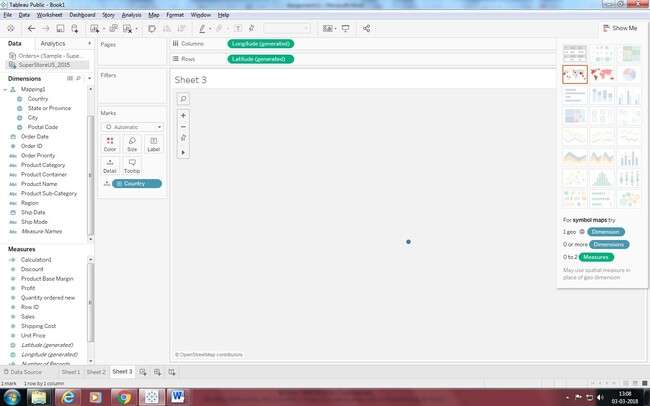
* 1. In the Data pane, drag the State field to the hierarchy and place it below the Country field.
  2. Repeat step 3 for the City and Postal Code fields.



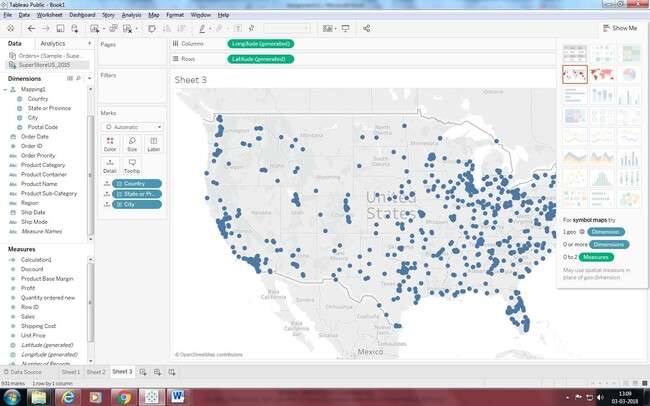




1. Build a basic map
2. In the Data pane, double-click Country.

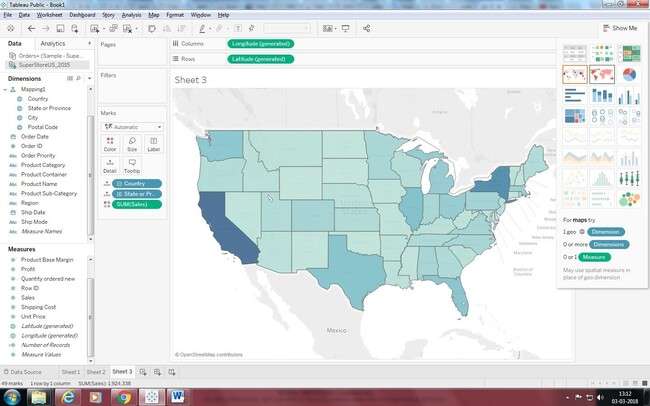


1. On the Marks card, click the + icon on the Country field.



1. Add visual detail Add color

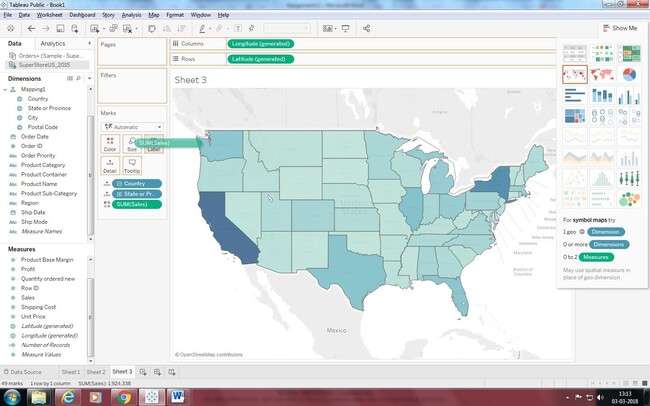
From Measures, drag Sales to Color on the Marks card.

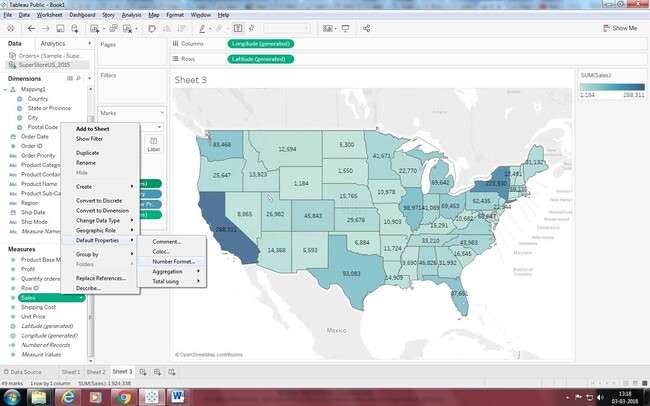
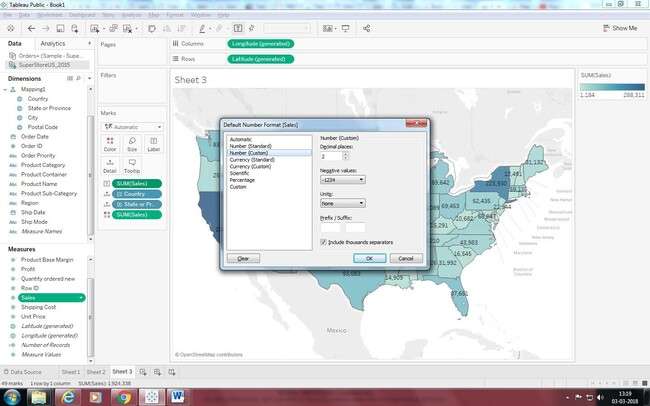


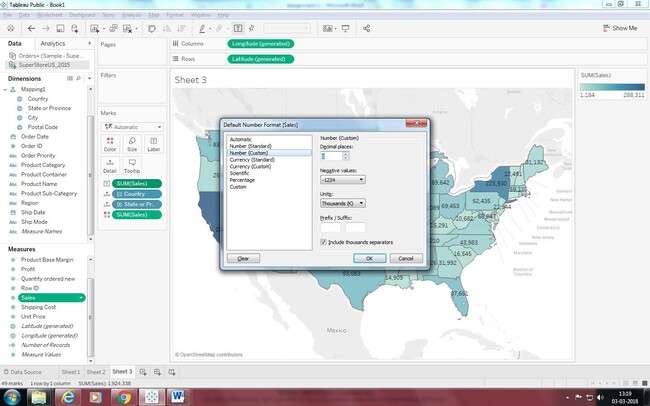
Add labels

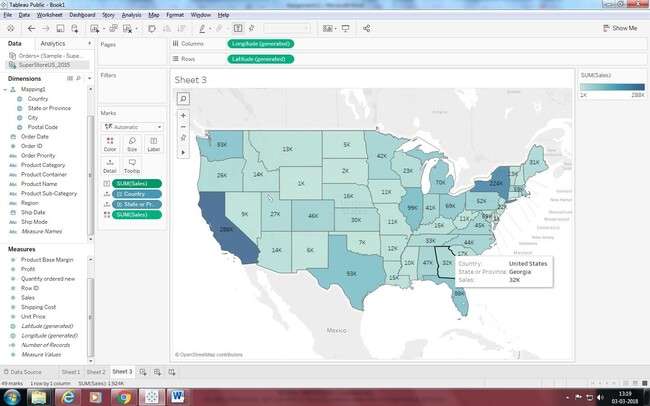
* 1. From Measures, drag Sales to Label on the Marks card.

Each state is labeled with sum of sales. The numbers need a little bit of formatting, however.

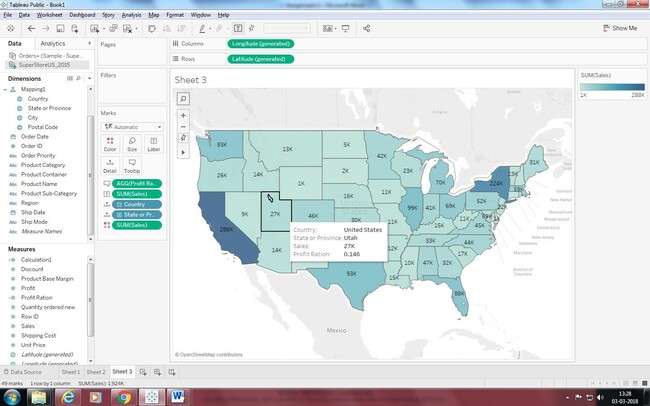
* 1. In the Data pane, right-click Sales and select Default Properties > Number Format.
  2. In the Default Number Format dialog box that opens, select Number (Custom), and then do the following:
     + For Decimal Places, enter 0.
     + For Units, select Thousands (K).
     + Click OK.



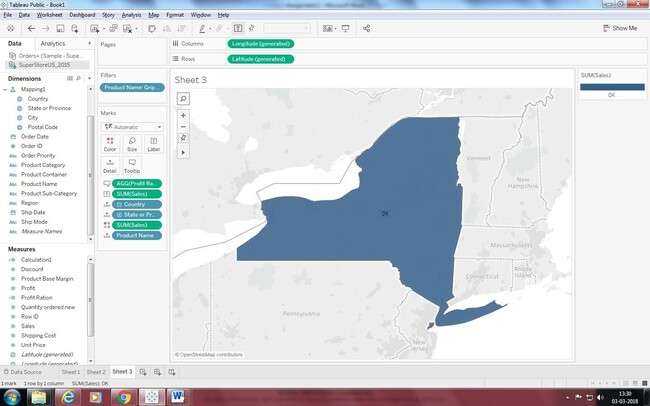




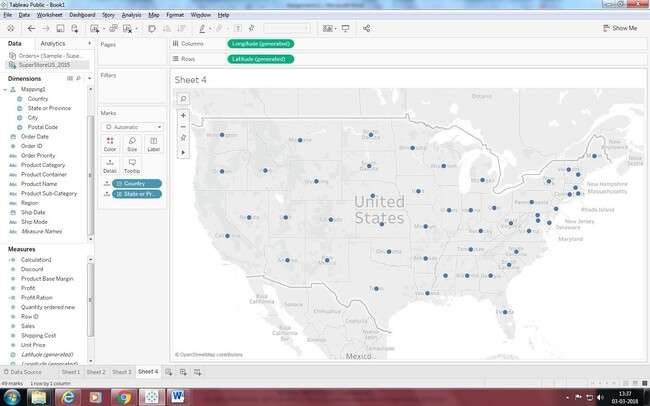
# Practical 2: Show Profit Ratio of each state as tooltip on map

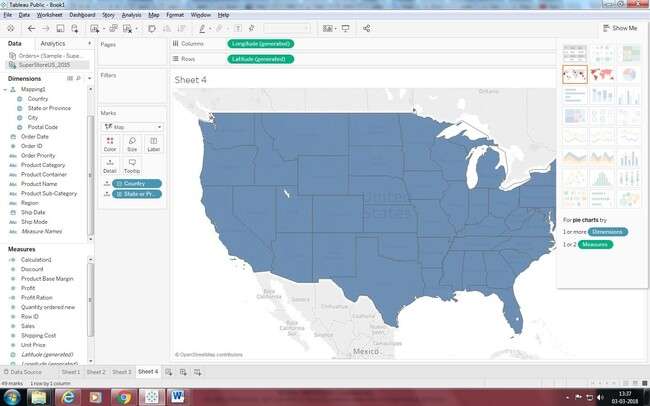


**Practical 3: Show Profit ratio for Grip Envelop products**

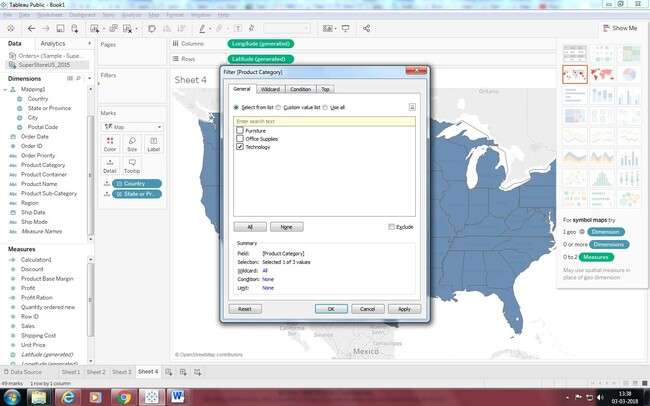


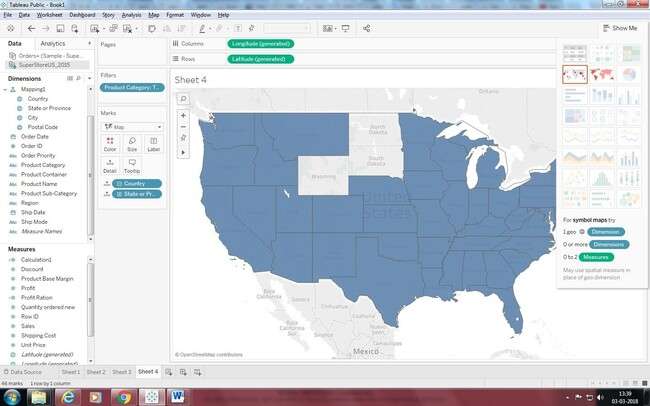
# Practical 4: In the technology product category which unprofitable state is surrounded byonly profitable states.



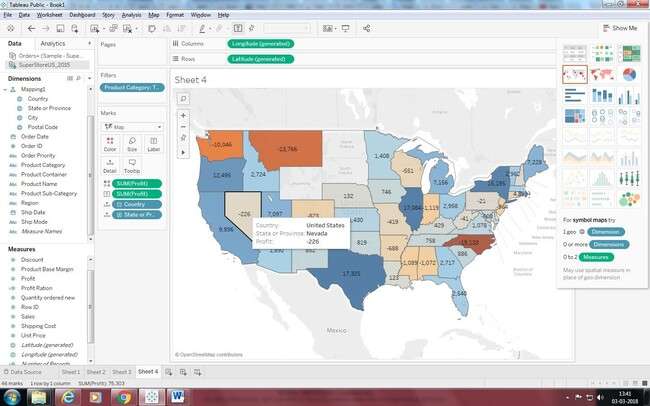


Drag the product category on the filter shelf and select in technology.





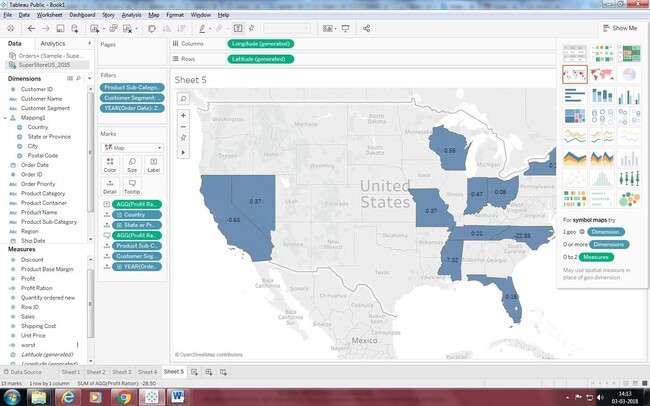
Now Drag the profit measure to color mark



Add profit Label.

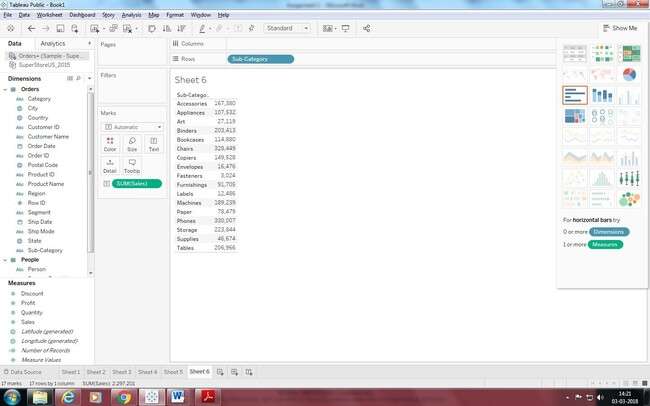
Now we can see Nevada State is surrounded by profitable states.

**Practical 5: Which state has the worst Gross Profit Ratio on Envelopes in the Corporate Customer Segment that were Shipped in 2015?**

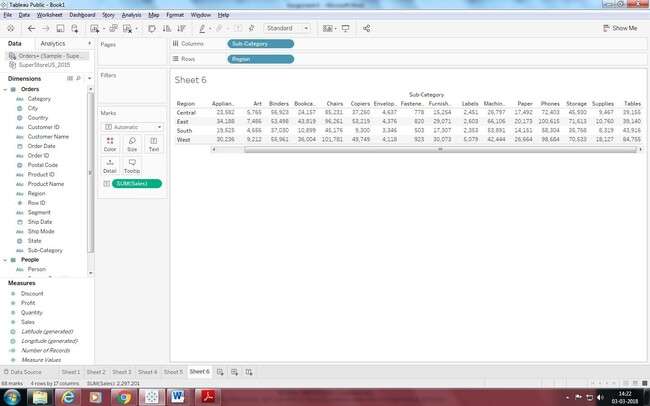


**Assignment 3: Preparing Reports Data Set: Super store**

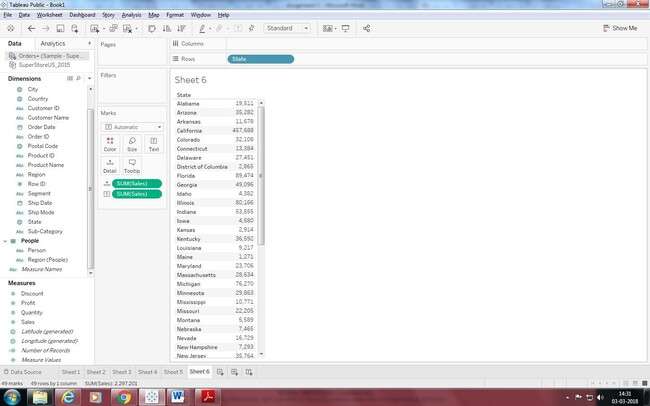
**Practical 1:- Prapare a report showing product category wise sales**



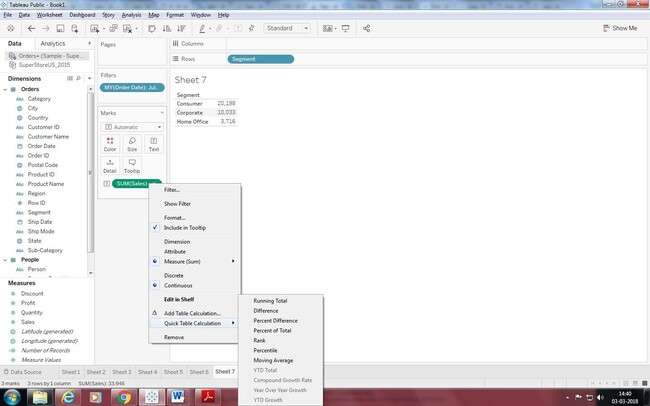
**Practical 2:- Report showing regionwise productwise sales**

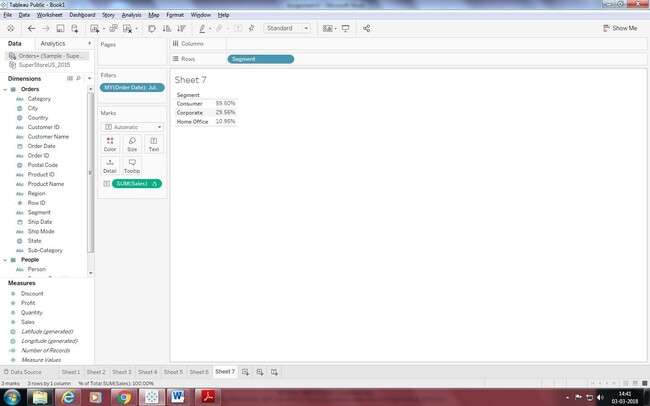


**Practical 3:- Report showing statewise sales**



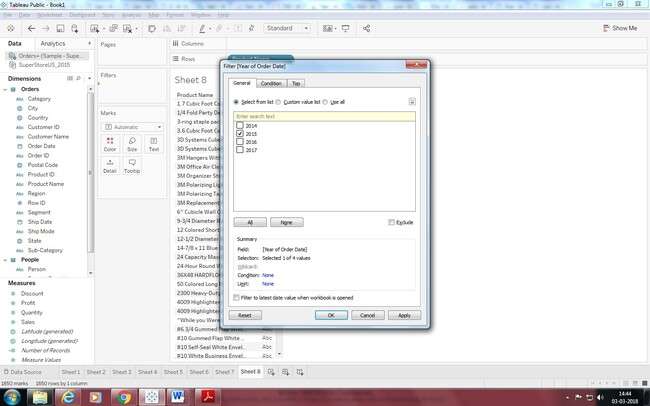
**Practical 4:- What is the percent of total Sales for the ‘Home Office’ Customer Segment in July of 2014?**

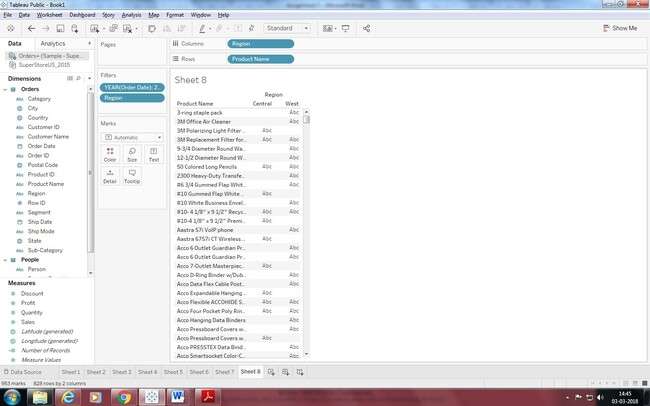




Practical 5:- Find the top 10 Product Names by Sales within each region. Which product is ranked #2 in both the Central & West regions in 2015?

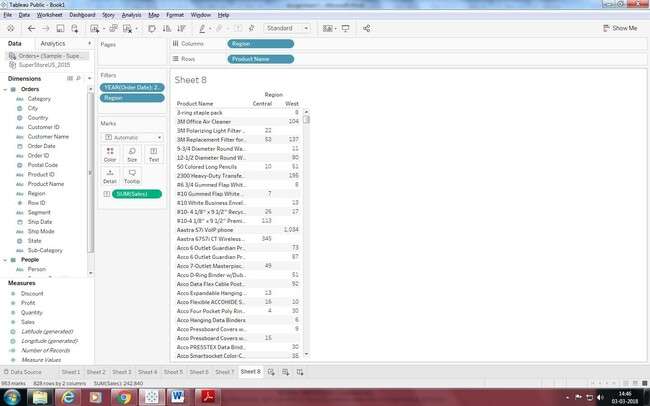
Drag “Product Name” dimension from data pane window to Row Shelf and then add an “order Date” on Filter shelf and select “Year” of Order date as 2015.After that put region on Filter shelf and select “Central” and “West” checkbox. Also, put a copy of region to the Column shelf as well.

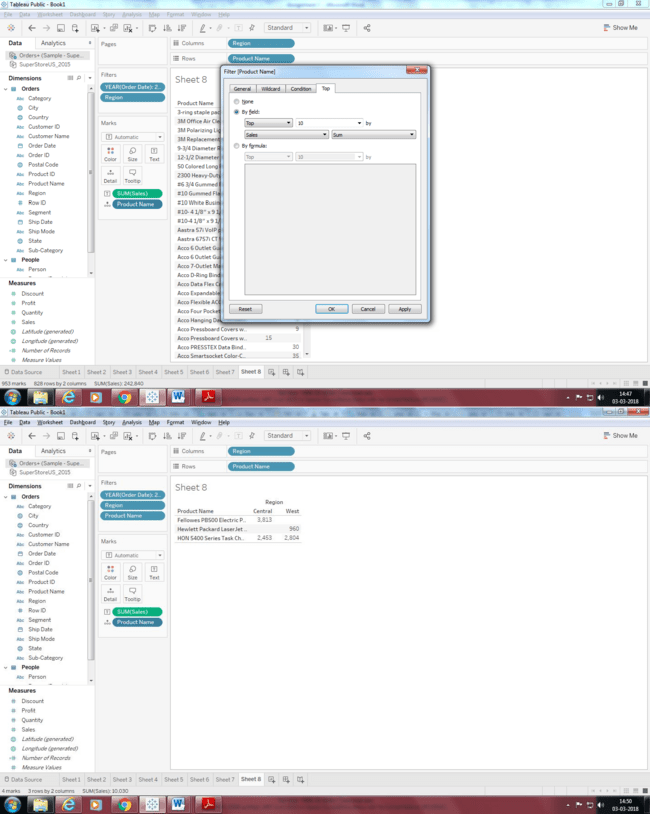




Drag a Sales measure to the text label. So for getting the Top 10 “product name” by sales, we need to add the “Product name” on Filter shelf. Once the Filter Pop up is open,

Select **“TOP” tab >By Field > Top 10 by Sum (Sales).**





Right click on the aggregated Sales measure and click on arrow sign then select **Quick Table Calculation > Rank.** As the default addressing is Table across, please change it into Table Down (**Compute using -> Table Down**).

