Practical No 1

Aim: HDFS: List of Commands (mkdir, touchz, copy from local/put, copy to local/get, movefrom local, cp, rmr, du, dus, stat).

A) ls-

Description: Shows list of files and directories.

Command: ls

Output:

```
[cloudera@quickstart ~]$ ls
cloudera-manager  Downloads                 kerberos  Public
cm_api.py         eclipse                   lib       Templates
Desktop           enterprise-deployment.json  Music     Videos
Documents         express-deployment.json   Pictures  workspace
```

B) pwd -

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

Command: pwd

Output:

```
[cloudera@quickstart ~]$ pwd
/home/cloudera
```

C) whoami-

Description: Shows the current user.

Command: whoami

Output:

```
[cloudera@quickstart ~]$ whoami
cloudera
```

D) mkdir-

Description: Create a new directory.

Command: mkdir firstdi

E) cd-

Description: This command use to change the directory.

Command: cd directory name (Ex- cd firstdir)

Output:

F) cd .. -

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

Command: cd ..

G) touch-

Description: It create a new text file.
Command: touch filename.txt (Ex-touch firstfile.txt).

H) vi-

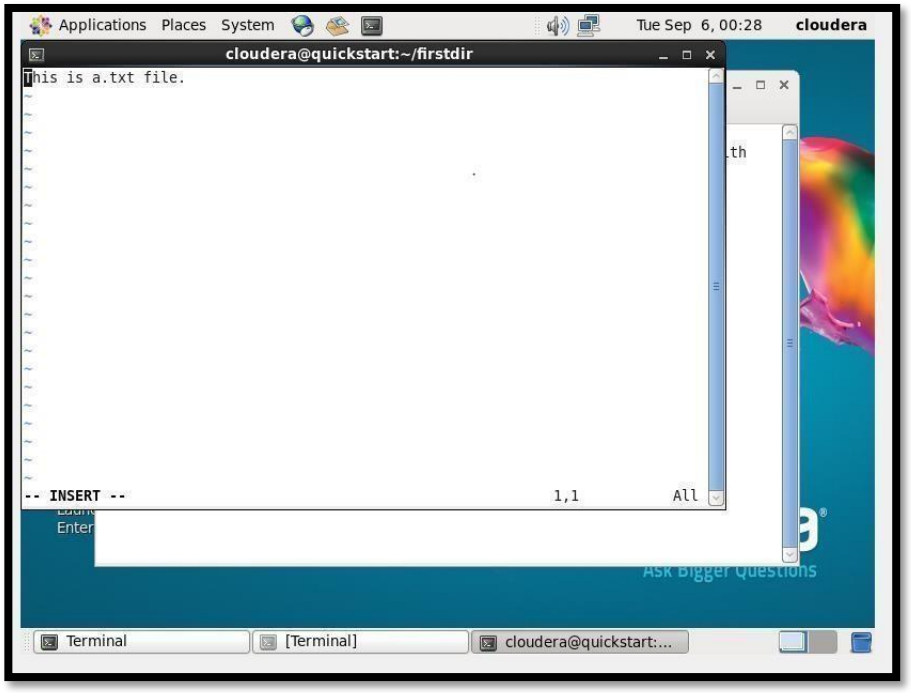
Description: It allow user to write something in text file.

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

Output:

```
[cloudera@quickstart ~]$ vi firstfile.txt
```

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.

I) cat-

Description: It return the content of file.
Command: cat filename (Ex- cat firstfile.txt)

Output:

```
[cloudera@quickstart ~]$ cat firstfile.txt
My Name Aniket.
Studying SYMCA.
Roll No 202114.
```

J) cp-

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

Command: cp source destination

Output:

```
[cloudera@quickstart ~]$ cp /home/cloudera/firstdir/a.txt /home/cloudera/secondd
ir
[cloudera@quickstart ~]$ cd seconddir
[cloudera@quickstart seconddir]$ pwd
/home/cloudera/seconddir
[cloudera@quickstart seconddir]$ ls
a.txt  b.txt
```

K) 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)

```
[cloudera@quickstart ~]$ mv /home/cloudera/firstdir/d.txt /home/cloudera/seconddir
[cloudera@quickstart ~]$ ls
cloudera-manager  Downloads                 firstdir       Music      Templates
cm_api.py         eclipse                   firstfile.txt  Pictures   Videos
Desktop           enterprise-deployment.json kerberos      Public     workspace
Documents         express-deployment.json   lib            seconddir
```

L) rm-

Description: It is use to to remove file.

Command: rm prasad.txt

```
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
[cloudera@quickstart ~]$ touch prasad.txt
[cloudera@quickstart ~]$ ls
cloudera-manager  eclipse                   Music       student_new.txt
cm_api.py         enterprise-deployment.json parcels     studnet_new.txt
data              express-deployment.json   Pictures    studnet.txt
Desktop           hiray                     prasad.txt  Templates
Documents         kerberos                  Public      Videos
Downloads         lib                       student     workspace
[cloudera@quickstart ~]$ rm prasad.txt
rm: remove regular empty file `prasad.txt'? yes
[cloudera@quickstart ~]$ ls
cloudera-manager  eclipse                   Music            studnet_new.txt
cm_api.py         enterprise-deployment.json parcels          studnet.txt
data              express-deployment.json   Pictures         Templates
Desktop           hiray                     Public           Videos
Documents         kerberos                  student          workspace
Downloads         lib                       student_new.txt
[cloudera@quickstart ~]$
```

HDFS

A) hdfs dfs -ls

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

Command: hdfs dfs -ls;

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 2 items
drwx------   - cloudera cloudera          0 2022-09-22 01:10 .staging
drwxr-xr-x   - cloudera cloudera          0 2022-09-13 23:26 Hirayhadoop
```

```
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera
Found 2 items
drwx------   - cloudera cloudera          0 2022-09-22 01:10 /user/cloudera/.
ging
drwxr-xr-x   - cloudera cloudera          0 2022-09-13 23:26 /user/cloudera/H
```

B) hdfs dfs -mkdir

Description: It is use to create new directory.

Command: hdfs dfs -mkdir deepruyam;

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -mkdir deepruyam
```

C) hdfs dfs -put

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

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

Output:

```
[cloudera@quickstart Hiray]$ hdfs dfs -put /home/cloudera/Hiray/student.txt /
r/cloudera/deepruyam
```

D) 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:

```
[cloudera@quickstart Hiray]$ hdfs dfs -get /user/cloudera/deepruyam/student.t
/home/cloudera/deep
```

E) hdfs dfs -cat

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

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

Output:

```
[cloudera@quickstart Hiray]$ hdfs dfs -cat /user/cloudera/deepruyam/student.t
Mayur|22|Mumbai
Deepak|22|Nashik
Rajendra|21|Pune
Yogesh|23|Delhi
```

F) hdfs dfs - touchz

Description: It is use creates empty file in hadoop.

Command: hdfs dfs -touchz user/cloudera/hiray_hadoop/student1.txt

Output:

```
[cloudera@quickstart Hiray]$ hdfs dfs -touchz /user/cloudera/deepruyam/empty.txt
```

G) hdfs dfs -usage ls

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

Command: hdfs dfs -usage ls

```
[cloudera@quickstart ~]$ hdfs dfs -usage ls
Usage: hadoop fs [generic options] -ls [-d] [-h] [-R] [<path> ...]
```

Output:

```
[cloudera@quickstart Hiray]$ hdfs dfs -usage ls
Usage: hadoop fs [generic options] -ls [-d] [-h] [-R] [<path> ...]
```

H) hdfs dfs - help ls

Description: It gives help of command
Command: hdfs dfs - help ls

Output:

```
[cloudera@quickstart Hiray]$ hdfs dfs -help ls
-ls [-d] [-h] [-R] [<path> ...] :
  List the contents that match the specified file pattern. If path is not
  specified, the contents of /user/<currentUser> will be listed. Directory entri
es
  are of the form:
        permissions - userId groupId sizeOfDirectory(in bytes)
  modificationDate(yyyy-MM-dd HH:mm) directoryName

  and file entries are of the form:
        permissions numberOfReplicas userId groupId sizeOfFile(in bytes)
  modificationDate(yyyy-MM-dd HH:mm) fileName


  -d  Directories are listed as plain files.

  -h  Formats the sizes of files in a human-readable fashion rather than a numbe
r
      of bytes.

  -R  Recursively list the contents of directories.
```

I) hdfs dfs -rm

Description: It will move the file in trash folder

Command: hdfs dfs -rm /user/cloudera/hiray_hadoop/student1.txt

Output:

```
[cloudera@quickstart Hiray]$ hdfs dfs -rm /user/cloudera/deepruyam/empty.txt
22/09/23 00:12:56 INFO fs.TrashPolicyDefault: Namenode trash configuration: Dele
tion interval = 1440 minutes, Emptier interval = 0 minutes.
Moved: 'hdfs://quickstart.cloudera:8020/user/cloudera/deepruyam/empty.txt' to tr
ash at: hdfs://quickstart.cloudera:8020/user/cloudera/.Trash/Current
```

J) hdfs dfs -rmr

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

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

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -rmr /user/cloudera/firsthiray
rmr: DEPRECATED: Please use 'rm -r' instead.
22/09/23 23:54:24 INFO fs.TrashPolicyDefault: Moved: 'hdfs://quickstart.cloudera
:8020/user/cloudera/firsthiray' to trash at: hdfs://quickstart.cloudera:8020/use
r/cloudera/.Trash/Current/user/cloudera/firsthiray
[cloudera@quickstart ~]$
```

K) hdfs dfs - du

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

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

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -du  /user/cloudera/firsthiray/data1.txt
50  50  /user/cloudera/firsthiray/data1.txt
[cloudera@quickstart ~]$
```

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**:

```
[cloudera@quickstart ~]$ hdfs dfs -dus  /user/cloudera/firsthiray/data1.txt
dus: DEPRECATED: Please use 'du -s' instead.
50  50  /user/cloudera/firsthiray/data1.txt
[cloudera@quickstart ~]$
```

K) hdfs dfs - stat

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

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

Output:

```
[cloudera@quickstart ~]$ hdfs dfs -stat  /user/cloudera/firsthiray/data1.txt
2022-09-24 07:01:31
[cloudera@quickstart ~]$
```

Practical No 2

Aim: Write a program in Map Reduce for Word Count operation.

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<Object, Text, Text, IntWritable>
	{
		private final static IntWritable one=new IntWritable(1);
		private Text word=new Text();

		public void map(Object key,Text Value,Context context)throws IOException,InterruptedException
		{
			StringTokenizer itr=new StringTokenizer(Value.toString());
			while(itr.hasMoreTokens())
			{
				word.set(itr.nextToken());
```

```
                        context.write(word, one);
                    }
                }
            }
            public static class IntSumReducer extends Reducer<Text,
IntWritable,Text,IntWritable>
            {
                private IntWritable result=new IntWritable();

                public void reduce(Text key,Iterable<IntWritable>values,Context
context)throws IOException,InterruptedException
                {
                    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);
    }
}
```

Input File:

```
[cloudera@quickstart ~]$ cd Hiray
[cloudera@quickstart Hiray]$ ls
second_student.txt  student.txt
[cloudera@quickstart Hiray]$ cd ..
[cloudera@quickstart ~]$ touch word_count.txt
[cloudera@quickstart ~]$ vi word_count.txt
[cloudera@quickstart ~]$ cat word_count.txt
mayur salokhe
yogesh madvalkar
mayur salokhe
mayur salokhe
suraj yadav
suraj yadav
deepak mishra
deepak mishra
yogesh madvalkar
rushi rane
rushi rane
[cloudera@quickstart ~]$ pwd
/home/cloudera
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/word_count.txt /user/cl
```

Output:

```
rmdir: `output2': No such file or directory
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/input1/word_count.txt
mayur salokhe
yogesh madvalkar
mayur salokhe
mayur salokhe
suraj yadav
suraj yadav
deepak mishra
deepak mishra
yogesh madvalkar
rushi rane
rushi rane
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WordCount /user/cloudera/input1/word_count.txt /user/cloudera/output1
22/10/13 02:05:17 INFO client.RMProxy: Connecting to ResourceManager at quickstart.cloudera/10.0.2.15:8032
22/10/13 02:05:20 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/10/13 02:05:21 INFO input.FileInputFormat: Total input paths to process : 1
22/10/13 02:05:21 INFO mapreduce.JobSubmitter: number of splits:1
22/10/13 02:05:22 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1665646671947_0001
22/10/13 02:05:24 INFO impl.YarnClientImpl: Submitted application application_1665646671947_0001
22/10/13 02:05:24 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1665646671947_0001/
22/10/13 02:05:24 INFO mapreduce.Job: Running job: job_1665646671947_0001
22/10/13 02:05:48 INFO mapreduce.Job: Job job_1665646671947_0001 running in uber mode : false
```

```
                Bytes Written=88
[cloudera@quickstart ~]$ hdfs dfs -ls /user/cloudera/output1
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2022-10-13 02:06 /user/cloudera/output1/_SUCCESS
-rw-r--r--   1 cloudera cloudera         88 2022-10-13 02:06 /user/cloudera/output1/part-r-00000
[cloudera@quickstart ~]$ hdfs dfs -cat /user/cloudera/output1/part-r-00000
deepak  2
madvalkar       2
mayur   3
mishra  2
rane    2
rushi   2
salokhe 3
suraj   2
yadav   2
yogesh  2
[cloudera@quickstart ~]$
```

Aim: Write a program in Map Reduce for Union operation.

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<LongWritable,Text,Text,IntWritable>
        {
                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<Text,IntWritable,Text,IntWritable>
    {
        private IntWritable result=new IntWritable();
        public void reduce(Text key,Iterable<IntWritable>values,Context context)throws
IOException, InterruptedException
        {
            int  sum=0;
            for(IntWritable val:values)
            {
                sum +=val.get();
            }
            result.set(sum);
            context.write(key, result);
        }
    }

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

```
[cloudera@quickstart ~]$ cat setA.txt
aa bb cc dd
[cloudera@quickstart ~]$ cat setB.txt
bb cc dd ee
```

Output:

hadoop jar Union.jar Union /user/cloudera/inputfile/SetA.txt /user/cloudera/inputfile/SetB.txt /output_union

```
[cloudera@quickstart ~]$ hadoop jar Union.jar Union /user/cloudera/inputfile/set
A.txt /user/cloudera/inputfile/setB.txt /output_union
21/10/03 09:21:34 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/10/03 09:24:01 INFO input.FileInputFormat: Total input paths to process : 2
21/10/03 09:24:02 INFO mapreduce.JobSubmitter: number of splits:2
21/10/03 09:24:04 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
33273879556_0001
21/10/03 09:24:09 INFO impl.YarnClientImpl: Submitted application application_16
33273879556_0001
21/10/03 09:24:10 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1633273879556_0001/
21/10/03 09:24:10 INFO mapreduce.Job: Running job: job_1633273879556_0001
21/10/03 09:27:14 INFO mapreduce.Job: Job job_1633273879556_0001 running in uber
 mode : false
21/10/03 09:27:14 INFO mapreduce.Job:  map 0% reduce 0%
21/10/03 09:28:12 INFO mapreduce.Job:  map 100% reduce 0%
21/10/03 09:28:33 INFO mapreduce.Job:  map 100% reduce 100%
21/10/03 09:28:34 INFO mapreduce.Job: Job job_1633273879556_0001 completed succe
ssfully
```

hadoop fs -cat /output_union/part-r-00000

```
[cloudera@quickstart ~]$ hadoop fs -cat /output_union/part-r-00000
aa      1
bb      1
cc      2
dd      2
ee      1
ff      1
```

Aim: Write a program in Map Reduce for Intersection operation.

```
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<LongWritable,Text,Text,Text>
        {
                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<LongWritable,Text,Text,Text>
{
        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<Text,Text,Text,Text>
{
        private Text valEmit=new Text("");
        public void Reduce(Text key,Iterable<Text> 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:

```
[cloudera@quickstart ~]$ cat setA.txt
aa bb cc dd
[cloudera@quickstart ~]$ cat setB.txt
bb cc dd ee
```

Output:

hadoop jar Difference1.jar Difference1 /user/cloudera/inputfile/setA.txt /user/cloudera/inputfile/setB.txt /output_diff1

```
[cloudera@quickstart ~]$ hadoop jar Difference1.jar Difference1 /user/cloudera/i
nputfile/setA.txt /user/cloudera/inputfile/setB.txt /output_diff1
21/10/03 10:01:28 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/10/03 10:01:28 INFO input.FileInputFormat: Total input paths to process : 1
21/10/03 10:01:29 INFO input.FileInputFormat: Total input paths to process : 1
21/10/03 10:01:29 INFO mapreduce.JobSubmitter: number of splits:2
21/10/03 10:01:29 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
33273879556_0003
21/10/03 10:01:30 INFO impl.YarnClientImpl: Submitted application application_16
33273879556_0003
21/10/03 10:01:30 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1633273879556_0003/
21/10/03 10:01:30 INFO mapreduce.Job: Running job: job_1633273879556_0003
21/10/03 10:01:36 INFO mapreduce.Job: Job job_1633273879556_0003 running in uber
 mode : false
21/10/03 10:01:36 INFO mapreduce.Job:  map 0% reduce 0%
21/10/03 10:02:07 INFO mapreduce.Job:  map 100% reduce 0%
21/10/03 10:02:43 INFO mapreduce.Job:  map 100% reduce 100%
21/10/03 10:02:44 INFO mapreduce.Job: Job job_1633273879556_0003 completed succe
ssfully
```

hadoop fs -cat /output_diff1/part-r-00000

```
[cloudera@quickstart ~]$ hadoop fs -cat /output_diff1/part-r-00000
aa
bb
```

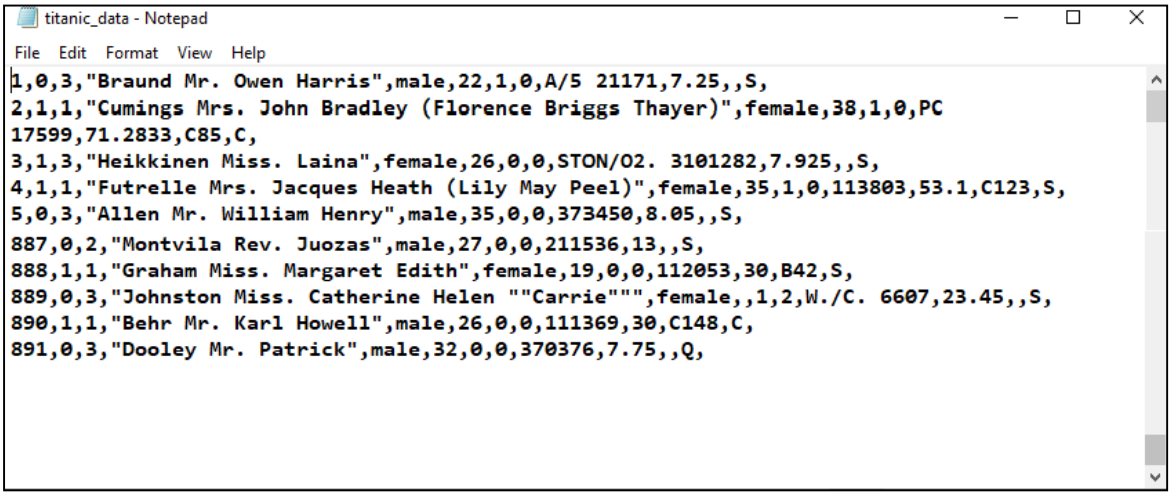
Aim: Write a program in Map Reduce for Grouping and Aggregation.

```
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<LongWritable, Text, Text, IntWritable>
        {
                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<Text, IntWritable, Text, IntWritable>
        {
            public void reduce(Text key, Iterable<IntWritable>values, 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:

hadoop fs jar AVG.jar AvgGroup /user/cloudera/inputfile/Titanic.txt /outputAvg

```
[cloudera@quickstart ~]$ hadoop jar AVG.jar AvgGroup /user/cloudera/inputfile/Ti
tanic.txt /outputAvg
21/10/07 21:00:44 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/10/07 21:00:47 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
21/10/07 21:00:54 INFO input.FileInputFormat: Total input paths to process : 1
21/10/07 21:00:55 INFO mapreduce.JobSubmitter: number of splits:1
21/10/07 21:00:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
33663593785_0001
21/10/07 21:01:18 INFO impl.YarnClientImpl: Submitted application application_16
33663593785_0001
21/10/07 21:01:18 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1633663593785_0001/
21/10/07 21:01:18 INFO mapreduce.Job: Running job: job_1633663593785_0001
21/10/07 21:02:15 INFO mapreduce.Job: Job job_1633663593785_0001 running in uber
 mode : false
21/10/07 21:02:15 INFO mapreduce.Job:  map 0% reduce 0%
21/10/07 21:02:46 INFO mapreduce.Job:  map 100% reduce 0%
21/10/07 21:03:08 INFO mapreduce.Job:  map 100% reduce 100%
21/10/07 21:03:08 INFO mapreduce.Job: Job job_1633663593785_0001 completed succe
ssfully
```

hadoop fs -cat /outputAvg/part-r-00000

```
[cloudera@quickstart ~]$ hadoop fs -ls /outputAvg
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2021-10-07 21:03 /outputAvg/_SUCCE
SS
-rw-r--r--   1 cloudera supergroup         18 2021-10-07 21:03 /outputAvg/part-r
-00000
[cloudera@quickstart ~]$ hadoop fs -cat /outputAvg/part-r-00000
female  28
male    30
```

Aim: Write a program in Map Reduce for Matrix Multiplication

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<LongWritable, Text, Text, Text> {

            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<m; i++)
                    {
                        outputKey.set(i + "," + indicesAndValue[2]);
                        outputValue.set("N," + indicesAndValue[1] + "," +
indicesAndValue[3]);
                        context.write(outputKey, outputValue);
                    }
                }
            }
        }
        public static class ReduceM extends Reducer<Text, Text, Text, Text> {

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

                String[] value;
                HashMap<Integer, Float> hashA = new HashMap<Integer, Float>();
                HashMap<Integer, Float> hashB = new HashMap<Integer, Float>();
```

```
                for(Text val : values){
                    value =val.toString().split(",");
                    if(value[0].equals("M"))
                    {
                        hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
                    }
                    else
                    {
                        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:

```
[cloudera@quickstart ~]$ cat final.txt
M,0,0,3
M,0,1,4
M,0,2,5
M,1,0,2
M,1,1,3
M,1,2,1
N,0,0,3
N,0,1,2
N,1,0,1
N,1,1,3
N,2,0,4
N,2,1,2
```

Output:

hadoop jar Matrix.jar Matrix /user/cloudera/inputfile/final.txt /output_59

```
[cloudera@quickstart ~]$ hadoop jar Matrix.jar Matrix /user/cloudera/inputfile/f
inal.txt /output_59
21/10/18 00:44:35 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
21/10/18 00:44:37 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
21/10/18 00:44:38 INFO input.FileInputFormat: Total input paths to process : 1
21/10/18 00:44:38 INFO mapreduce.JobSubmitter: number of splits:1
21/10/18 00:44:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_16
34534649142_0009
21/10/18 00:44:39 INFO impl.YarnClientImpl: Submitted application application_16
34534649142_0009
21/10/18 00:44:40 INFO mapreduce.Job: The url to track the job: http://quickstar
t.cloudera:8088/proxy/application_1634534649142_0009/
21/10/18 00:44:40 INFO mapreduce.Job: Running job: job_1634534649142_0009
21/10/18 00:45:28 INFO mapreduce.Job: Job job_1634534649142_0009 running in uber
 mode : false
21/10/18 00:45:28 INFO mapreduce.Job:  map 0% reduce 0%
21/10/18 00:46:26 INFO mapreduce.Job:  map 100% reduce 0%
21/10/18 00:46:49 INFO mapreduce.Job:  map 100% reduce 100%
21/10/18 00:46:50 INFO mapreduce.Job: Job job_1634534649142_0009 completed succe
ssfully
```

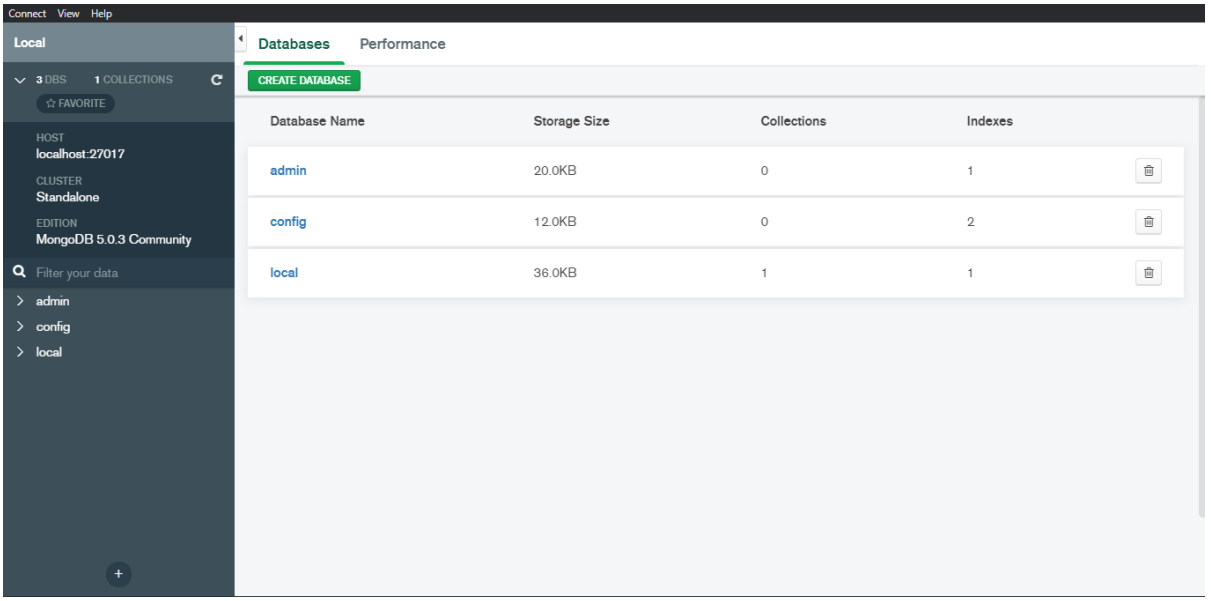
hadoop fs -cat /output_59/part-r-00000

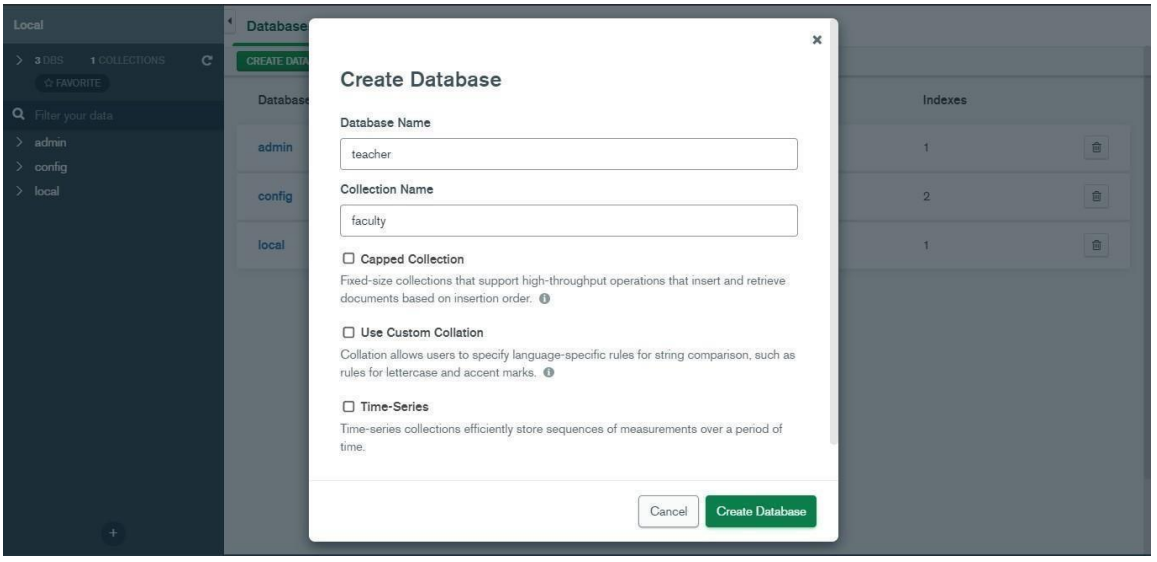
```
[cloudera@quickstart ~]$ hadoop fs -cat /output_59/part-r-00000
0,0 33.0
0,1 28.0
1,0 13.0
1,1 15.0
```

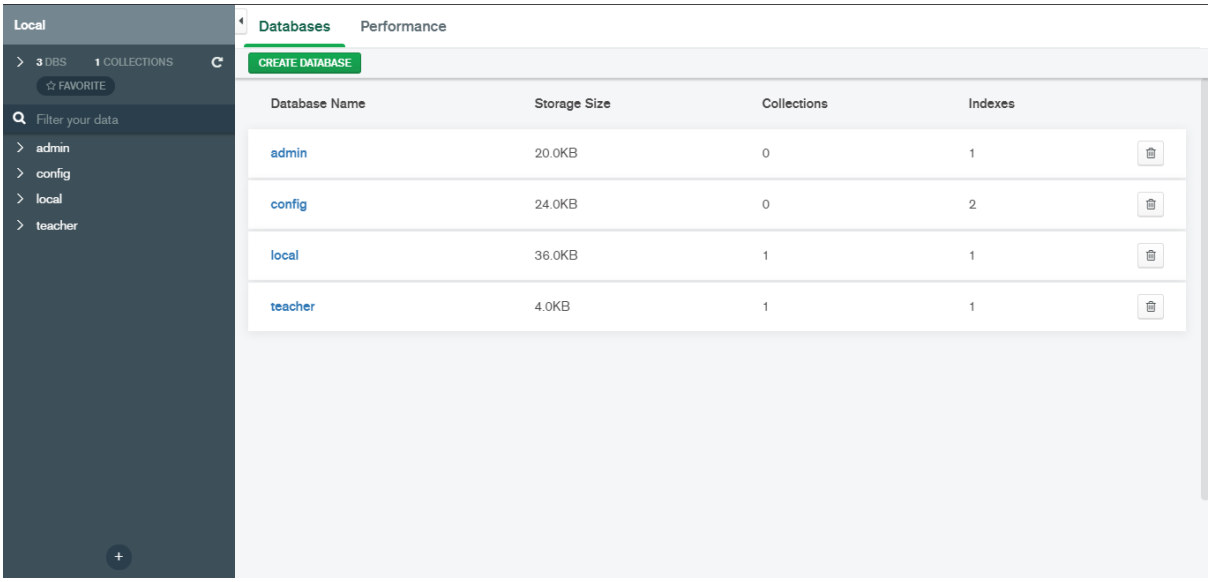
Practical No 3

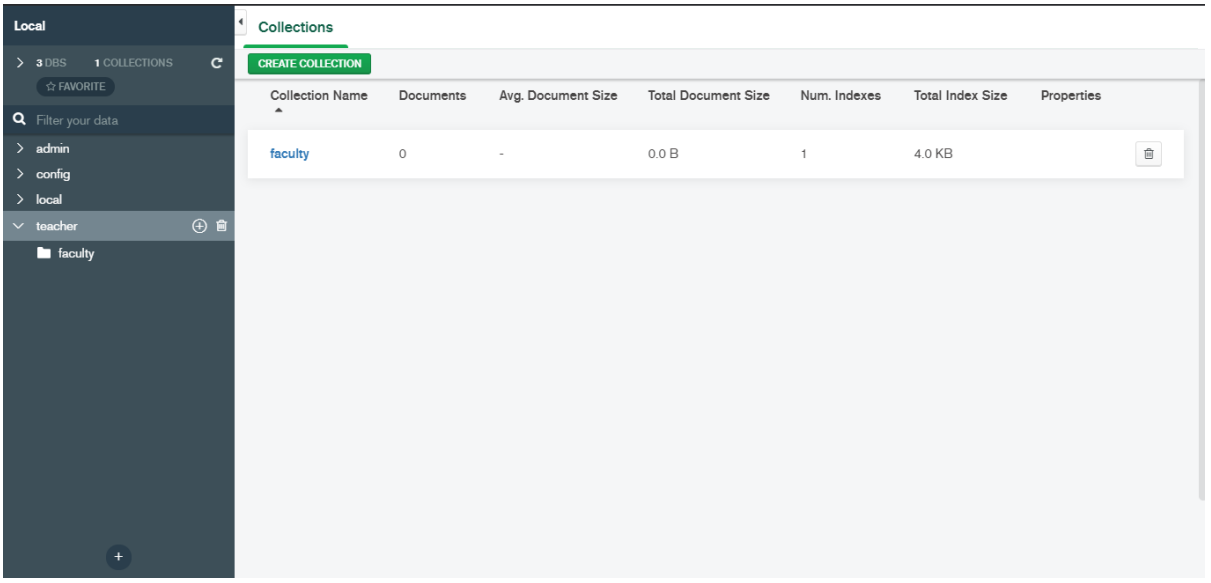
A) Installation

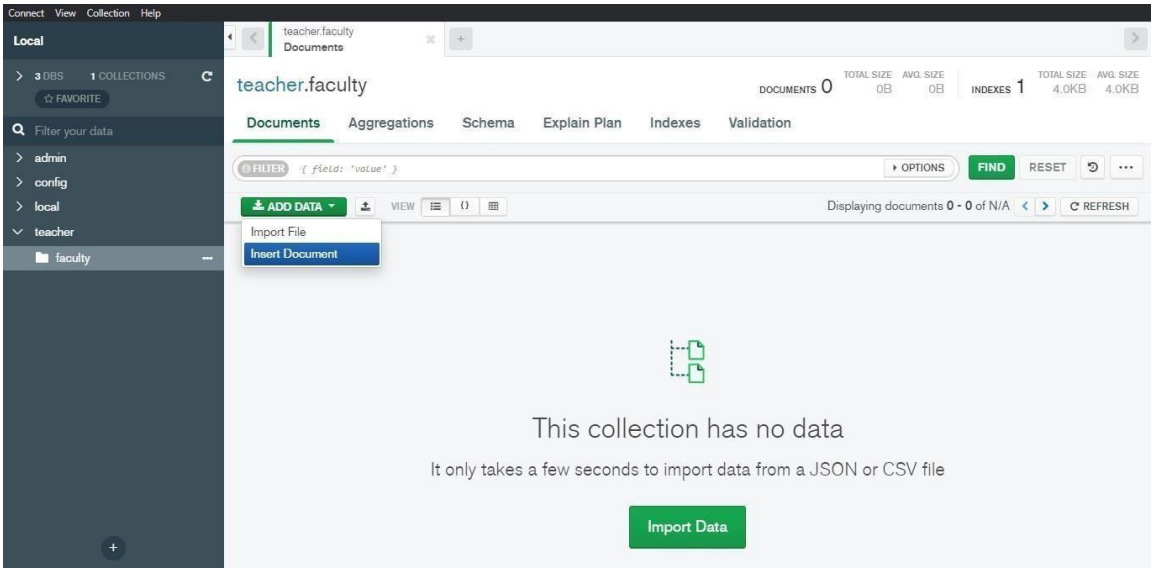
B) Sample Database Creation

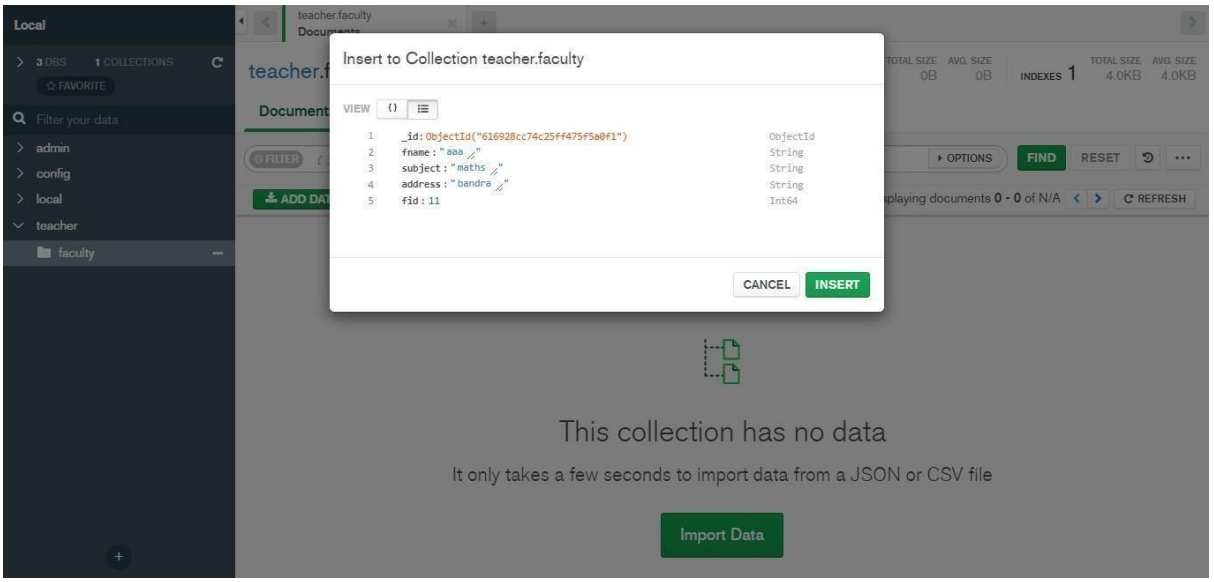


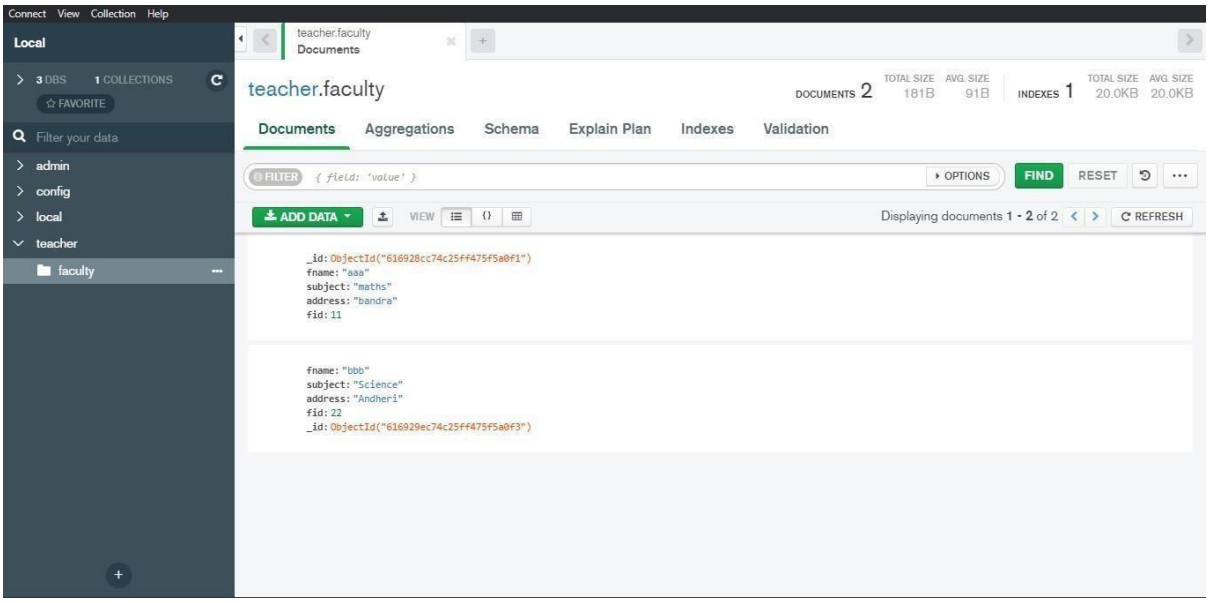



Local
3 DBS 1 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
teacher
Databases
Performance
CREATE DATABASE
Database Name
Storage Size
Collections
Indexes
admin 20.0KB 0 1
config 24.0KB 0 2
local 36.0KB 1 1
teacher 4.0KB 1 1


Local
3 DBS 1 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
teacher
faculty
Collections
CREATE COLLECTION
Collection Name
Documents
Avg. Document Size
Total Document Size
Num. Indexes
Total Index Size
Properties
faculty 0 - 0.0 B 1 4.0 KB


Connect View Collection Help
Local
3 DBS 1 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
teacher
faculty
teacher.faculty
Documents
DOCUMENTS 0
INDEXES 1
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
OPTIONS
FIND
RESET
ADD DATA
Import File
Insert Document
Displaying documents 0 - 0 of N/A
REFRESH
This collection has no data
It only takes a few seconds to import data from a JSON or CSV file
Import Data





C) Query the Sample Database using MongoDB querying commands

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:

```
show dbs
```

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
teacher  0.000GB
```

use teacher

```
> use teacher
switched to db teacher
```

show collections

```
> show collections
faculty
```

db.faculty.find()

```
> db.faculty.find()
{ "_id" : ObjectId("616928cc74c25ff475f5a0f1"), "fname" : "aaa", "subject" : "maths", "address" : "bandra", "fid" : Numb
erLong(11) }
{ "_id" : ObjectId("616929ec74c25ff475f5a0f3"), "fname" : "bbb", "subject" : "Science", "address" : "Andheri", "fid" : N
umberLong(22) }
```

db.createCollection("book")

```
> db.createCollection("book")
{ "ok" : 1 }
```

show collections

```
> show collections
book
faculty
```

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:

db.book.insert({"b_id":33, "b_language":"English", "author":"xyz", "edition":"first"})

```
> db.book.insert({"b_id":33, "b_language":"English", "author":"xyz", "edition":"first"})
WriteResult({ "nInserted" : 1 })
```

db.book.find()

```
> db.book.find()
{ "_id" : ObjectId("61692cb20ef07ac0671e83a5"), "b_id" : 33, "b_language" : "English", "author" : "xyz", "edition" : "fi
rst" }
```

db.book.insert({"b_id":11, "b_language":"Marathi", "author":"abc", "edition":"first"})

db.book.insert({"b_id":22, "b_language":"Hindi", "author":"pqr", "edition":"second"})

```
> db.book.insert({"b_id":11, "b_language":"Marathi", "author":"abc", "edition":"first"})
WriteResult({ "nInserted" : 1 })
> db.book.insert({"b_id":22, "b_language":"Hindi", "author":"pqr", "edition":"second"})
WriteResult({ "nInserted" : 1 })
```

db.book.find()

```
> db.book.find()
{ "_id" : ObjectId("61692cb20ef07ac0671e83a5"), "b_id" : 33, "b_language" : "English", "author" : "xyz", "edition" : "fi
rst" }
{ "_id" : ObjectId("61692dd50ef07ac0671e83a6"), "b_id" : 11, "b_language" : "Marathi", "author" : "abc", "edition" : "fi
rst" }
{ "_id" : ObjectId("61692e020ef07ac0671e83a7"), "b_id" : 22, "b_language" : "Hindi", "author" : "pqr", "edition" : "seco
nd" }
```

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.

```
db.employee.insert([
    {"id":45,"firstname":"Tom", "lastname":"Cruise"},
    {"id":50,"firstname":"Maria", "lastname":"Sharapova"},
    {"id":37,"firstname":"James", "lastname":"Bond"},
    {"id":65,"firstname":"Eva", "lastname":"Bond"},
    {"id":59,"firstname":"Ginger", "lastname":"Bond"}])
```

```
> db.employee.insert([
... {"id":45,"firstname":"Tom", "lastname":"Cruise"},
... {"id":50,"firstname":"Maria", "lastname":"Sharapova"},
... {"id":37,"firstname":"James", "lastname":"Bond"},
... {"id":65,"firstname":"Eva", "lastname":"Bond"},
... {"id":59,"firstname":"Ginger", "lastname":"Bond"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 5,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
db.employee.find()
```

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83a9"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
```

```
db.employee.find({"id":{$lt:50}}).pretty()
```

```
> db.employee.find({"id":{$lt:50}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83a9"),
        "id" : 45,
        "firstname" : "Tom",
        "lastname" : "Cruise"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ab"),
        "id" : 37,
        "firstname" : "James",
        "lastname" : "Bond"
}
```

```
db.employee.find({"id":{$gt:50}}).pretty()
```

```
> db.employee.find({"id":{$gt:50}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83ac"),
        "id" : 65,
        "firstname" : "Eva",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ad"),
        "id" : 59,
        "firstname" : "Ginger",
        "lastname" : "Bond"
}
```

db.employee.find({"firstname":{$in:["Tom","Eva","Ginger"]}}).pretty()

```
> db.employee.find({"firstname":{$in:["Tom","Eva","Ginger"]}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83a9"),
        "id" : 45,
        "firstname" : "Tom",
        "lastname" : "Cruise"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ac"),
        "id" : 65,
        "firstname" : "Eva",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ad"),
        "id" : 59,
        "firstname" : "Ginger",
        "lastname" : "Bond"
}
```

db.employee.find({"firstname":{$nin:["Tom","Eva","Ginger"]}}).pretty()

```
> db.employee.find({"firstname":{$nin:["Tom","Eva","Ginger"]}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83aa"),
        "id" : 50,
        "firstname" : "Maria",
        "lastname" : "Sharapova"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ab"),
        "id" : 37,
        "firstname" : "James",
        "lastname" : "Bond"
}
```

db.employee.find({"id":{$ne:50}}).pretty()

```
> db.employee.find({"id":{$ne:50}}).pretty()
{
        "_id" : ObjectId("616935d50ef07ac0671e83a9"),
        "id" : 45,
        "firstname" : "Tom",
        "lastname" : "Cruise"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ab"),
        "id" : 37,
        "firstname" : "James",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ac"),
        "id" : 65,
        "firstname" : "Eva",
        "lastname" : "Bond"
}
{
        "_id" : ObjectId("616935d50ef07ac0671e83ad"),
        "id" : 59,
        "firstname" : "Ginger",
        "lastname" : "Bond"
}
```

db.employee.find()

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83a9"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
```

```
db.employee.update({"firstname": "Tom"},{$set:{"firstname": "Jerry"}})
```

```
> db.employee.update({"firstname": "Tom"},{$set:{"firstname": "Jerry"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
db.employee.find()
```

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83a9"), "id" : 45, "firstname" : "Jerry", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
```

```
db.employee.insert([

        {"id": 45, "firstname": "Tom","lastname": "Cruise"},

        {"id": 45,"firstname": "Tom","lastname": "Cruise"}

])
```

```
> db.employee.insert([
... {
... "id": 45,
... "firstname": "Tom",
... "lastname": "Cruise"
... },
... {
... "id": 45,
... "firstname": "Tom",
... "lastname": "Cruise"
... }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
db.employee.find()
```

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Tom", "lastname" : "Cruise" }
```

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)

```
show dbs
```

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
student  0.000GB
teacher  0.000GB
```

use student

```
> use student
switched to db student
```

db.dropDatabase()

```
> db.dropDatabase()
{ "ok" : 1 }
```

show dbs

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
teacher  0.000GB
```

```
db.employee.insert([
        {"id":78,"firstname": "Emma","lastname": "Cruise"},
        {"id": 79,"firstname": "Emma","lastname": "Cruise"}
])
```

```
> db.employee.insert([
... {"id":78,"firstname": "Emma","lastname": "Cruise"},
... {"id": 79,"firstname": "Emma","lastname": "Cruise"}
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

db.employee.find()

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b4"), "id" : 78, "firstname" : "Emma", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
```

db.employee.remove({"id":78})

```
> db.employee.remove({"id":78})
WriteResult({ "nRemoved" : 1 })
```

db.employee.find()

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
```

e) Indexing

db.book.findOne({"edition":"first"})

```
> db.book.findOne({"edition":"first"})
{
        "_id" : ObjectId("61692cb20ef07ac0671e83a5"),
        "b_id" : 33,
        "b_language" : "English",
        "author" : "xyz",
        "edition" : "first"
}
```

show collections

```
> show collections
book
faculty
```

db.library.insert({"library_id":56, "lib_name":"hiray", "lib_address":"bandra"})

```
> db.library.insert({"library_id":56, "lib_name":"hiray", "lib_address":"bandra"})
WriteResult({ "nInserted" : 1 })
```

show collections

```
> show collections
book
faculty
library
```

db.employee.update({"firstname":"Tom"},{$set:{"firstname": "Scooby"}},{multi:true})

```
> db.employee.update({"firstname":"Tom"},{$set:{"firstname": "Scooby"}},{multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })
```

db.employee.find()

```
> db.employee.find()
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
```

db.employee.find().sort({"id":1})

```
> db.employee.find().sort({"id":1})
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
```

db.employee.find().sort({"id":-1})

```
> db.employee.find().sort({"id":-1})
{ "_id" : ObjectId("61693dc80ef07ac0671e83b5"), "id" : 79, "firstname" : "Emma", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ac"), "id" : 65, "firstname" : "Eva", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ad"), "id" : 59, "firstname" : "Ginger", "lastname" : "Bond" }
{ "_id" : ObjectId("616935d50ef07ac0671e83aa"), "id" : 50, "firstname" : "Maria", "lastname" : "Sharapova" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b2"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("61693c060ef07ac0671e83b3"), "id" : 45, "firstname" : "Scooby", "lastname" : "Cruise" }
{ "_id" : ObjectId("616935d50ef07ac0671e83ab"), "id" : 37, "firstname" : "James", "lastname" : "Bond" }
```

db.employee.drop()

```
> db.employee.drop()
true
```

```
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
}])
```

```
> db.employee.find()
{ "_id" : ObjectId("61705a49309ea63c54635fe0"), "userId" : "rirani", "jobTitleName" : "Developer", "firstName" : "Romin"
, "lastName" : "Irani", "preferredFullName" : "Romin Irani", "employeeCode" : "E1", "region" : "CA", "phoneNumber" : "40
8-1234567", "emailAddress" : "romin.k.irani@gmail.com", "age" : 43 }
{ "_id" : ObjectId("61705a49309ea63c54635fe1"), "userId" : "nirani", "jobTitleName" : "Developer", "firstName" : "Neil",
 "lastName" : "Irani", "preferredFullName" : "Neil Irani", "employeeCode" : "E2", "region" : "CA", "phoneNumber" : "408-
1111111", "emailAddress" : "neilrirani@gmail.com", "age" : 36 }
{ "_id" : ObjectId("61705a49309ea63c54635fe2"), "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 }
{ "_id" : ObjectId("61705d73309ea63c54635fe3"), "userId" : "rirani", "jobTitleName" : "Developer", "firstName" : "Romin"
, "lastName" : "Irani", "preferredFullName" : "Romin Irani", "employeeCode" : "E1", "region" : "CB", "phoneNumber" : "40
8-1234567", "emailAddress" : "romin.k.irani@gmail.com", "age" : 45 }
{ "_id" : ObjectId("61705d73309ea63c54635fe4"), "userId" : "nirani", "jobTitleName" : "Developer", "firstName" : "Neil",
 "lastName" : "Irani", "preferredFullName" : "Neil Irani", "employeeCode" : "E2", "region" : "CB", "phoneNumber" : "408-
1111111", "emailAddress" : "neilrirani@gmail.com", "age" : 32 }
{ "_id" : ObjectId("61705d73309ea63c54635fe5"), "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 }
```

```
db.employee.aggregate([{$group{_id:"$region",age_s:{$sum:"$age"}}}])
```

```
> db.employee.aggregate([{$group:{_id:"$region",age_s:{$sum:"$age"}}}])
{ "_id" : "CB", "age_s" : 113 }
{ "_id" : "CA", "age_s" : 113 }
```

```
db.employee.find({"userId":"rirani"}).limit(2)
```

```
> db.employee.find({"userId":"rirani"}).limit(2).pretty()
{
        "_id" : ObjectId("61705a49309ea63c54635fe0"),
        "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" : 43
}
{
        "_id" : ObjectId("61705d73309ea63c54635fe3"),
        "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
}
```

Practical No 4

Aim 1: Introduction of HIVE Data types

- **Integer Types**

Type	Size	Range
TINYINT	1-byte signed integer	-128 to 127
SMALLINT	2-byte signed integer	32,768 to 32,767
INT	4-byte signed integer	2,147,483,648 to 2,147,483,647
BIGINT	8-byte signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

- Decimal Type

Type	Size	Range
FLOAT	4-byte	Single precision floating point number
DOUBLE	8-byte	Double precision floating point number

- **Complex Type**

Type	Size	Range
Struct	It is similar to C struct or an object where fields are accessed using the "dot" notation.	struct('James','Roy')
Map	It contains the key-value tuples where the fields are accessed using array notation.	map('first','James','last','Roy')
Array	It is a collection of similar type of values thatindexable using zero-based integers.	array('James','Roy')

- **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 char is 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.

Aim 2: Implementation of Create Database & Table in HIVE

Step 1:- Create Database Hiray

```
hive> create database hiray_mca
    > ;
OK
Time taken: 10.381 seconds
hive>
```

Step 2:- Use hiray_mca;

```
hive> use hiray_mca;
OK
Time taken: 0.59 seconds
hive>
```

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

```
Create table student(name string, age int, location
string)Row format delimited
Fields terminated by '|'
Lines terminated by '\n'
Stored as textfile;
```

```
hive> create table student(name string,age int,location string)
    > Row format delimited
    > Fields terminated by '|'
    > Lines terminated by '\n'
    > Stored as textfile;
OK
Time taken: 0.784 seconds
```

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;

```
hive> Load data local inpath '/home/cloudera/hiray/student.txt' overwrite into table student
    > ;
Loading data to table hiray_mca.student
Table hiray_mca.student stats: [numFiles=1, numRows=0, totalSize=83, rawDataSize=0]
OK
Time taken: 6.824 seconds
```

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

Select * from student;

```
hive> select * from student
    > ;
OK
sheetal 20      Mumbai
aniket  24      Airoli
prasad  20      Thane
chadan  26      Navi Mumbai
Time taken: 0.152 seconds, Fetched: 4 row(s)
```

TO 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.

```
Applications  Places  System                    Tue Sep 20, 1:42 AM  cloudera
cloudera@quickstart:~
File  Edit  View  Search  Terminal  Help
ad statement
hive> load data inpath '/user/cloudera/student.txt' into table student;
```

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

```
hive> select * from student
    > ;
OK
sheetal 20      Mumbai
aniket  24      Airoli
prasad  20      Thane
chadan  26      Navi Mumbai
Time taken: 0.152 seconds, Fetched: 4 row(s)
```

Step 4- Exit from hive by issuing below command

```
hive> Exit;
```

Step-5 Check the existence of student.txt file in hdfs by issuing below command

hdfs dfs –ls

```
[cloudera@quickstart Hiray]$ hdfs dfs -ls
Found 3 items
drwx------   - cloudera cloudera          0 2022-09-20 00:30 .Trash
drwx------   - cloudera cloudera          0 2022-09-20 02:10 .staging
drwxr-xr-x   - cloudera cloudera          0 2022-09-20 01:41 firsthiray
[cloudera@quickstart Hiray]$
```

Student.txt not existing.

Aim 3: 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;

```
hive> create table student_new(name string,age int,location string)
    > row format delimited
    > fields terminated by '|'
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.154 seconds
```

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;

```
hive> Load data local inpath '/home/cloudera/hiray/student.txt' overwrite into table student_new;
Loading data to table hiray_mca.student_new
Table hiray_mca.student_new stats: [numFiles=1, numRows=0, totalSize=83, rawDataSize=0]
OK
Time taken: 0.63 seconds
hive>
```

3) Select * from student_new;

```
hive> select * from student_new;
OK
Prasad  21      Thane
Aniket  24      Airoli
Chandan 21      Panvel
Pawan   25      Vashi
Time taken: 0.194 seconds, Fetched: 4 row(s)
```

4) **Create table student_partition(name string , age int) partitioned by (location string);**

```
hive> create table student_partition(name string,age int)partitioned by(location string);
OK
Time taken: 0.402 seconds
hive>
```

5) **Show table**

```
hive> show tables;
OK
customer
new_emp
order
student
student1
student_new
student_partition
updated_emp
Time taken: 0.089 seconds, Fetched: 8 row(s)
hive>
```

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;

```
hive> SET hive.exec.dynamic.partition=true;
hive> SET hive.exec.dynamic.partition.mode=non-strict;
hive> SET hive.enforce.bucketing=true;
hive>
```

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

insert overwrite table student_partition partition

```
Time taken for adding to write entity : 4
Partition hiray_mca.student_partition{location=mumbai} stats: [numFiles=1, numRows=2, totalSize=22, rawDataSize=20]
Partition hiray_mca.student_partition{location=nagpur} stats: [numFiles=1, numRows=1, totalSize=10, rawDataSize=9]
Partition hiray_mca.student_partition{location=nashik} stats: [numFiles=1, numRows=1, totalSize=9, rawDataSize=8]
Partition hiray_mca.student_partition{location=pune} stats: [numFiles=1, numRows=1, totalSize=9, rawDataSize=8]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.31 sec   HDFS Read: 3800 HDFS Write: 333 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 310 msec
OK
Time taken: 30.839 seconds
hive>
```

(location)select * from student_new;

8) Write below command at hive prompt

Describe formatted student_partition;

```
hive> Describe formatted student_partition;
OK
# col_name              data_type               comment

name                    string
age                     int

# Partition Information
# col_name              data_type               comment

location                string

# Detailed Table Information
Database:               hiray_mca
Owner:                  cloudera
CreateTime:             Thu Sep 22 01:22:11 PDT 2022
LastAccessTime:         UNKNOWN
Protect Mode:           None
Retention:              0
Location:               hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mca.db/student_partition
Table Type:             MANAGED_TABLE
Table Parameters:
        transient_lastDdlTime   1663834931

# Storage Information
SerDe Library:          org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
InputFormat:            org.apache.hadoop.mapred.TextInputFormat
OutputFormat:           org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:             No
Num Buckets:            -1
Bucket Columns:         []
Sort Columns:           []
Storage Desc Params:
        serialization.format    1
Time taken: 0.116 seconds, Fetched: 32 row(s)
hive>
```

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

hdfs dfs –ls <write the copied location path here>;

It will display the partitioned values.

```
[cloudera@quickstart ~]$ hdfs dfs -ls hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mca.db/student_partit
ion;
Found 4 items
drwxrwxrwx   - cloudera hive          0 2022-09-22 01:27 hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mc
a.db/student_partition/location=mumbai
drwxrwxrwx   - cloudera hive          0 2022-09-22 01:27 hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mc
a.db/student_partition/location=nagpur
drwxrwxrwx   - cloudera hive          0 2022-09-22 01:27 hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mc
a.db/student_partition/location=nashik
drwxrwxrwx   - cloudera hive          0 2022-09-22 01:27 hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mc
a.db/student_partition/location=pune
[cloudera@quickstart ~]$
```

AIM 4 : Implementation of HIVE Built –in Operators

- Relational Operators
- Arithmetic Operators

Relational Operators

- Creating database

```
hive> create database employee_details;
OK
Time taken: 2.608 seconds
```

- create a table

```
hive> create table emplyoee(Id int,Name string, Salary int, Designation strin
g, Dept string);
OK
Time taken: 0.975 seconds
```

- insert employee details In Table

```
hive> insert into employee values(1,'Prasad',45000,'Technical Manager','TP');
Query ID = cloudera_20220926233737_0bdd3b69-9975-4d98-b07e-af0c6e9a0193
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1664259686258_0001, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1664259686258_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1664259686258_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-09-26 23:38:20,207 Stage-1 map = 0%,  reduce = 0%
2022-09-26 23:38:32,273 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.78 se
c
MapReduce Total cumulative CPU time: 2 seconds 780 msec
Ended Job = job_1664259686258_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/employee/.hi
ve-staging_hive_2022-09-26_23-37-55_472_5049490377861273917-1/-ext-10000
Loading data to table default.employee
Table default.employee stats: [numFiles=1, numRows=1, totalSize=36, rawDataSize=
35]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 2.78 sec   HDFS Read: 4102 HDFS Write: 1
08 SUCCESS
```

- View Table

```
hive> select * from employee;
OK
1       Prasad  45000   Technical Manager       TP
2       Aniket  50000   ORACLE Deveopler        TP
3       Chandan 60000   Python_Devloper TP
4       Kiran   80000   Hr Admin        HR
Time taken: 0.186 seconds, Fetched: 4 row(s)
hive>
```

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

  Syntax: SELECT * FROM employee WHERE Id=3;

```
hive> SELECT * FROM employee WHERE Id=3;
OK
3       Chandan 60000   Python_Devloper TP
Time taken: 0.233 seconds, Fetched: 1 row(s)
```

- **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.

```
hive> SELECT * FROM employee WHERE Salary>=40000;
OK
1       Prasad  45000   Technical Manager       TP
2       Aniket  50000   ORACLE Deveopler        TP
3       Chandan 60000   Python_Devloper TP
4       Kiran   80000   Hr Admin        HR
Time taken: 0.159 seconds, Fetched: 4 row(s)
```

Arithmetic Operators

Examples of Arithmetic Operator in Hive

- **example to increase the salary of each employee by 50.**
  Syntax: select id, name, salary * 50 from employee;

```
hive> select id, name, salary * 50 from employee;
OK
1       Prasad  2250000
2       Aniket  2500000
3       Chandan 3000000
4       Kiran   4000000
Time taken: 0.123 seconds, Fetched: 4 row(s)
```

- **example to decrease the salary of each employee by 50**
  Syntax: select id, name, salary – 50 from employee;

```
hive> select id, name, salary - 50 from employee;
OK
1       Prasad  44950
2       Aniket  49950
3       Chandan 59950
4       Kiran   79950
Time taken: 0.107 seconds, Fetched: 4 row(s)
```

- **example to find out the 10% salary of each employee.**
  Syntax: select id, name, (salary*10)/100 from employee;

```
hive> select id, name, (salary*10)/100 from employee;
OK
1       Prasad  4500.0
2       Aniket  5000.0
3       Chandan 6000.0
4       Kiran   8000.0
Time taken: 0.098 seconds, Fetched: 4 row(s)
```

AIM 5: Implementation of HIVE Built –in Functions

1. **Example to fetch the square root of each customer salary.**
   Syntax: select cust_id,cust_name,sqrt(salary) from customer;

```
11      Rama    122.47448713915891
12      Pooja   141.4213562373095
13      Aliya   141.4213562373095
14      Neha    1581.1388300841897
15      Emma    519.6152422706632
Time taken: 1.249 seconds, Fetched: 5 row(s)
hive>
```

2. **Example to fetch the natural logarithm of num (Salary).**
   Synatx: select cust_id,cut_name,ln(salary) from customer;

```
hive> select cust_id,cust_name, ln(salary) from customer;
OK
11      Rama    9.615805480084347
12      Pooja   9.903487552536127
13      Aliya   9.903487552536127
14      Neha    14.73180128983843
15      Emma    12.506177237980511
Time taken: 0.12 seconds, Fetched: 5 row(s)
hive>
```

select sum(salary) from customer;

```
2825000
Time taken: 36.085 seconds, Fetched: 1 row(s)
hive>
```

a. **Max**
   select max(salary) from customer;

```
2500000
Time taken: 25.636 seconds, Fetched: 1 row(s)
hive>
```

b. **min**
   select min(salary) from customer;

```
15000
Time taken: 29.283 seconds, Fetched: 1 row(s)
hive>
```

c. count

select count(*) from customer;

```
OK
5
Time taken: 31.247 seconds, Fetched: 1 row(s)
hive>
```

d. avg

select avg(salary) from customer;

```
OK
565000.0
Time taken: 28.103 seconds, Fetched: 1 row(s)
hive>
```

Aim 6: Implementation of HIVE QL

- **Order by clause**

  Syntax: select * from customer order by cust_id desc;

```
OK
15      Emma    34      270000
14      Neha    31      2500000
13      Aliya   30      20000
12      Pooja   33      20000
11      Rama    32      15000
Time taken: 49.784 seconds, Fetched: 5 row(s)
hive>
```

- **Where By Clause**

  Syntax:select * from customer WHERE salary=20000;

```
hive> select * from customer WHERE salary=20000;
OK
12      Pooja   33      20000
13      Aliya   30      20000
Time taken: 0.398 seconds, Fetched: 2 row(s)
hive>
```

- **Limit Clause**

1. **Return first 3 rows**

Syntax: Select * from employee limit 2;

```
hive> select * from customer limit 2;
OK
11      Rama    32      15000
12      Pooja   33      20000
Time taken: 0.306 seconds, Fetched: 2 row(s)
hive>
```

2. **Create New table emp.**

Syntax: create table emp as select cust_id,cust_name,age fromcustomer;

```
hive> create table emp as select cust_id,cust_name,age from customer;
Query ID = cloudera_20220922004343_b4d62492-1d34-48e0-92c3-9cbfb52302f7
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1663831256993_0001, Tracking URL = http://quickstart.cloudera
:8088/proxy/application_1663831256993_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1663831256993_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2022-09-22 00:44:24,698 Stage-1 map = 0%,  reduce = 0%
2022-09-22 00:44:36,440 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.21 se
c
MapReduce Total cumulative CPU time: 1 seconds 210 msec
Ended Job = job_1663831256993_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mca.db
/.hive-staging_hive_2022-09-22_00-43-57_029_7155179538366327436-1/-ext-10001
Moving data to: hdfs://quickstart.cloudera:8020/user/hive/warehouse/hiray_mca.db
/emp
Table hiray_mca.emp stats: [numFiles=1, numRows=5, totalSize=57, rawDataSize=52]
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.21 sec   HDFS Read: 3648 HDFS Write: 1
26 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 210 msec
OK
Time taken: 43.945 seconds
```

3. **Select * from emp;**

```
hive> select * from emp;
OK
11      Rama    32
12      Pooja   33
13      Aliya   30
14      Neha    31
15      Emma    34
Time taken: 0.087 seconds, Fetched: 5 row(s)
hive>
```

4. **Rename Table**

```
hive> Alter table emp rename to new_emp;
OK
Time taken: 0.185 seconds
hive>
```

5. **Create a new table referring to existing table**

```
hive> Create table updated_emp like customer;
OK
Time taken: 0.205 seconds
hive>
```

- **Group By Clause**

```
select count(cust_id),salary  from customer GROUP BY salary ;
```

```
Total MapReduce CPU Time Spent: 2 seconds 820 msec
OK
1       15000
2       20000
1       270000
1       2500000
Time taken: 28.251 seconds, Fetched: 4 row(s)
hive>
```

- Join:

Create table

```
hive> create table customer(cust_id int,cust_name string,age int,salary int)
    > comment 'CUSTOMER DETAIL'
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.582 seconds
```

```
hive> create table order(oid int,customer_id int,amount int,payment string)
    > comment 'ORDER DETAIL'
    > row format delimited
    > fields terminated by ','
    > lines terminated by '\n'
    > stored as textfile;
OK
Time taken: 0.912 seconds
```

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’) ;

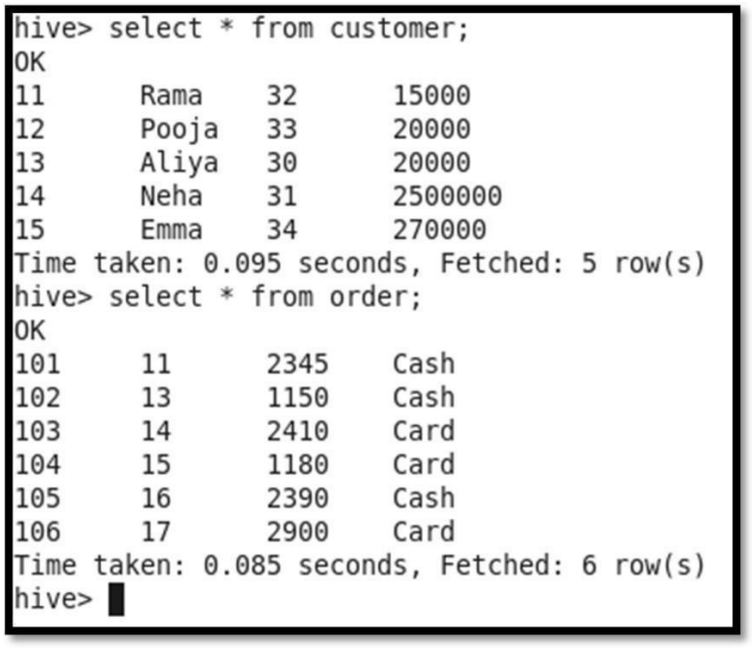
insert into order values(104, 15, 1180,’Card’) ;

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;

FULL OUTER JOIN

Syntax: 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)

FULL OUTER JOIN OUTPUT

```
OK
11      Rama    32      2345    Cash
12      Pooja   33      NULL    NULL
13      Aliya   30      1150    Cash
14      Neha    31      2410    Card
15      Emma    34      1180    Card
NULL    NULL    NULL    2390    Cash
NULL    NULL    NULL    2900    Card
Time taken: 75.269 seconds, Fetched: 7 row(s)
hive>
```

LFET OUTER JOIN

Syntax: 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);

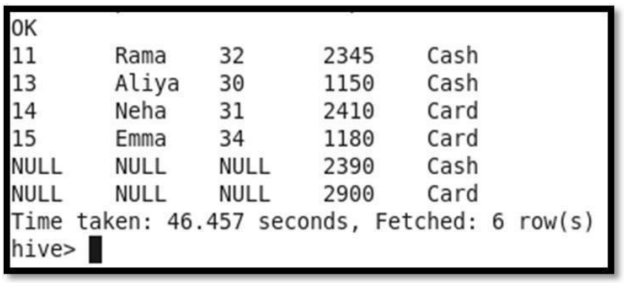
LEFT OUTER JOIN OUTPUT

```
OK
11      Rama    32      2345    Cash
12      Pooja   33      NULL    NULL
13      Aliya   30      1150    Cash
14      Neha    31      2410    Card
15      Emma    34      1180    Card
Time taken: 49.197 seconds, Fetched: 5 row(s)
hive>
```

RIGHT OUTER JOIN

Syntax: 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);

RIGHT OUTER JOIN OUTPUT



Practical No 5

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;

```
[cloudera@quickstart ~]$ pig -x local;
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell)
.
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more in
fo.
2022-09-25 23:52:10,836 [main] INFO  org.apache.pig.Main - Apache Pig version 0.
12.0-cdh5.4.2 (rexported) compiled May 19 2015, 17:03:41
2022-09-25 23:52:10,839 [main] INFO  org.apache.pig.Main - Logging error message
s to: /home/cloudera/pig_1664175130564.log
2022-09-25 23:52:11,074 [main] INFO  org.apache.pig.impl.util.Utils - Default bo
otup file /home/cloudera/.pigbootup not found
2022-09-25 23:52:12,304 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-09-25 23:52:12,305 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2022-09-25 23:52:12,310 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: file:///
2022-09-25 23:52:14,192 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2022-09-25 23:52:14,209 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-09-25 23:52:14,210 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2022-09-25 23:52:14,939 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-09-25 23:52:14,945 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
```

Step 2:- Check the availability of files

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

```
grunt> sh ls;
cloudera-manager
cm_api.py
Desktop
Documents
Downloads
eclipse
enterprise-deployment.json
express-deployment.json
kerberos
lib
Music
Pictures
Public
Templates
Videos
workspace
```

If you want to run hdfs commands in grunt then type

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

```
grunt> fs -ls;
Found 42 items
-rw-------   1 cloudera cloudera        310 2022-09-25 23:50 .ICEauthority
-rw-r--r--   1 cloudera cloudera         18 2015-06-09 03:35 .bash_logout
-rw-r--r--   1 cloudera cloudera        176 2015-06-09 03:35 .bash_profile
-rw-r--r--   1 cloudera cloudera        176 2015-06-09 03:35 .bashrc
drwxr-xr-x   - cloudera cloudera       4096 2022-09-25 23:50 .cache
drwxrwxr-x   - cloudera cloudera       4096 2022-09-25 23:50 .config
drwx------   - cloudera cloudera       4096 2022-09-25 23:50 .dbus
-rw-------   1 cloudera cloudera         16 2022-09-25 23:50 .esd_auth
drwxr-xr-x   - cloudera cloudera       4096 2022-09-25 23:50 .fontconfig
drwx------   - cloudera cloudera       4096 2022-09-25 23:50 .gconf
drwx------   - cloudera cloudera       4096 2022-09-25 23:52 .gconfd
drwxr-xr-x   - cloudera cloudera       4096 2022-09-25 23:50 .gnome2
drwx------   - cloudera cloudera       4096 2022-09-25 23:50 .gnome2_private
-rw-rw-r--   1 cloudera cloudera        152 2022-09-25 23:50 .gtk-bookmarks
drwx------   - cloudera cloudera       4096 2022-09-25 23:50 .gvfs
drwxr-xr-x   - cloudera cloudera       4096 2022-09-25 23:50 .local
drwxrwxr-x   - cloudera cloudera       4096 2015-06-09 10:01 .mozilla
drwxr-xr-x   - cloudera cloudera       4096 2022-09-25 23:50 .nautilus
-rw-rw-r--   1 cloudera cloudera         30 2022-09-25 23:53 .pig_history
drwx------   - cloudera cloudera       4096 2022-09-25 23:50 .pulse
-rw-------   1 cloudera cloudera        256 2022-09-25 23:50 .pulse-cookie
-rw-r-----   1 cloudera cloudera          5 2022-09-25 23:50 .vboxclient-clipboa
rd.pid
-rw-r-----   1 cloudera cloudera          5 2022-09-25 23:50 .vboxclient-display
.pid
-rw-r-----   1 cloudera cloudera          5 2022-09-25 23:50 .vboxclient-dragand
drop.pid
```

Step 3:- write below command at grunt shell to load local data into a relation
Command: Grunt> load_ani = ‘aniket.txt’;

```
grunt> load_ani = load 'aniket.txt';
grunt>
```

Step 4:- Write below command to display the data o relation. Here the name of relation is load_stu.
Command: Dump load_ani;

```
grunt> Dump load_ani;
```

```
l - Total input paths to process : 1
(aniket|25|airoli)
(chadan|22|panvel)
(prasad|23|thane)
(sam|25|mumbai)
grunt>
```

(Note:- Dump command is used to display records only it do not stores record.)
Step 5:- To exit from pig use quit command
Command: quit

```
grunt> quit;
[cloudera@quickstart ~]$
```

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

```
[cloudera@quickstart ~]$ pig
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
2022-09-26 00:09:14,819 [main] INFO  org.apache.pig.Main - Apache Pig version 0.12.0-cdh5.4.2 (re
xported) compiled May 19 2015, 17:03:41
2022-09-26 00:09:14,820 [main] INFO  org.apache.pig.Main - Logging error messages to: /home/cloud
era/pig_1664176154761.log
2022-09-26 00:09:14,885 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file /home/c
loudera/.pigbootup not found
2022-09-26 00:09:16,066 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.jo
b.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2022-09-26 00:09:16,066 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.defaul
t.name is deprecated. Instead, use fs.defaultFS
2022-09-26 00:09:16,066 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngi
ne - Connecting to hadoop file system at: hdfs://quickstart.cloudera:8020
2022-09-26 00:09:18,720 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.jo
b.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2022-09-26 00:09:18,721 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngi
ne - Connecting to map-reduce job tracker at: localhost:8021
2022-09-26 00:09:18,723 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.defaul
t.name is deprecated. Instead, use fs.defaultFS
2022-09-26 00:09:18,858 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.defaul
t.name is deprecated. Instead, use fs.defaultFS
2022-09-26 00:09:18,864 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.jo
b.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2022-09-26 00:09:19,009 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.defaul
t.name is deprecated. Instead, use fs.defaultFS
```

Step 2:- Check the availability of files
Command: Grunt> fs -ls

```
grunt> fs -ls
Found 1 items
-rw-r--r--   1 cloudera cloudera         64 2022-09-26 00:17 aniket.txt
```

Step 3:- write below command at grunt shell to load local data into a relation
Command: Grunt> load_ani = load '/user/cloudera/aniket.txt';

```
grunt> load_ani = load '/user/cloudera/aniket.txt';
grunt>
```

Step 4:- Write below command to display the data of relation. Here the name of relation is load_stu.
Command: Dump load_ani;

```
l - Total input paths to process : 1
(aniket|25|airoli)
(chadan|22|panvel)
(prasad|23|thane)
(sam|25|mumbai)
grunt>
```

(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.

```
[cloudera@quickstart ~]$ cat Student.txt
1,aniket,25,9892568586,airoli
2,raj,24,7898784512,thane
3,sam,26,1234567890,panvel
4,chadan,22,7894756512,airoli
5,mahesh,27,4512457812,thane
6,sham,22,4578349712,panvel
7,ram,30,6789657512,airoli
[cloudera@quickstart ~]$
```

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

Pig –x local;

```
[cloudera@quickstart ~]$ pig -x local;
log4j:WARN No appenders could be found for logger (org.apache.hadoop.util.Shell)
.
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more in
fo.
2022-09-25 23:52:10,836 [main] INFO  org.apache.pig.Main - Apache Pig version 0.
12.0-cdh5.4.2 (rexported) compiled May 19 2015, 17:03:41
2022-09-25 23:52:10,839 [main] INFO  org.apache.pig.Main - Logging error message
s to: /home/cloudera/pig_1664175130564.log
2022-09-25 23:52:11,074 [main] INFO  org.apache.pig.impl.util.Utils - Default bo
otup file /home/cloudera/.pigbootup not found
2022-09-25 23:52:12,304 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-09-25 23:52:12,305 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2022-09-25 23:52:12,310 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.HExecutionEngine - Connecting to hadoop file system at: file:///
2022-09-25 23:52:14,192 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2022-09-25 23:52:14,209 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-09-25 23:52:14,210 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
ess
2022-09-25 23:52:14,939 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
ation - fs.default.name is deprecated. Instead, use fs.defaultFS
2022-09-25 23:52:14,945 [main] INFO  org.apache.hadoop.conf.Configuration.deprec
```

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 exists in your system). Your path and path in screenshot may differ.)

```
grunt> student_data = Load '/home/cloudera/pigdata/student.txt' USING PigStorage
(',') as (id:int,name:chararray,age:int,contact:long,address:chararray);
grunt> student_agegroup = GROUP student_data by age;
grunt> Dump student_agegroup;
```

```
ne.util.MapRedUtil - Total input paths to process : 1
(22,{(6,sham,22,4578349712,panvel),(4,chadan,22,7894756512,airoli)})
(24,{(2,raj,24,7898784512,thane)})
(25,{(1,aniket,25,9892568586,airoli)})
(26,{(3,sam,26,1234567890,panvel)})
(27,{(5,mahesh,27,4512457812,thane)})
(30,{(7,ram,30,6789657512,airoli)})
(,{(,,,,)})
grunt>
```

INNER JOIN IN PIG

Create customers.txt . Refer below screen shot.

```
[cloudera@quickstart ~]$ touch customers.txt
[cloudera@quickstart ~]$ vi customers.txt
[cloudera@quickstart ~]$ cat customers.txt
1,Ramesh,32,Ahmedabad,2000.00
2,Khailan,25,Delhi,1500.00
3,Kaushik,23,kota,2000.00
4,Chaitali,25,Mumbai,6500.00
```

Create orders.txt. Refer below screen shot.

```
[cloudera@quickstart ~]$ cat order.txt
102,2009-10-08 00:00:00,3,3000
100,2009-10-08 00:00:00,3,1500
101,2009-11-20 00:00:00,3,1560
103,2008-05-20 00:00:00,4,2060
```

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

```
>> customers= LOAD '/home/cloudera/customers.txt' USING PigStorage(',') as (id:int,name:chararray,age:int,address:chararray,salary:int);
>> orders= LOAD '/home/cloudera/order.txt' USING PigStorage(',') as (oid:int,date:chararray,customer id:int,amount:int);
>> customers_orders = JOIN customers by id,orders by customer_id;
>> dump customers_orders;
```

LEFT OUTER JOIN

```
>> customers_orders_leftouter = JOIN customers by id LEFT OUTER,orders by customer_id;
>> dump customers_orders_leftouter;
```

RIGHT OUTER JOIN

```
>> customers_orders_rightouter = JOIN customers by id RIGHT,orders by customer_id;
>> dump customers_orders_rightouter;
>>
```

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

```
[cloudera@quickstart ~]$ touch Emp.txt
[cloudera@quickstart ~]$ vi Emp.txt
[cloudera@quickstart ~]$ cat Emp.txt
Manish,Singh,1
Promod,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

```
[cloudera@quickstart ~]$ hdfs dfs -put Emp.txt /user/cloudera/
[cloudera@quickstart ~]$ hdfs dfs -ls
Found 2 items
-rw-r--r--   1 cloudera cloudera         93 2022-09-29 00:48 Emp.txt
drwxr-xr-x   - cloudera cloudera          0 2022-09-26 23:49 hiraymca
[cloudera@quickstart ~]$
```

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;

```
grunt> fs -cat /user/cloudera/Emp.txt;
Manish,Singh,1
Promod,Shinde,2
Aniket,kamble,3
Prasad,Rokade,4
Chadan,Sawant,5
Manoj,Singh,6
grunt>
```

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;

```
grunt> B = Load '/user/cloudera/Emp.txt' USING PigStorage (',') as (Fname:chararray,Lname:chararr
ay,id:int);
grunt> Dump B;
```

```
l - Total input paths to process : 1
(Manish,Singh,1)
(Promod,Shinde,2)
(Aniket,kamble,3)
(Prasad,Rokade,4)
(Chadan,Sawant,5)
(Manoj,Singh,6)
grunt>
```

STEP 6:- Apply filter on row
Grunt> filterrowdata = filter B by Lname != 'Singh';

```
grunt> filterrowdata = filter B by Lname!='Singh';
grunt> Dump filterrowdata;
```

Above command will not show the record of singh
Grunt> dump filterrowdata;

```
l - Total input paths to process : 1
(Promod,Shinde,2)
(Aniket,kamble,3)
(Prasad,Rokade,4)
(Chadan,Sawant,5)
grunt>
```

STEP 7:- Apply filter on column
Grunt> filterdata = foreach B generate Fname,Lname;
Grunt> dump filterdata;

```
grunt> filterdata = foreach B generate Fname,Lname;
grunt> Dump filterdata;
```

```
l - Total input paths to process : 1
(Manish,Singh)
(Promod,Shinde)
(Aniket,kamble)
(Prasad,Rokade)
(Chadan,Sawant)
(Manoj,Singh)
grunt>
```

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_emp  = Load '/user/cloudera/Emp.txt' USING PigStorage (',') as (Fname:chararray,Lname
:chararray,id:int);
grunt> Dump load_emp;
```

```
l - Total input paths to process : 1
(Manish,Singh,1)
(Promod,Shinde,2)
(Aniket,kamble,3)
(Prasad,Rokade,4)
(Chadan,Sawant,5)
(Manoj,Singh,6)
grunt>
```

Grunt> load_student = Load '/user/cloudera/student.txt' USING PigStorage (',') as
(Fname:chararray,Lname:chararray,id:int);
Grunt> dump load_student;

```
grunt> load_student = Load '/user/cloudera/student.txt' USING PigStorage (',') as (Fname:chararra
y,Lname:chararray,id:int);
grunt> Dump load_student;
```

```
l - Total input paths to process : 1
(Raj,Sharma,101)
(Sham,Yadav,102)
(Aniket,Jadhav,103)
(Mayur,Sharma,104)
(Chadan,Yadav,105)
(Sager,Kadam,106)
```

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.

```
grunt> union_data = UNION load_emp,load_student;
grunt> Dump union_data;
```

```
l - Total input paths to process : 2
(Raj,Sharma,101)
(Sham,Yadav,102)
(Aniket,Jadhav,103)
(Mayur,Sharma,104)
(Chadan,Yadav,105)
(Sager,Kadam,106)
(,,)
(Manish,Singh,1)
(Promod,Shinde,2)
(Aniket,kamble,3)
(Prasad,Rokade,4)
(Chadan,Sawant,5)
(Manoj,Singh,6)
grunt>
```

SPLITTING DATA (using SPLIT operator)

Grunt> student_data = load '/user/cloudera/student.txt' ;

```
grunt> student_data = load '/home/cloudera/student.txt';
```

Grunt> dump student_data;

```
(prasad|23|Thane)
(aniket|24|Navi Mumbai)
(chandan|22|Panvel)
(sheetal|20|Mumbai)
(manoj|21|banglore)
(vikas|20|Mumbai)
(Meena|21|Banglore)
(ashish|20|Mumbai)
(priya|21|banglore)
()
```

Grunt> SPLIT student_data into student_split_1 if age >20, student_split_2 if (age>21 and age<25);

```
grunt> SPLIT student_data  into student_split_1  if age >20,  student_split_2 if (age>21 and age<2
5);
```

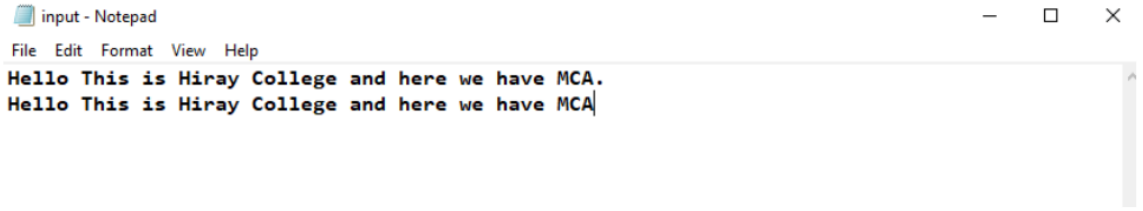
Grunt> dump student_split_1;
Grunt> dump student_split_2;
The above command will split the relation student into relation student_split_1(it contains the data of age greater than 20) and student_spilt_2 (it contains the data of age less than 21 and age greater than 22)

Practical No 6

Spark Python Mode

Word Count

Input File:




```
import findspark

findspark.init()

import pyspark

findspark.find()
```

```
'C:\\apps\\spark-3.0.3-bin-hadoop2.7'
```

```
from pyspark import SparkContext, SparkConf

sc=SparkContext("local","Counting Words")

nwords=sc.textFile("D:\MCA-SY\BDAnV\Spark-word-count\input.txt").flatMap(lambda line:
line.split(" "))

wordCo=nwords.map(lambda nwords:(nwords,1))

wordresult=wordCo.reduceByKey(lambda wordsCount, b: (wordsCount + b))

wordresult.collect()
```

```
[('Hello', 2),
 ('This', 2),
 ('is', 2),
 ('Hiray', 2),
 ('College', 2),
 ('and', 2),
 ('here', 2),
 ('we', 2),
 ('have', 2),
 ('MCA.', 1),
 ('MCA', 1)]
```

KMean

```
import findspark
findspark.init()
findspark.find()
```

```
'C:\\apps\\spark-3.0.3-bin-hadoop2.7'
```

```
import pyspark
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("K Means Cluster Model").getOrCreate()
spark
```

SparkSession - in-memory

SparkContext

Spark UI

Version	`v3.0.3`
Master	`local[*]`
AppName	`K Means Cluster Model`

```
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import VectorAssembler, VectorIndexer
from pyspark.ml.feature import StandardScaler
from pyspark.ml.evaluation import ClusteringEvaluator
data=spark.read.csv("seeds_dataset.csv", header=True, inferSchema=True)
type(data)
```

```
pyspark.sql.dataframe.DataFrame
```

```
data.head()
```

```
Row(area=15.26, perimeter=14.84, compactness=0.871, length_of_kernel=5.763, width_of_kernel=3.312, asy
mmetry_coefficient=2.221, length_of_groove=5.22)
```

```
data.printSchema()
```

```
root
 |-- area: double (nullable = true)
 |-- perimeter: double (nullable = true)
 |-- compactness: double (nullable = true)
 |-- length_of_kernel: double (nullable = true)
 |-- width_of_kernel: double (nullable = true)
 |-- asymmetry_coefficient: double (nullable = true)
 |-- length_of_groove: double (nullable = true)
```

```
data.describe().show()
```

```
+-------+------------------+------------------+--------------------+------------------+------------------+
|summary|              area|         perimeter|         compactness|   length_of_kernel|    width_of_kernel|  asymmetry_coefficient|  length_of_groove|
+-------+------------------+------------------+--------------------+------------------+------------------+
|  count|               210|               210|                 210|               210|               210|                210|               210|
|   mean|14.847523809523816|14.559285714285718|  0.8709985714285714|  5.628533333333335|  3.2586047619047624|   3.7001999999999997| 5.408071428571429|
| stddev|2.9096994306873647|1.3059587265640225|0.023629416583846364|0.44306347772645016|0.37771444490658673|    1.503558970254739|0.4914804991024053|
|    min|             10.59|             12.41|              0.8081|              4.899|              2.63|                0.765|             4.519|
|    max|             21.18|             17.25|              0.9183|              6.675|             4.033|                8.456|              6.55|
+-------+------------------+------------------+--------------------+------------------+------------------+
```

```
#Data Processing

assembler=VectorAssembler(inputCols=data.columns,outputCol='features')

final_data=assembler.transform(data)

#Scaler

scaler=StandardScaler(inputCol='features',outputCol='scaledfeatures')

final_data=scaler.fit(final_data).transform(final_data)

final_data.head()
```

```
Row(area=15.26, perimeter=14.84, compactness=0.871, length_of_kernel=5.763, width_of_kernel=3.312, asymmetry_coefficient=2.221, length_of_groove=5.22, features=DenseVector([15.26, 14.84, 0.871, 5.763, 3.312, 2.221, 5.22]), scaledfeatures=DenseVector([5.2445, 11.3633, 36.8608, 13.0072, 8.7685, 1.4772, 10.621]))
```

```
# train and test

train_data,test_data=final_data.randomSplit([0.7,0.3])

train_data.show(2)
```

```
+-----+---------+-----------+----------------+---------------+--------------------+----------------+-
-------------------+--------------------+
| area|perimeter|compactness|length_of_kernel|width_of_kernel|asymmetry_coefficient|length_of_groove|
features|      scaledfeatures|
+-----+---------+-----------+----------------+---------------+--------------------+----------------+-
-------------------+--------------------+
|10.74|    12.73|     0.8329|           5.145|          2.642|               4.702|           4.963|
[10.74,12.73,0.83...|[3.69110289768413...|
| 10.8|    12.57|      0.859|           4.981|          2.821|               4.773|           5.063|
[10.8,12.57,0.859...|[3.71172358426337...|
+-----+---------+-----------+----------------+---------------+--------------------+----------------+-
-------------------+--------------------+
only showing top 2 rows
```

```
# Build and Evaluate

# K-Means

classifier=KMeans(k=2,featuresCol='scaledfeatures')

model=classifier.fit(train_data)

# Predictions

predictions=model.transform(test_data)

#Evaluate clustring by computing Silhoutte score

score=ClusteringEvaluator().evaluate(predictions)

print(score)
```

```
0.7227070129257039
```

```
# Printing clusters centers

centers=model.clusterCenters()

print("Cluster Centers:")

for center in centers:

  print(center)
```

```
Cluster Centers:
[ 4.46157311 10.50131634 36.57279791 12.05836797  8.03273921  2.5990782
 10.33786112]
[ 6.26387723 12.32253895 37.3628276  13.88054747  9.6847573   2.35261202
 12.20660657]
```

```
predictions.show(100)
```

```
+-----+---------+-----------+---------------+--------------+--------------------+---------------+-
------------------+-------------------+----------+
| area|perimeter|compactness|length_of_kernel|width_of_kernel|asymmetry_coefficient|length_of_groove|
features|      scaledfeatures|prediction|
+-----+---------+-----------+---------------+--------------+--------------------+---------------+-
------------------+-------------------+----------+
|10.59|    12.41|     0.8648|          4.899|         2.787|               4.975|          4.794|
[10.59,12.41,0.86...|[3.63955118123602...|         0|
|10.79|    12.93|     0.8107|          5.317|         2.648|               5.462|          5.194|
[10.79,12.93,0.81...|[3.70828680316683...|         0|
|11.21|    13.13|     0.8167|          5.279|         2.687|               6.169|          5.275|
[11.21,13.13,0.81...|[3.85263160922151...|         0|
|11.23|    12.63|      0.884|          4.902|         2.879|               2.269|          4.703|
[11.23,12.63,0.88...|[3.85950517141459...|         0|
|11.34|    12.87|     0.8596|          5.053|         2.849|               3.347|          5.003|
[11.34,12.87,0.85...|[3.89730976347654...|         0|
|11.36|    13.05|     0.8382|          5.175|         2.755|               4.048|          5.263|
[11.36,13.05,0.83...|[3.90418332566962...|         0|
| 11.4|    13.08|     0.8375|          5.136|         2.763|               5.588|          5.089|
[11.4,13.08,0.837...|[3.91793045005578...|         0|
|11.48|    13.05|     0.8473|           5.18|         2.758|               5.876|          5.002|
[11.48,13.05,0.84...|[3.94542469882810...|         0|
|11.56|    13.31|     0.8198|          5.363|         2.683|               4.062|          5.182|
[11.56,13.31,0.81...|[3.97291894760042...|         0|
|11.75|    13.52|     0.8082|          5.444|         2.678|               4.378|           5.31|
[11.75,13.52,0.80...|[4.03821778843468...|         0|
```

Page Ranking

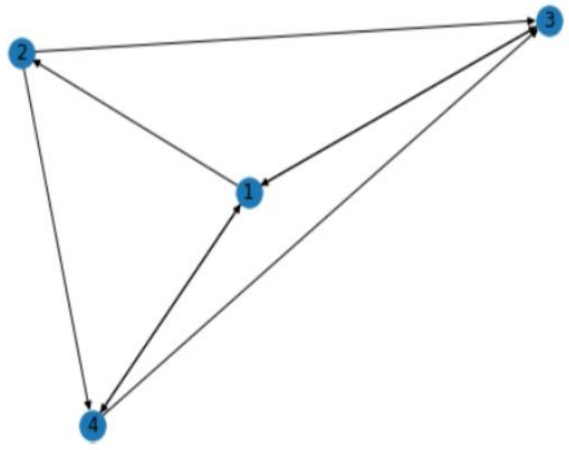
```
import numpy as np

import matplotlib.pyplot as plt

import networkx as nx

# To create an empty graph or initializing a graph

G = nx.DiGraph()

# creating pages as nodes

pages = ["1","2","3","4"]

G.add_nodes_from(pages)

# print nodes of graph

print(G.nodes())
```

```
['1', '2', '3', '4']
```

```
G.add_edges_from([('1','2'),('1','3'),('1','4'),('2','3'),('2','4'),('3','1'),('4','1'),('4','3')])
# printing outward links for each node   ['Page 1 = 3']  ['Page 2 = 2']
print("Number of outward links for each node:")
for page in pages:
   print(["Page %s = %s" %(page,str(len(G.out_edges(page))))])
```

```
Number of outward links for each node:
['Page 1 = 3']
['Page 2 = 2']
['Page 3 = 1']
['Page 4 = 2']
```

```
# we will print graph using matplotlib
nx.draw(G, with_labels=True)
plt.show()
```



```
linkmatrix = np.matrix([[0,0,1,1/2],
                        [1/3,0,0,0],
                        [1/3,1/2,0,1/2],
                        [1/3,1/2,0,0]])
def findPageRank(linkmatrix,pages):
    eigval,eigvector = np.linalg.eig(linkmatrix)
    final_eigval = np.abs(eigval).max()
    PageRank = np.where(eigval==final_eigval)
    print("The most important page is %s" %(str(pages[PageRank[0][0]])))
findPageRank(linkmatrix,pages)
```

```
The most important page is 1
```

Tableau

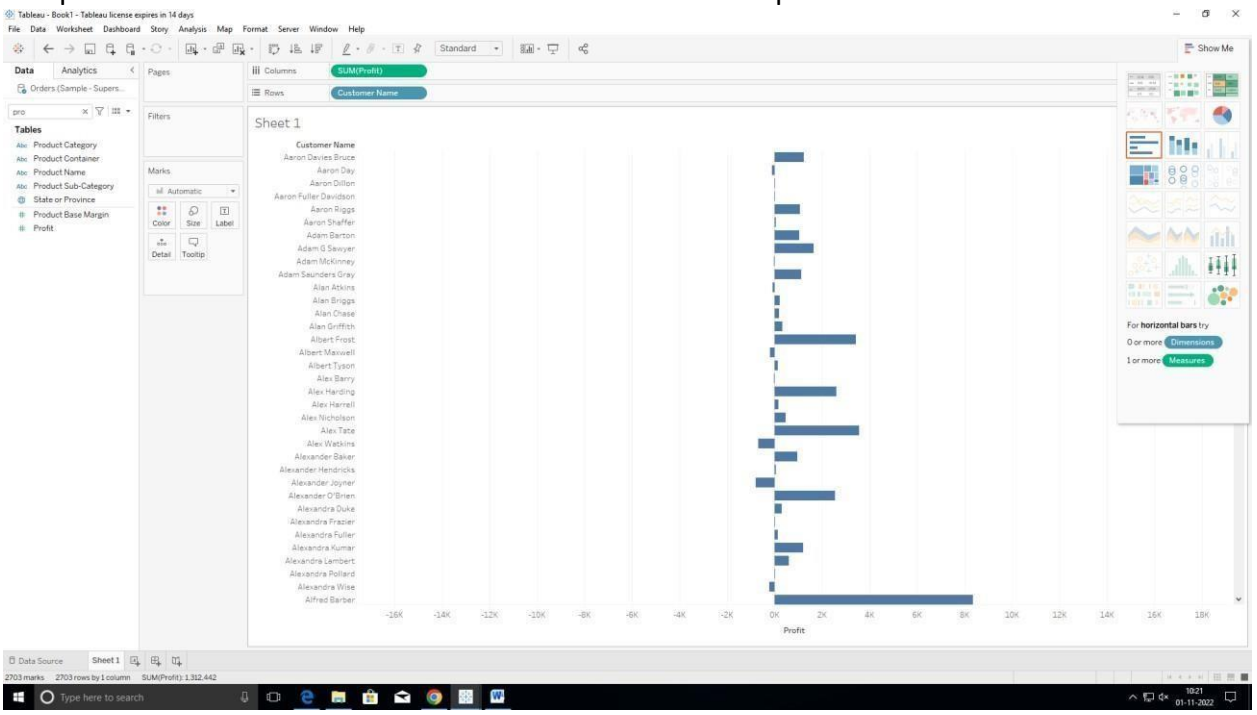
- Open Tableau public and connect to the data source.

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

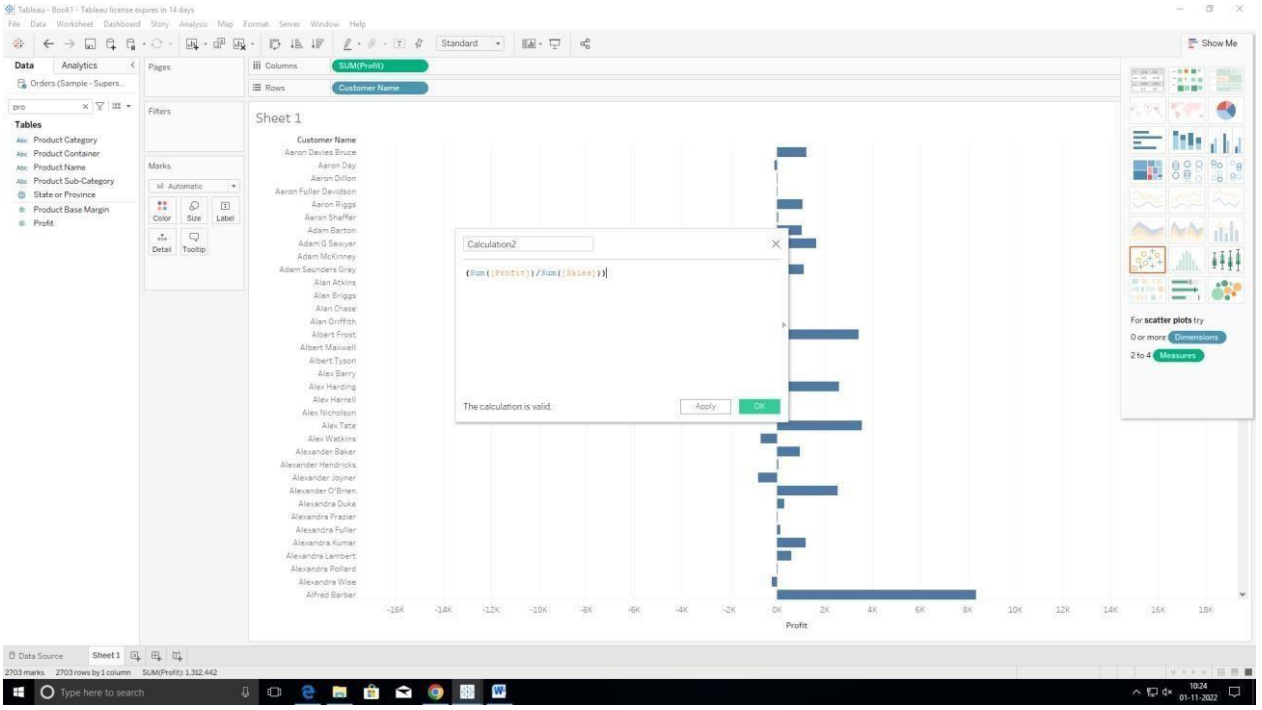
Step 1: Open the superstoreus2015 excel data set

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

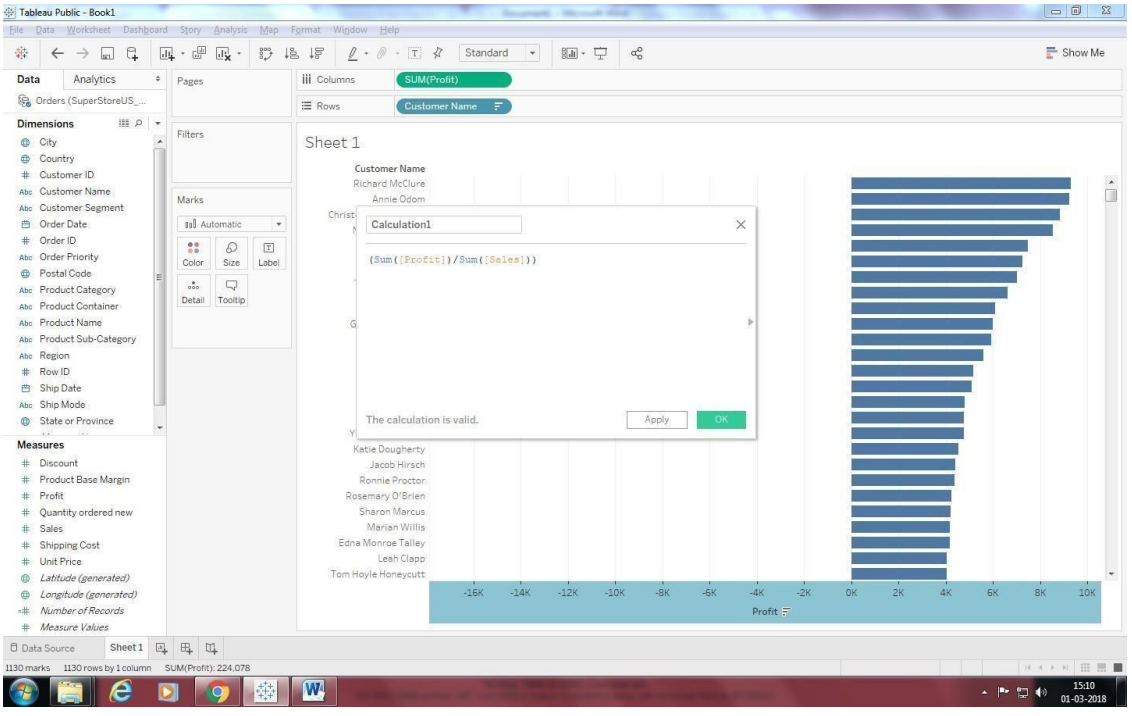


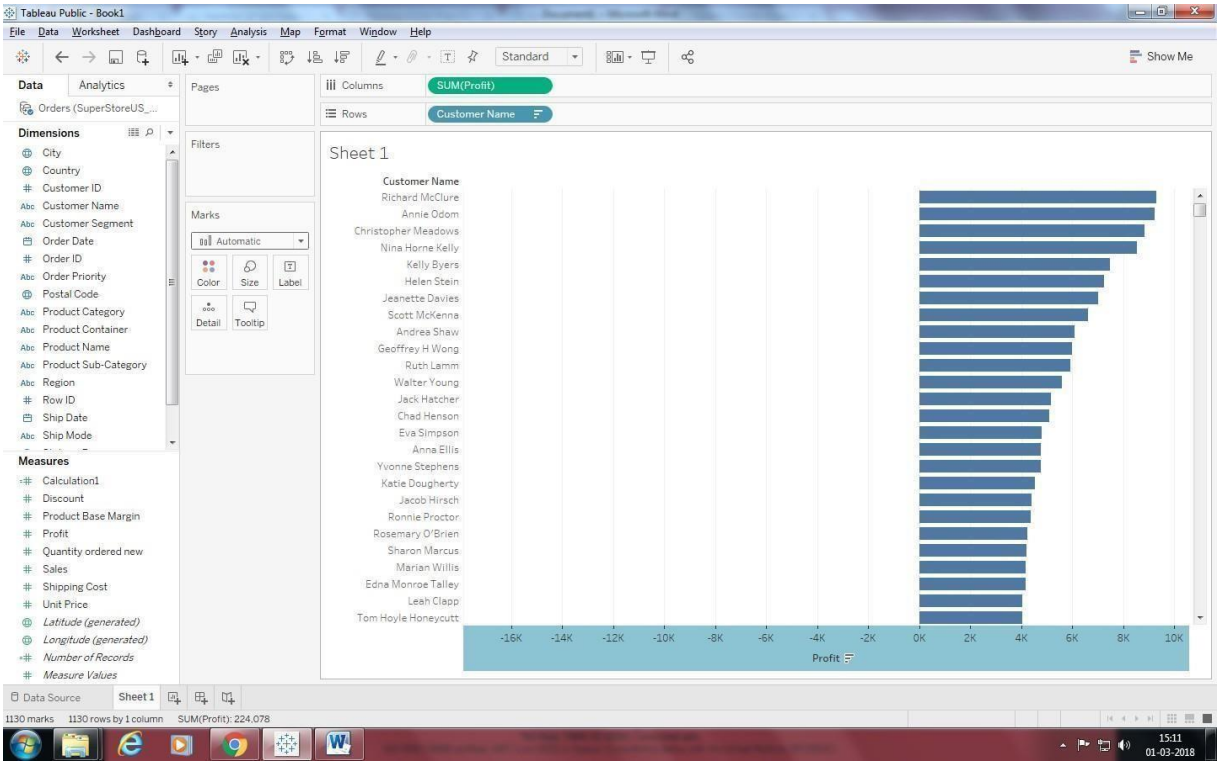
Step 5: To calculate profit Ratio

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

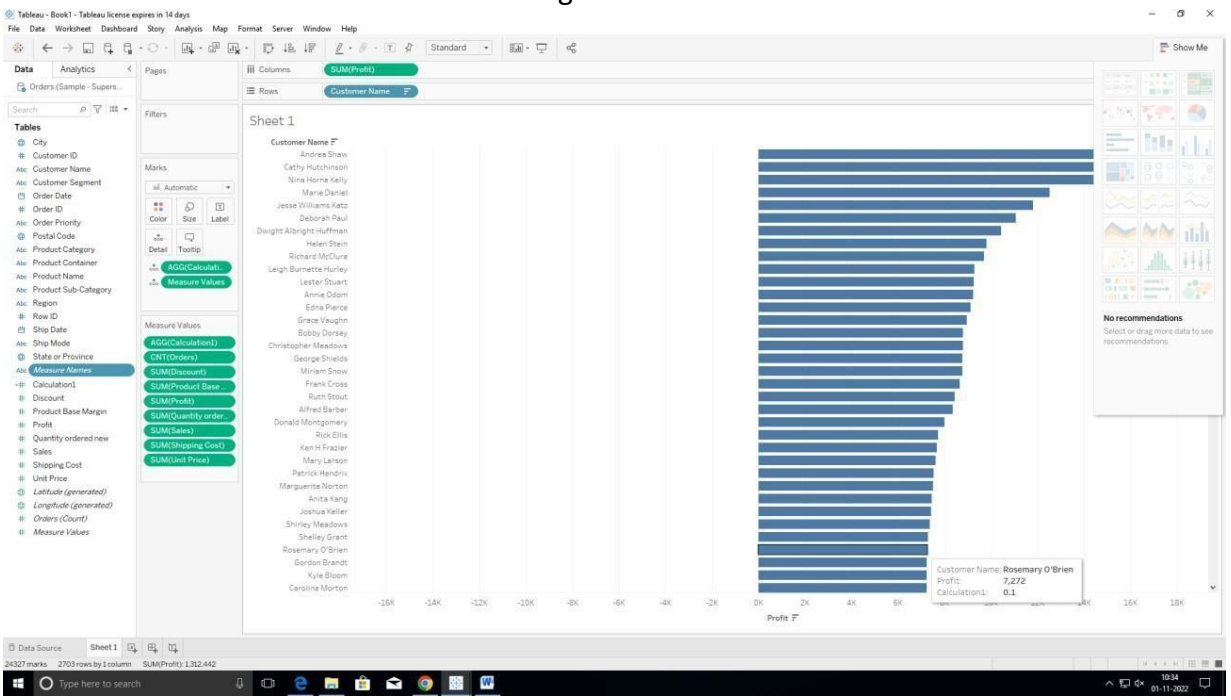
This formula needs to be entered as tooltip or label

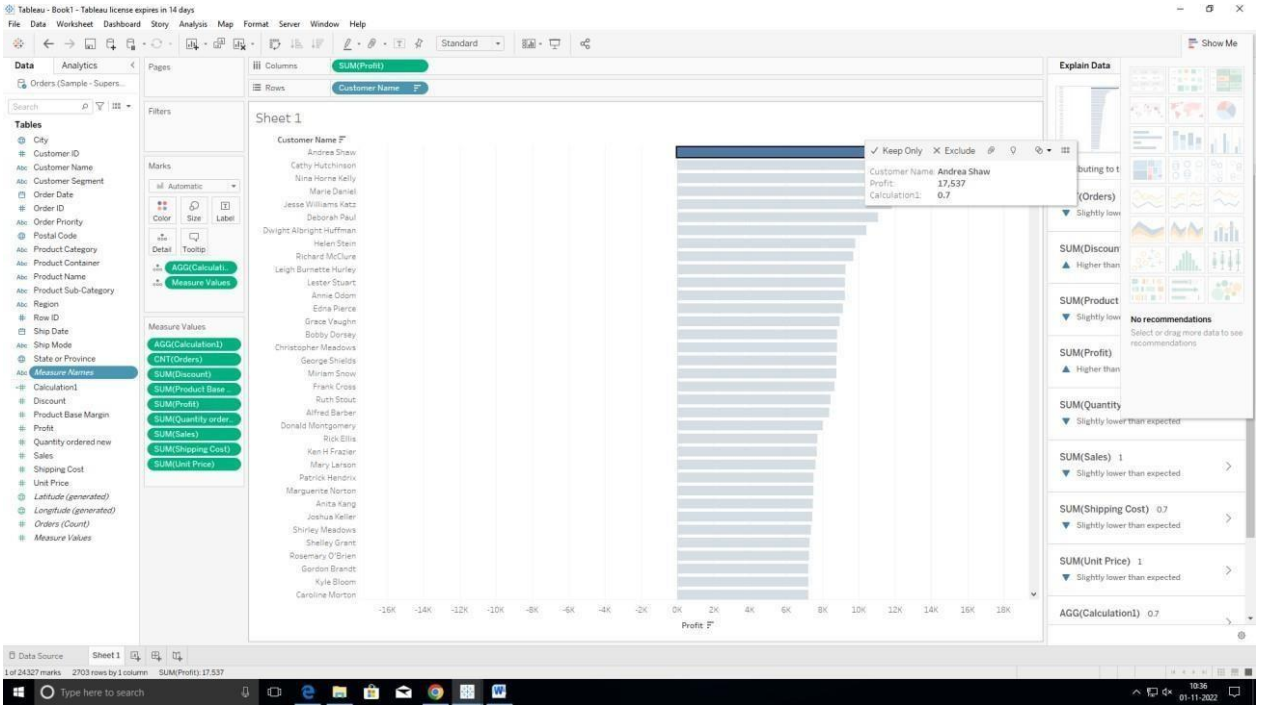
Click on Analysis>Create Calculated Field and enter the formula



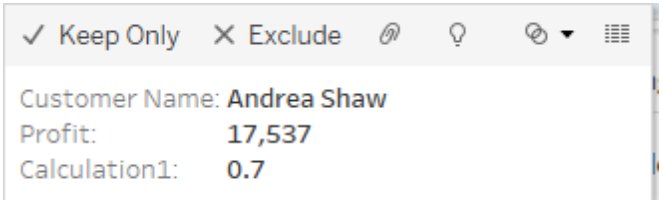


You can see Calculation1 in measures. Drag is to Marks area.



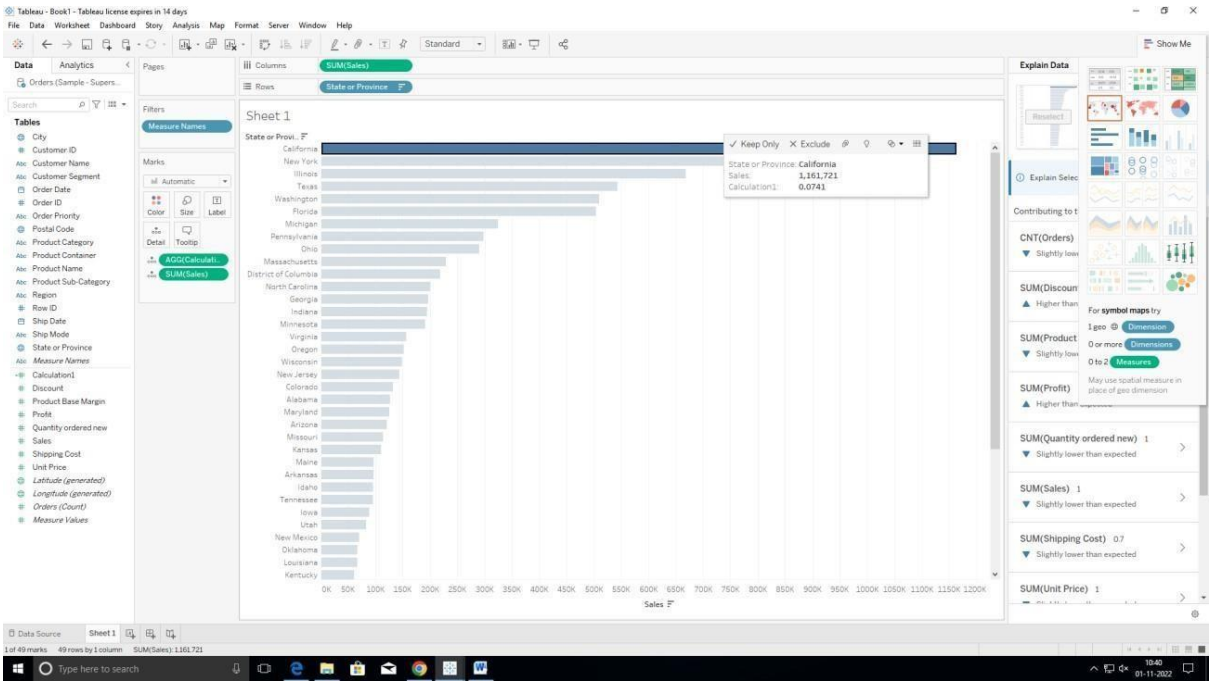


Final answer is:

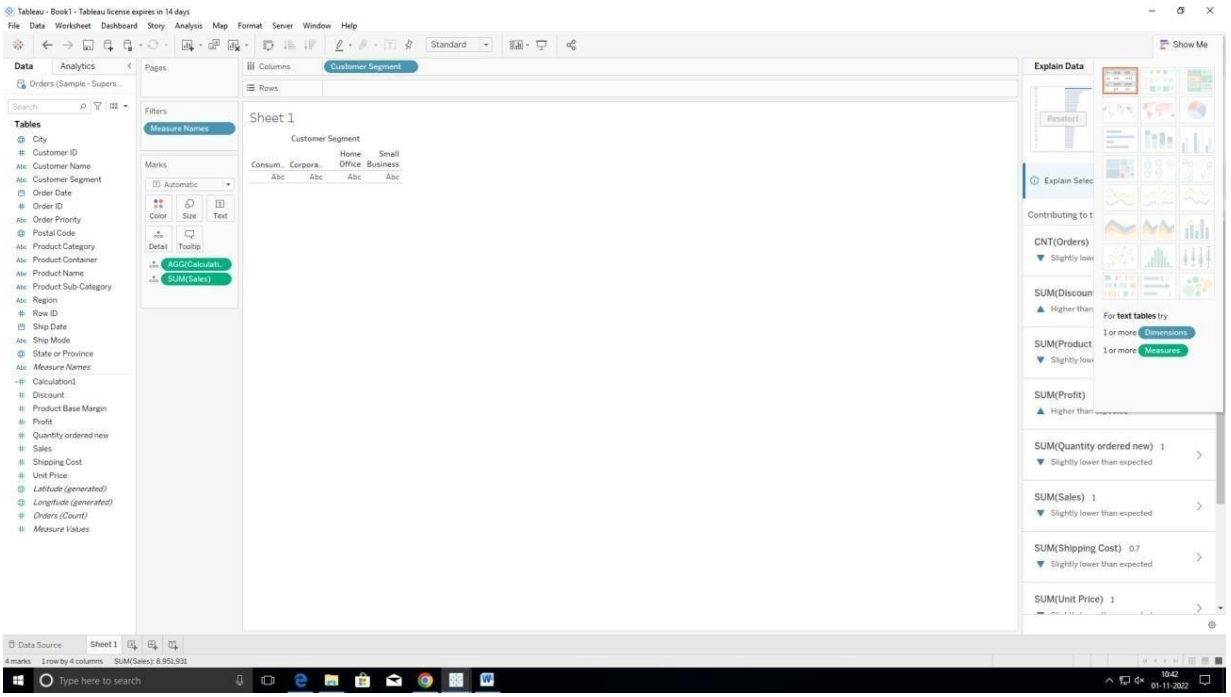


Q2: Which state has the highest Sales (Sum)? What is the total Sales for that state?

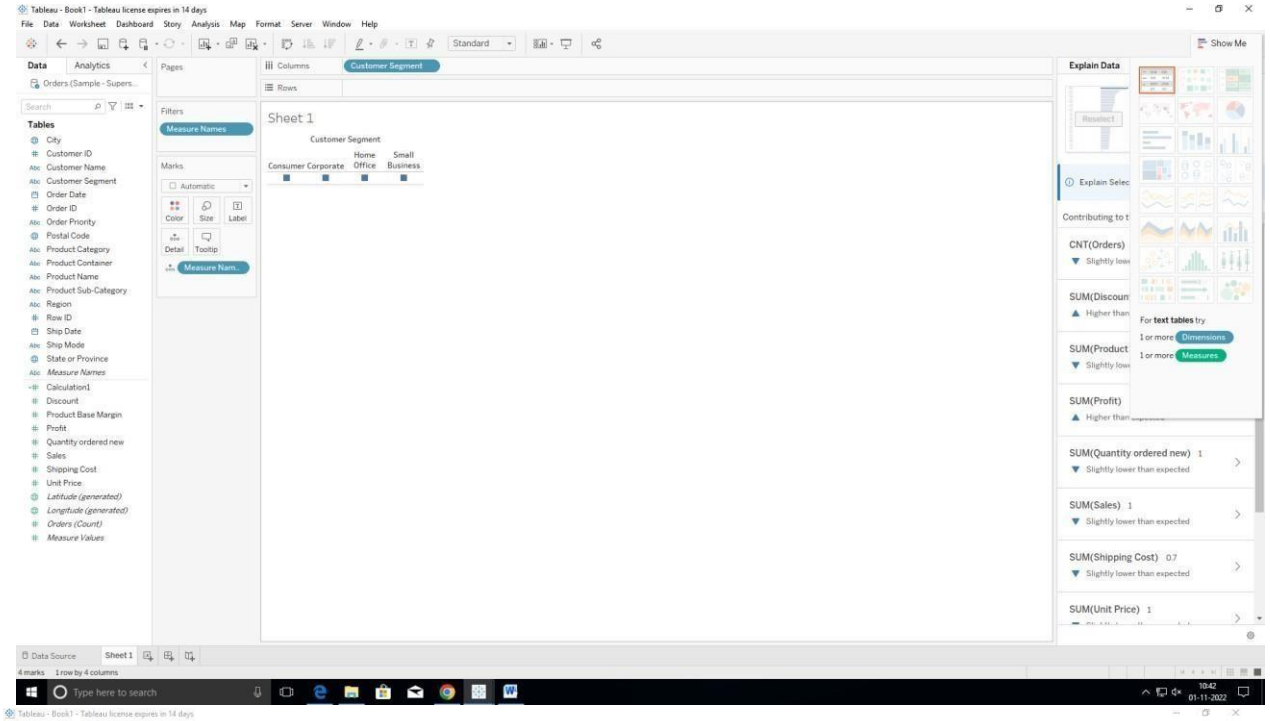
Ans:

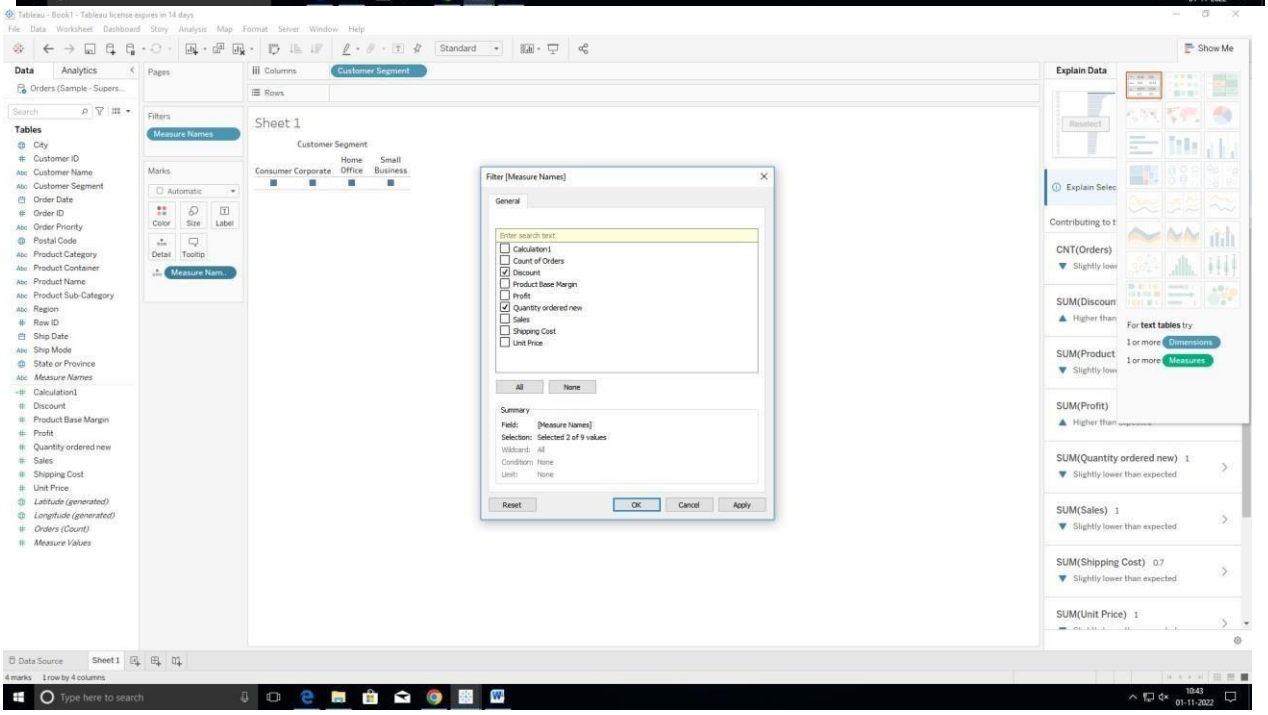


Q 3: Which customer segment has both the highest order quantity and average discount rate? What is the order quantity and average discount rate for that state?

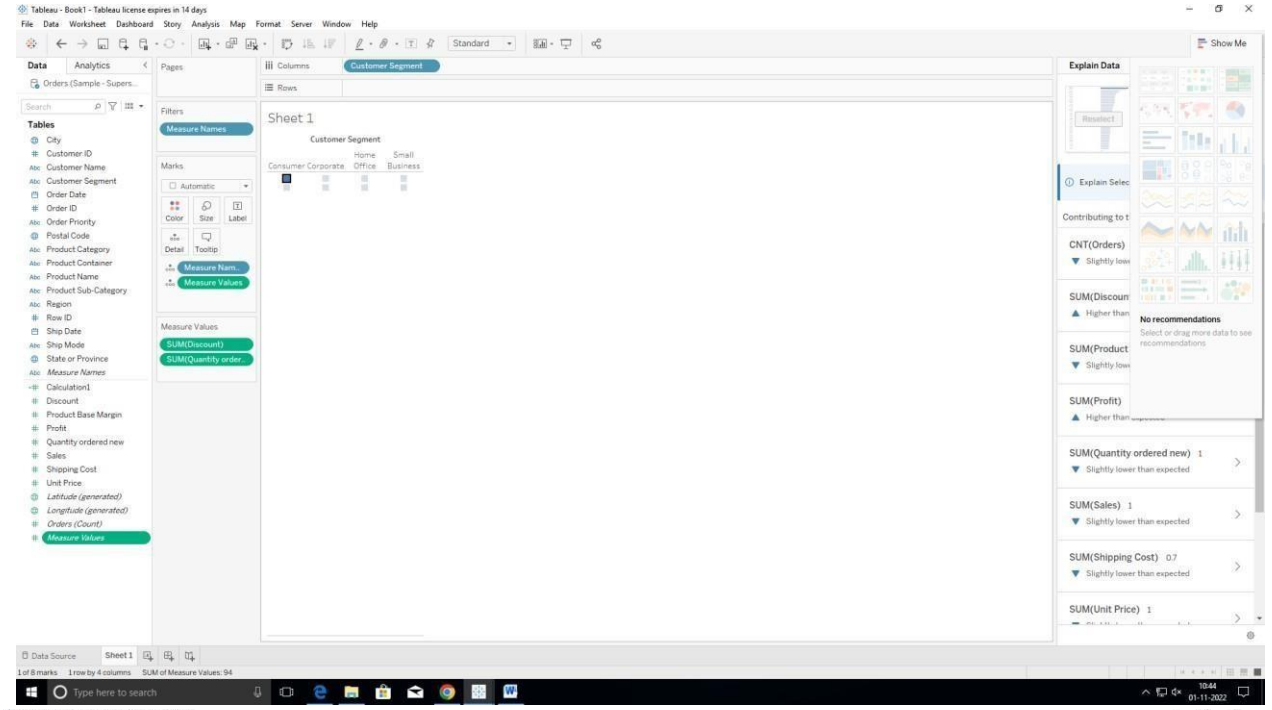


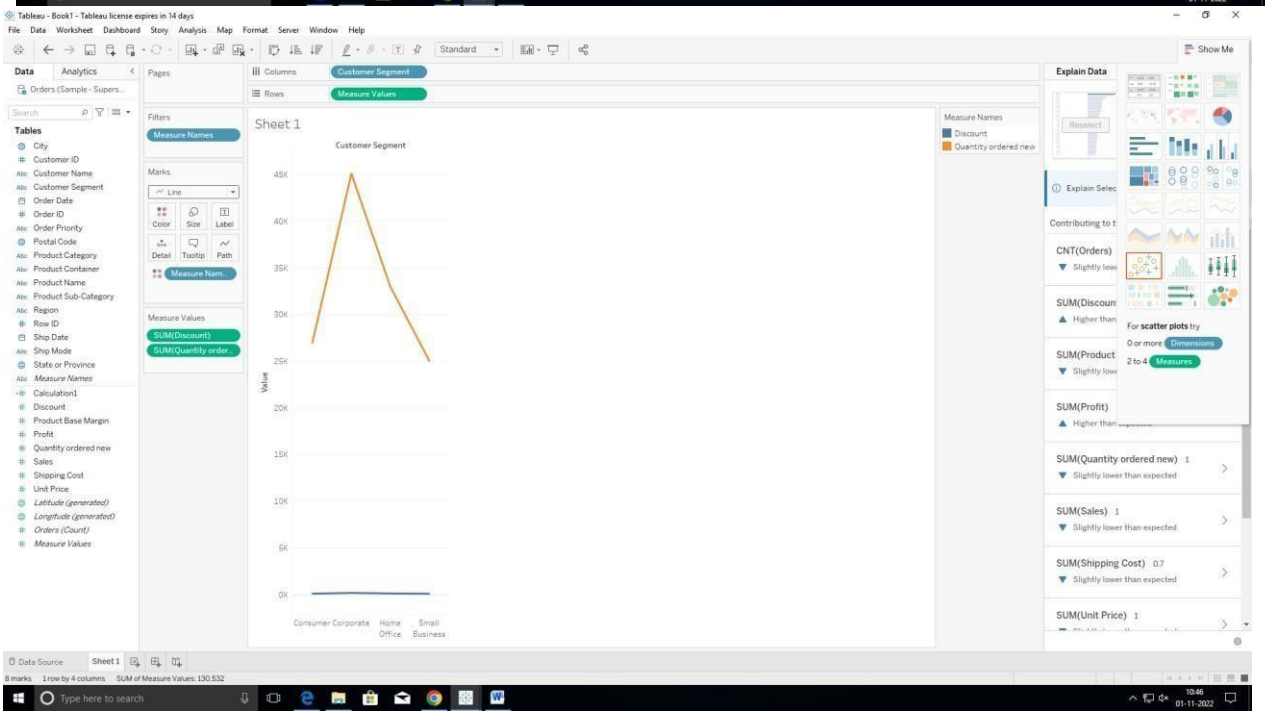
Name: Shubham Sambhaji Kadam Class: SYMCA Div: B Roll No.: 2021112





Name: Shubham Sambhaji Kadam Class: SYMCA Div: B Roll No.: 2021112

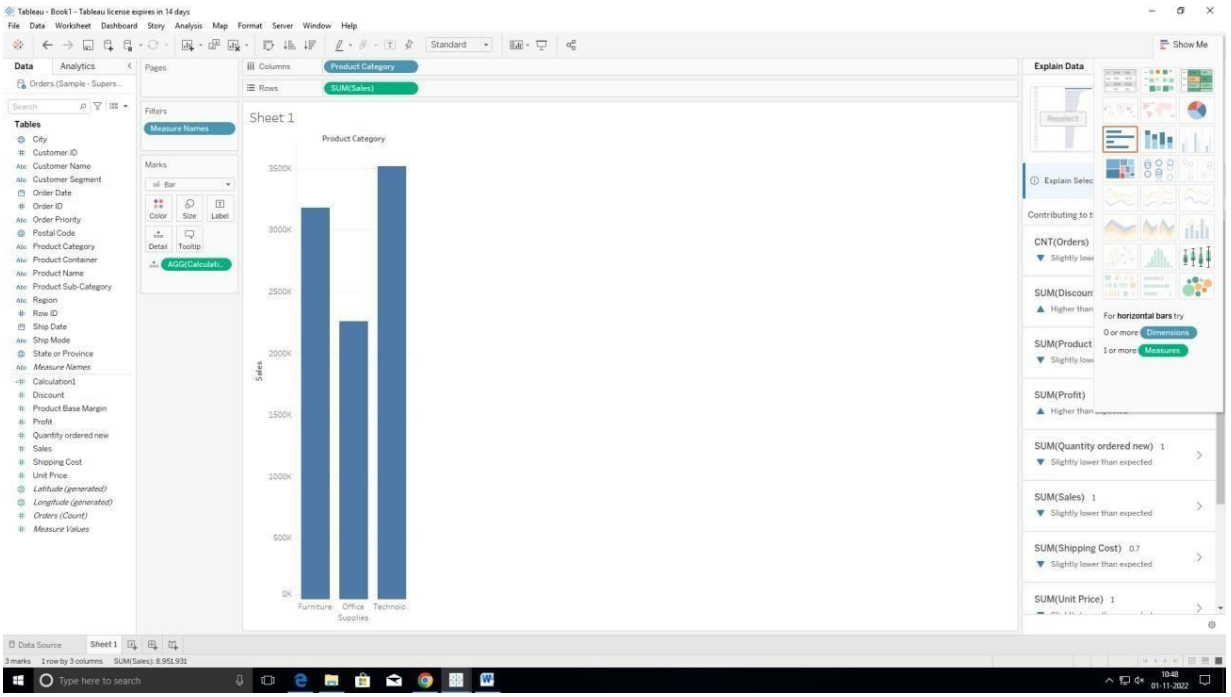




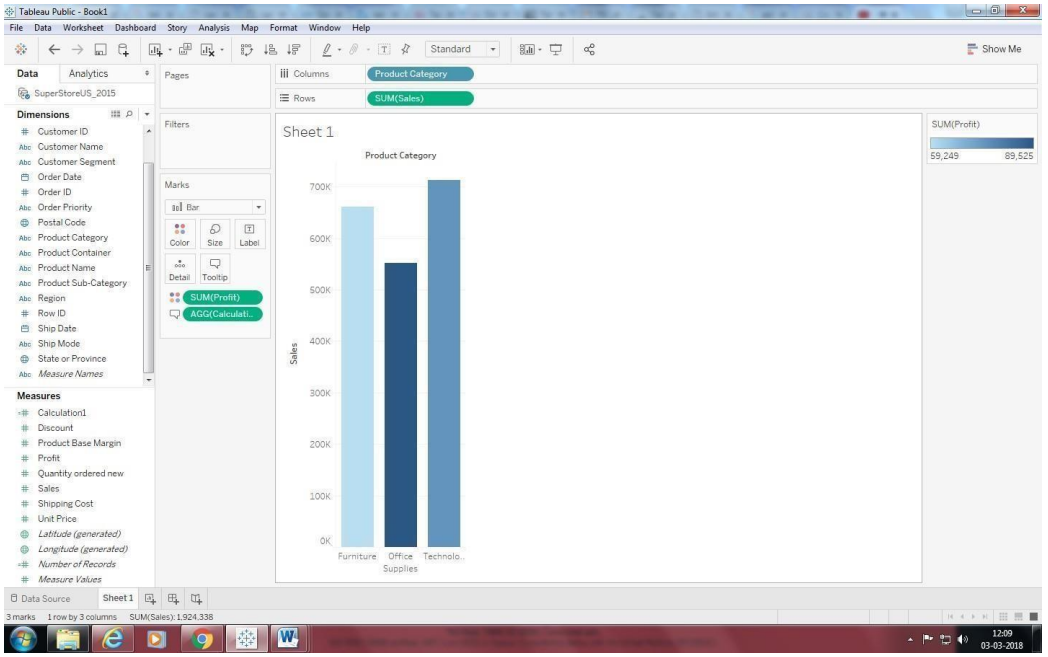
Q 4: Which Product Category has the highest total Sales? Which Product Category has the worst Profit? Name the Product Category and $ amount for each.

Ans:

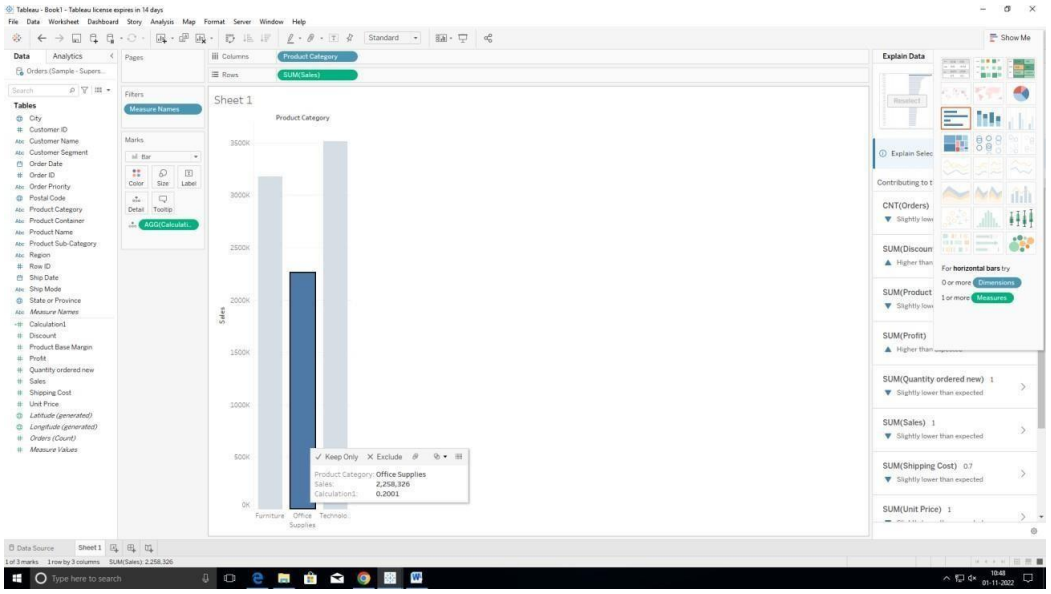
a. Bar Chart displaying total Sales for each Product Category



b. Add a color scale indicating Profit

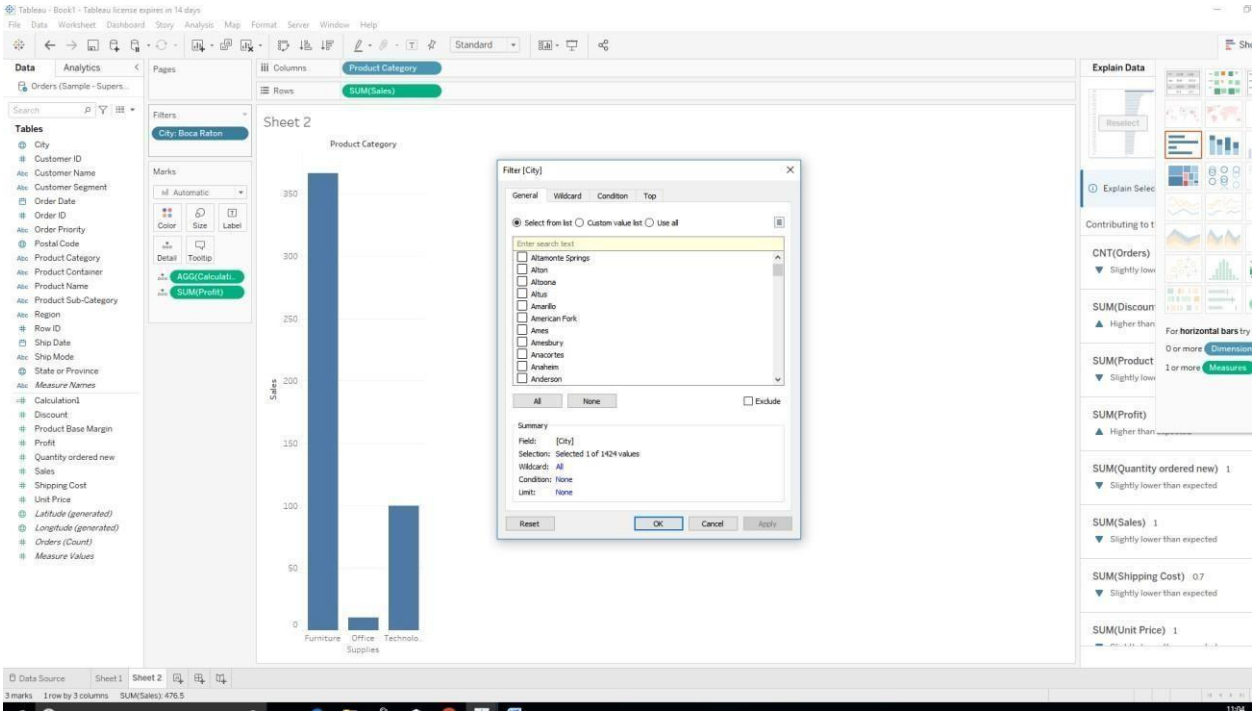


c. Each Product Category labeled with total Sales and Each Product Category labeled with Profit

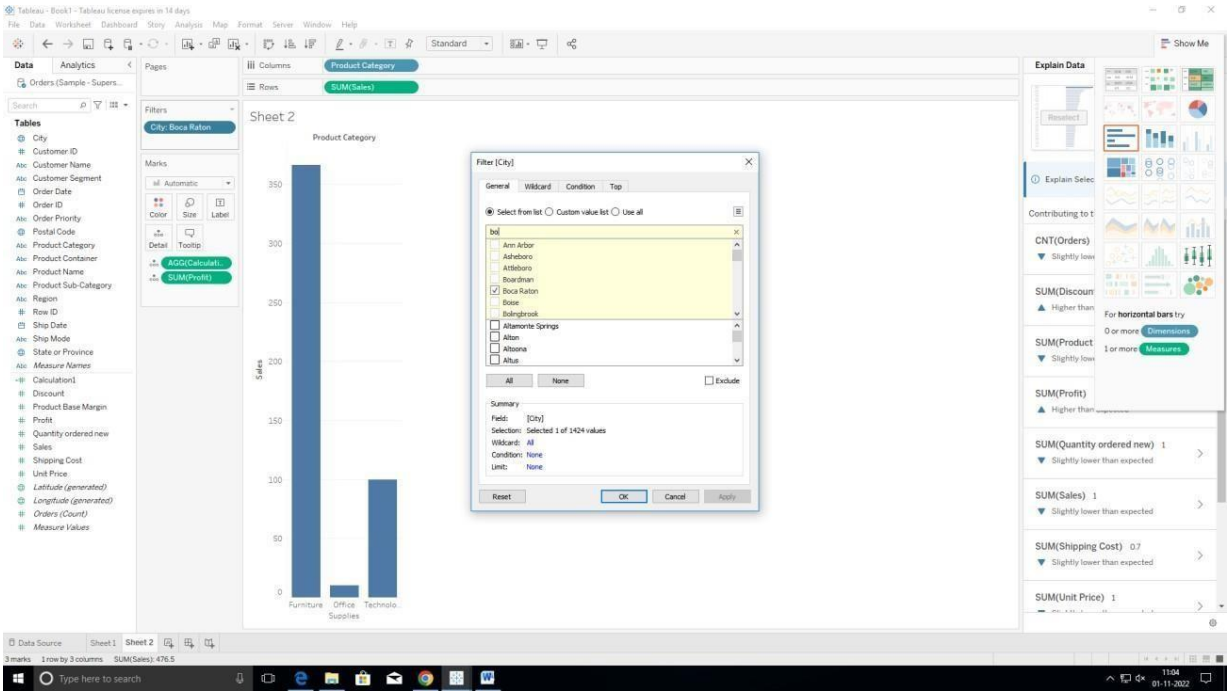


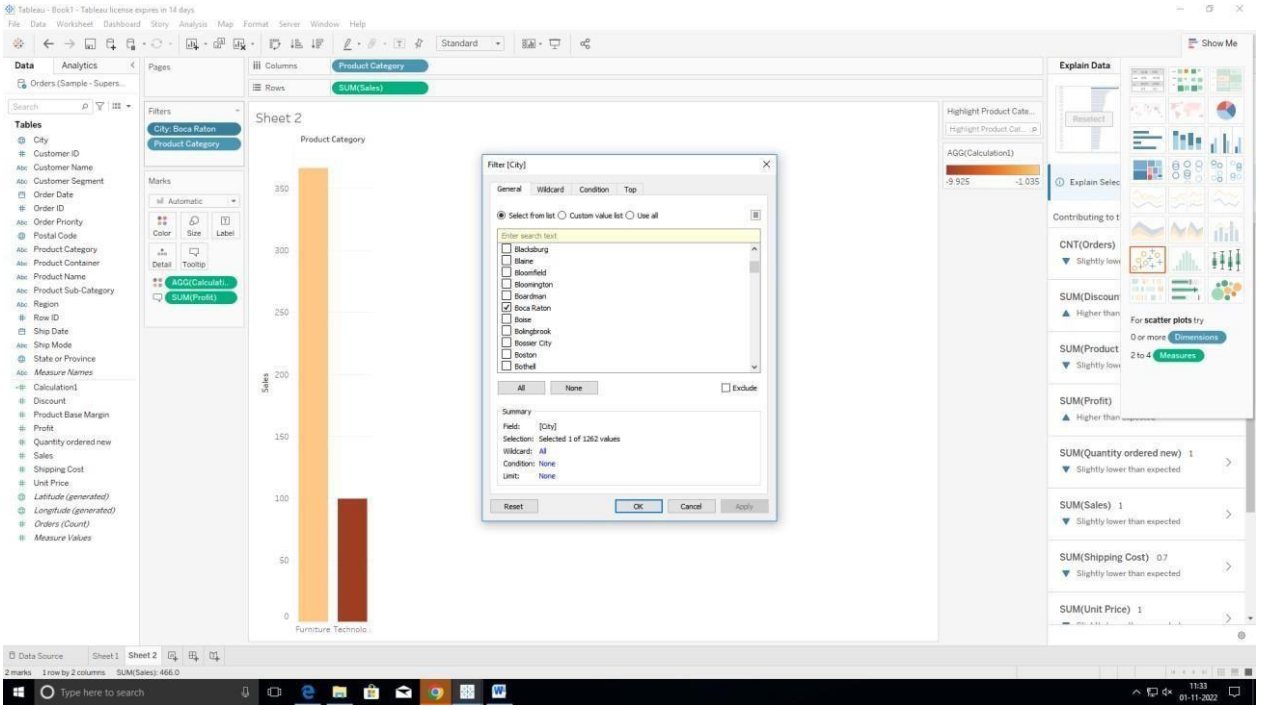
Q 5: Use the same visualization created for Question #4.What was the Profit onTechnology (Product Category) in Boca Raton (City) ?

Add Filter City

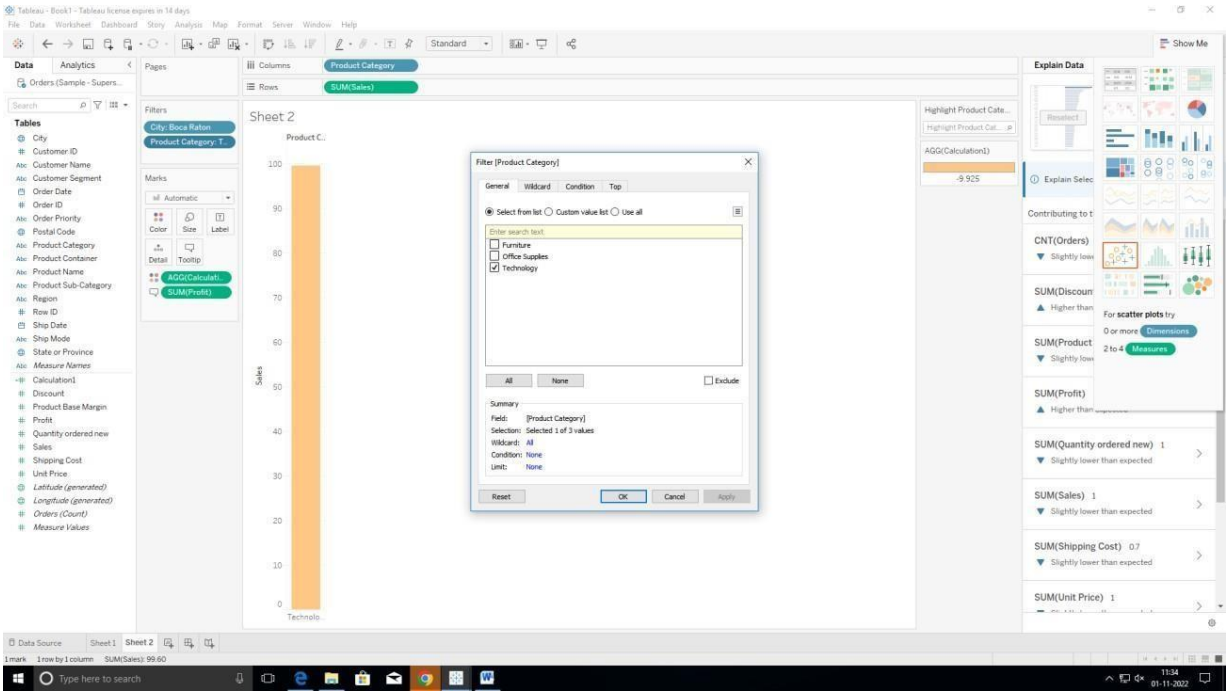


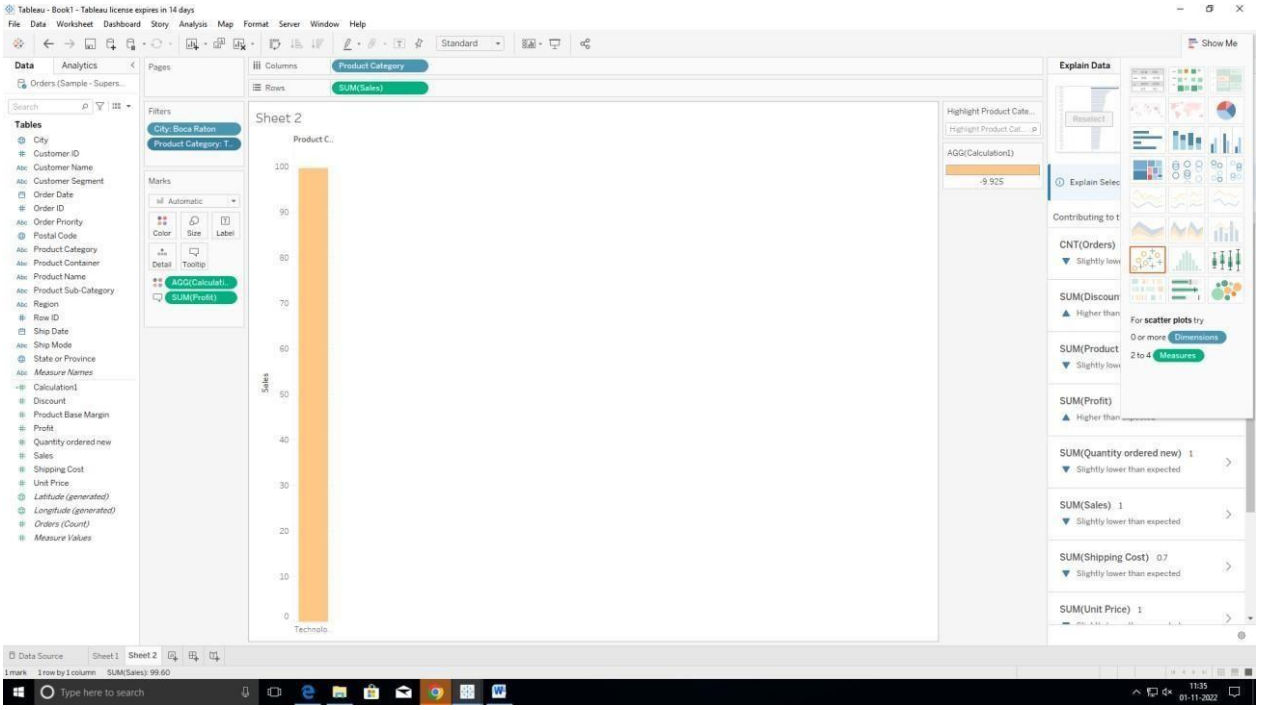
Select the city to show a single bar for the city





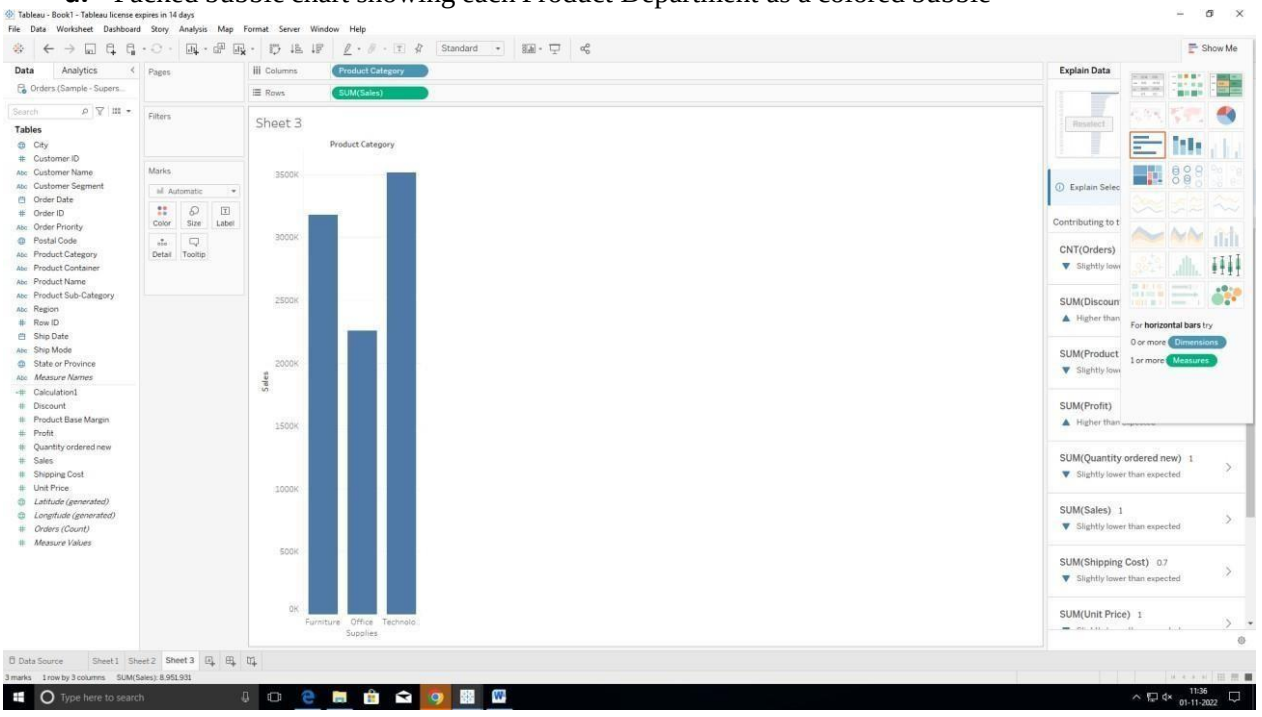
Apply a filter for Technology



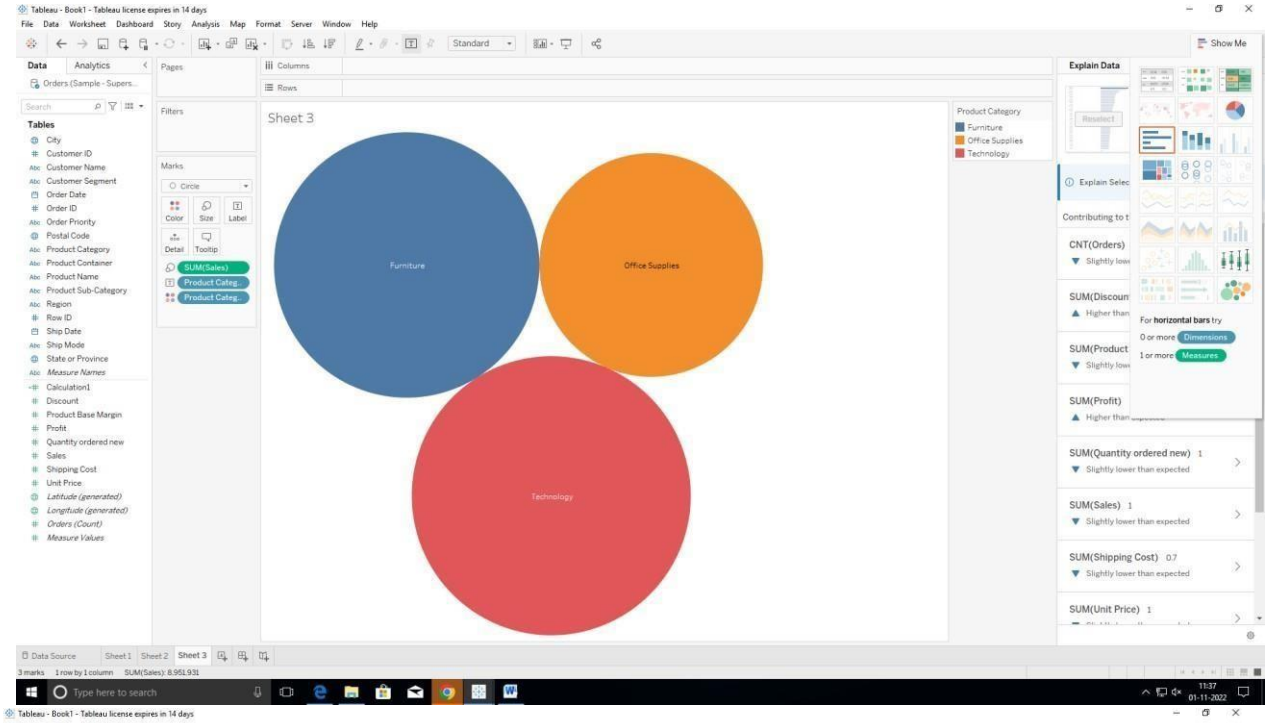


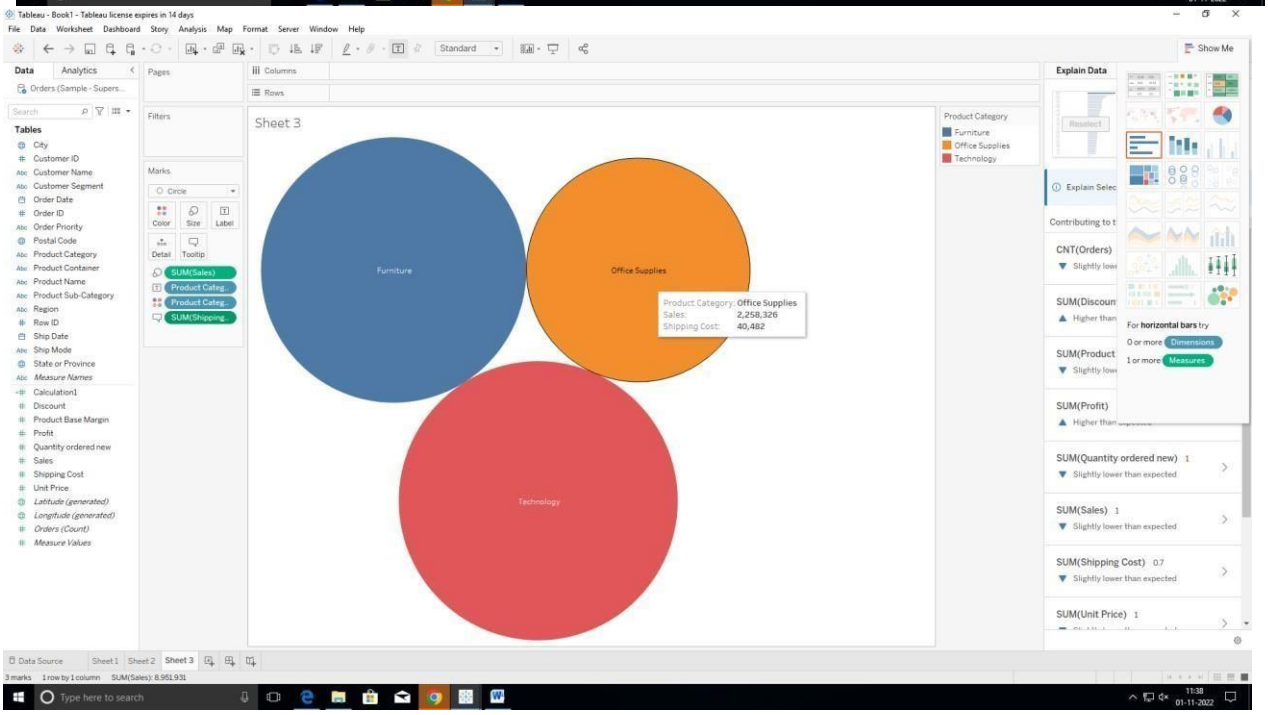
Q 6: Which Product Department has the highest Shipping Costs? Name the Departmentand cost.

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



b. Use Shipping Cost to display the size of each bubble

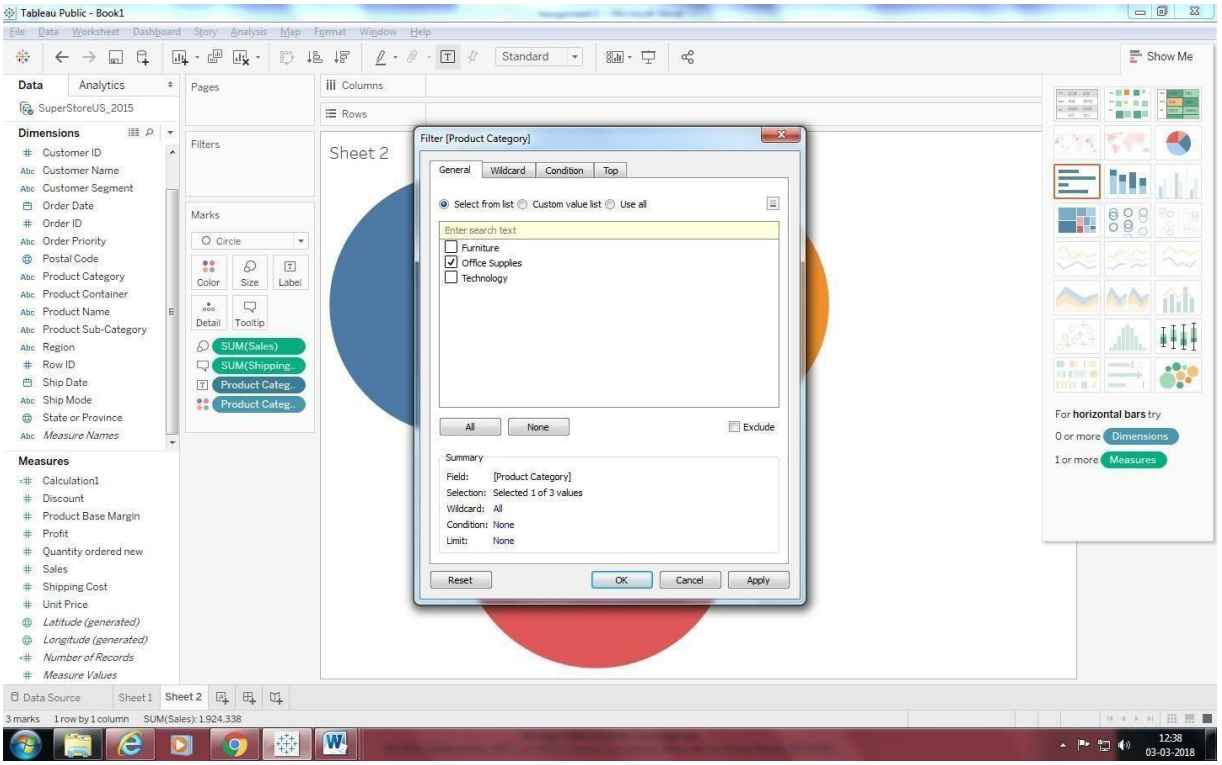




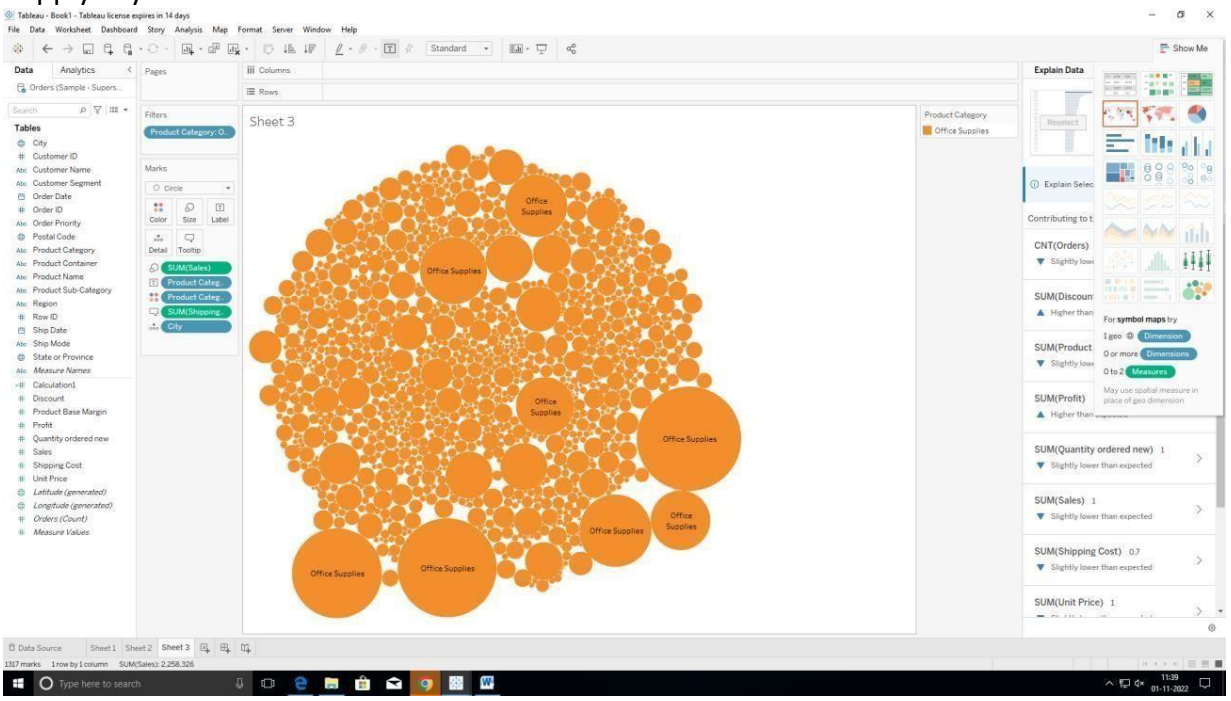
Q 7: Use the same visualization created for Question #6. What was the shipping cost ofOffice Supplies for Xerox 1905 in the Home Customer Segment in Cambridge?

Ans: Apply filters as per requirement

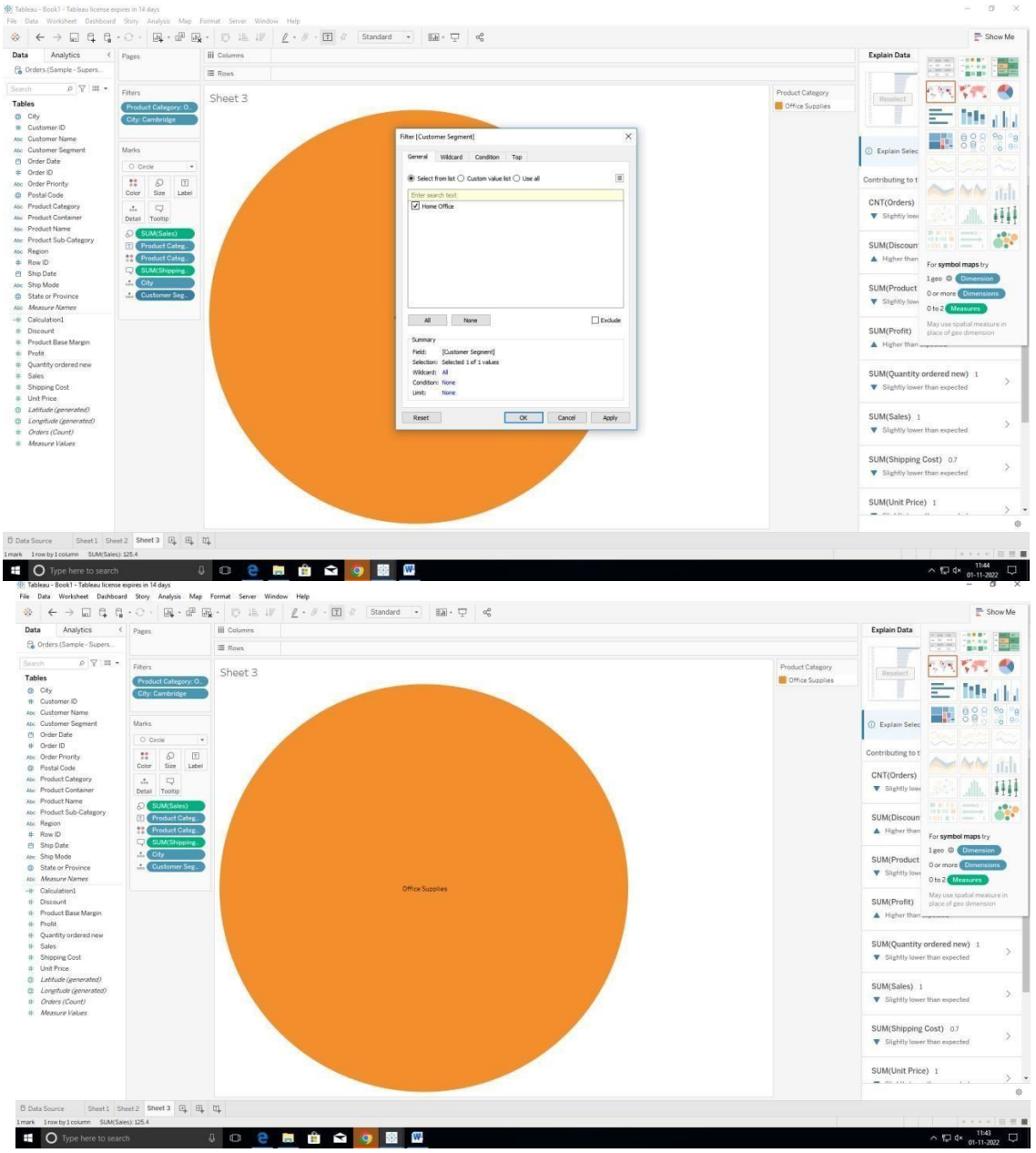
=

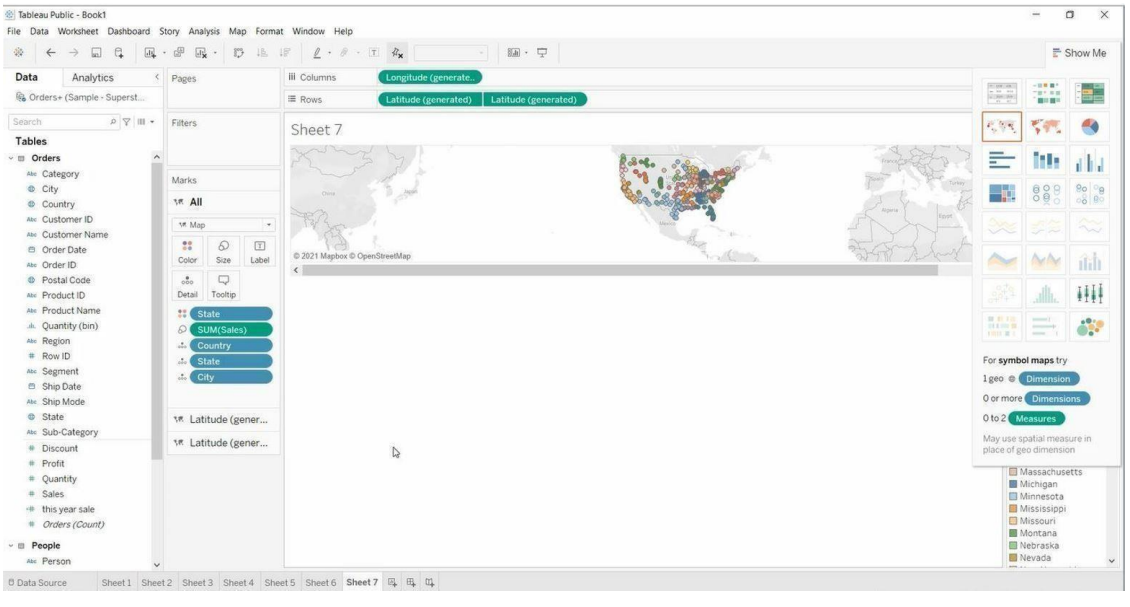


Apply city Filter



Name: Shubham Sambhaji Kadam Class: SYMCA Div: B Roll No.: 2021112





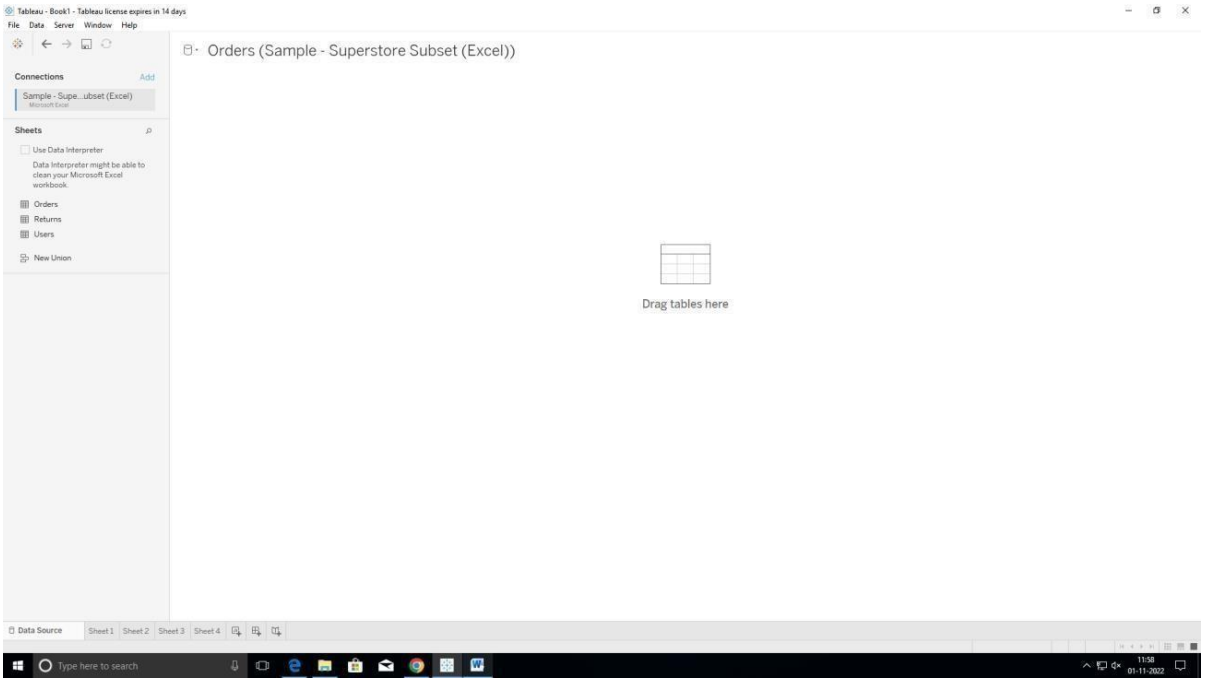
Assignment 2: Preparing Maps

Data Set for this Lab Sample- Superstore

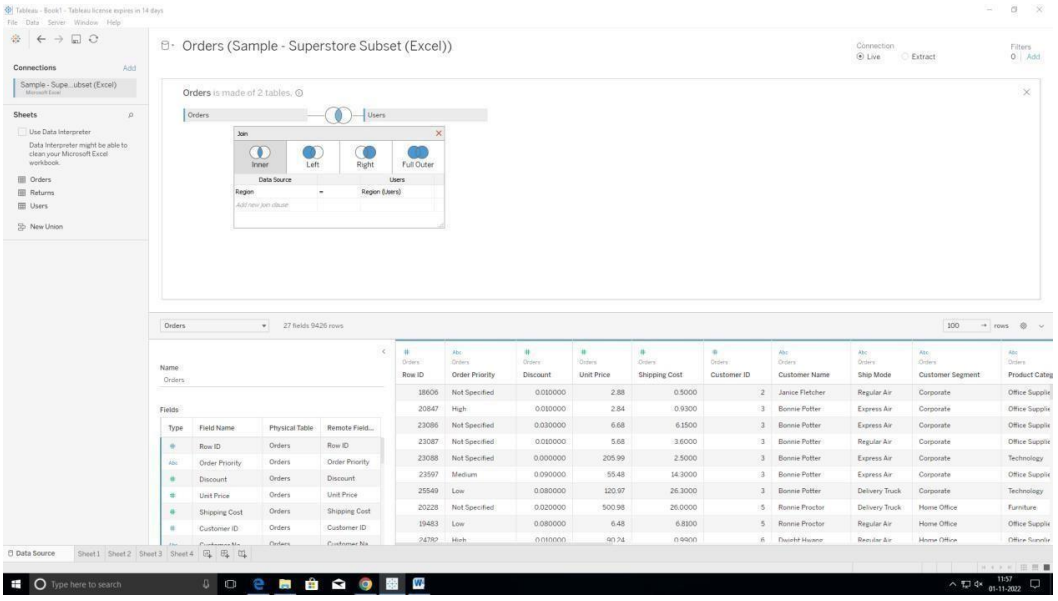
Q 1: Prepare a Geographic map to show sales in each state.

Ans:

a) Connect to dataset



b) Join sheets



c. Create a Geographic Hierarchy

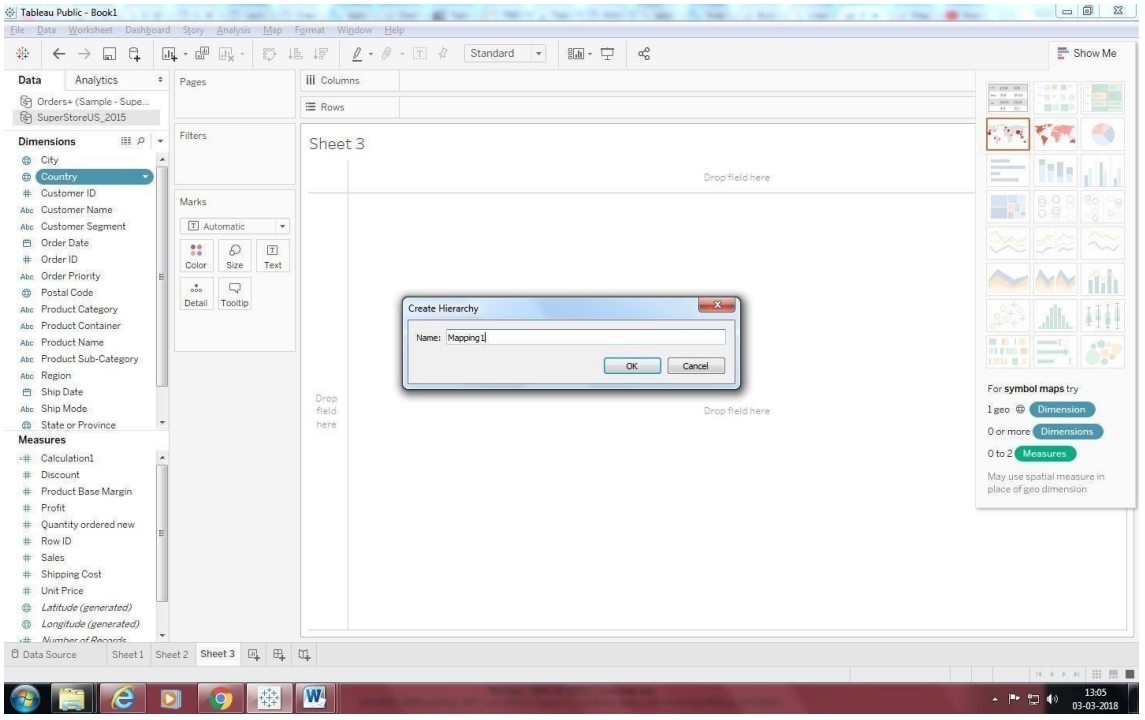
1. In the Data pane, right-click the geographic field, **Country**, and then select **Hierarchy** > **Create Hierarchy**.

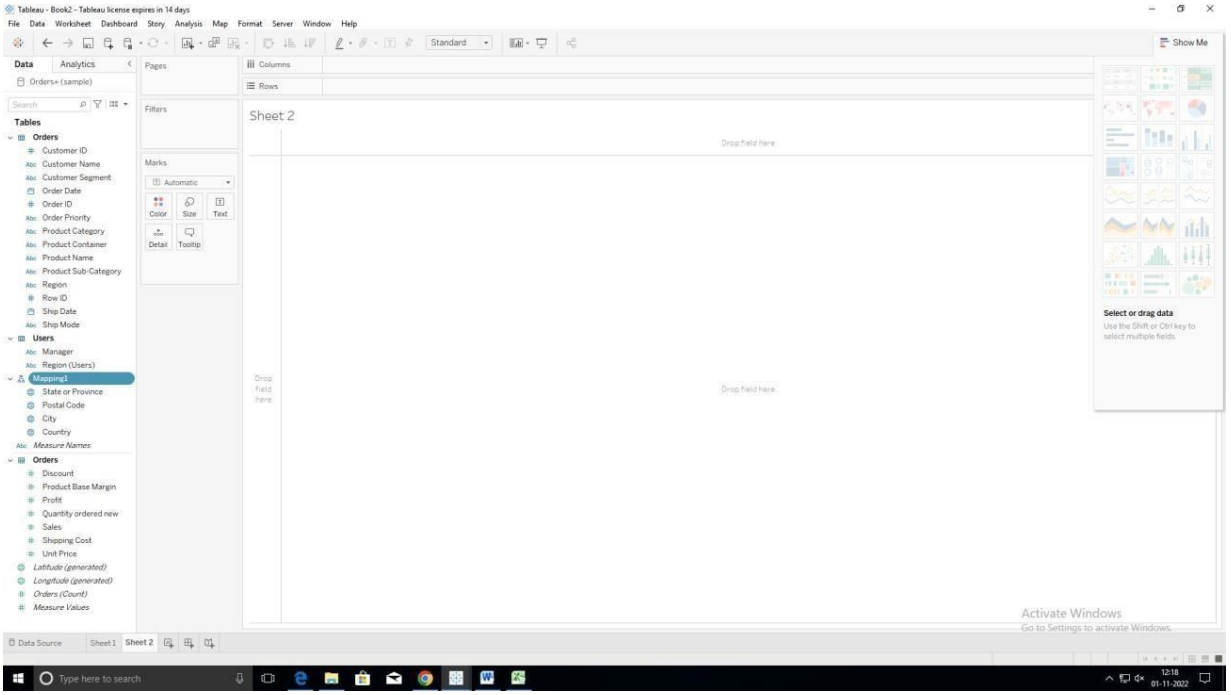
2. In the Create Hierarchy dialog box that opens, give the hierarchy a name, such as

Mapping Items, and then click **OK**.

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

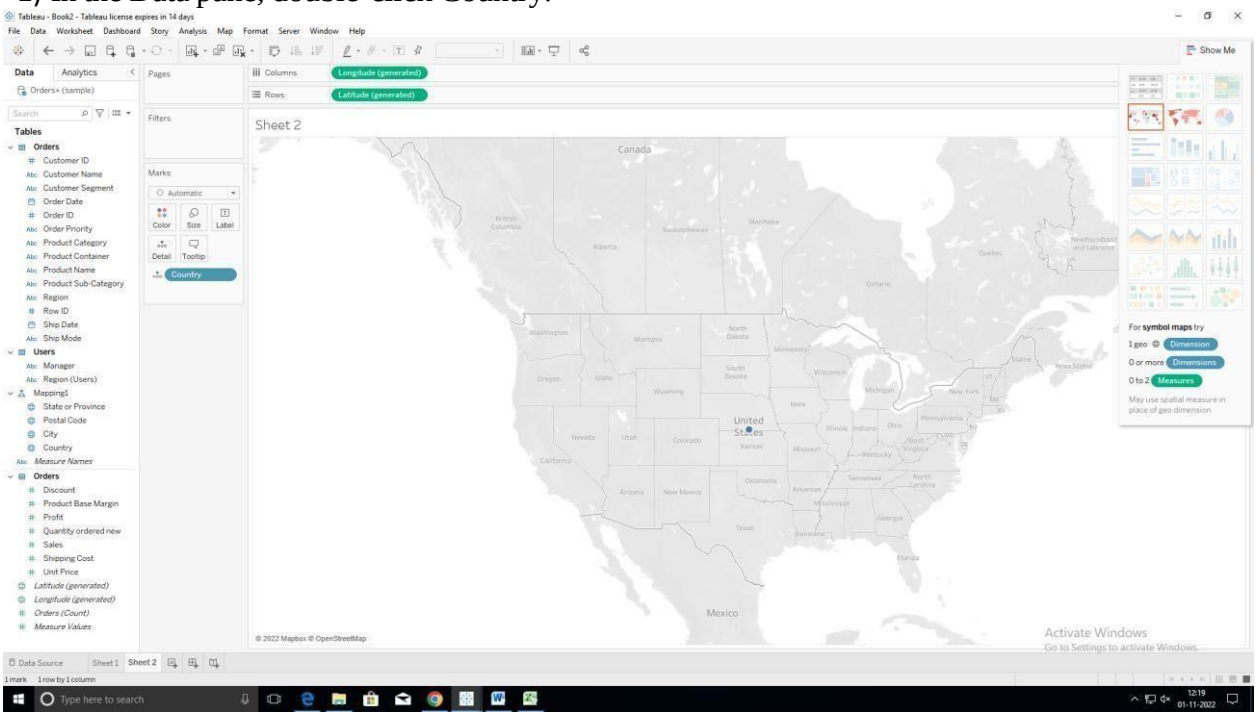
3. In the Data pane, drag the State field to the hierarchy and place it below the Country field.
4. Repeat step 3 for the City and Postal Code fields.



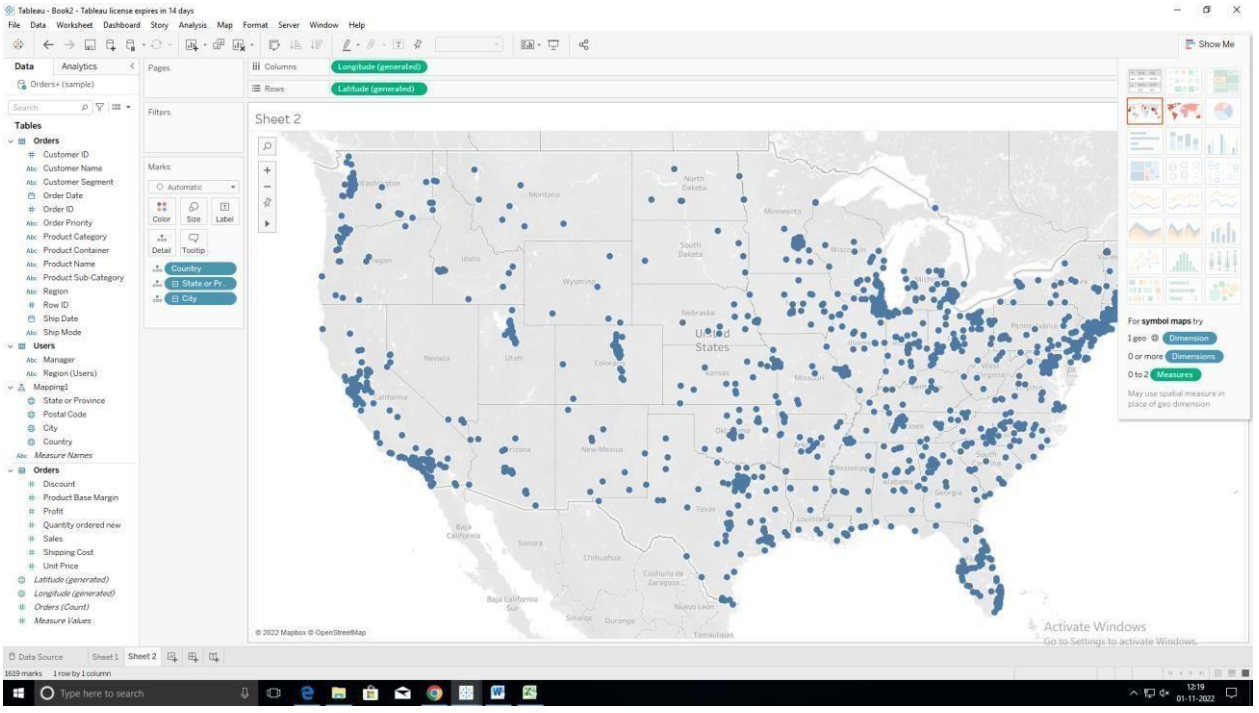


d. Build a basic map

1) In the Data pane, double-click Country.



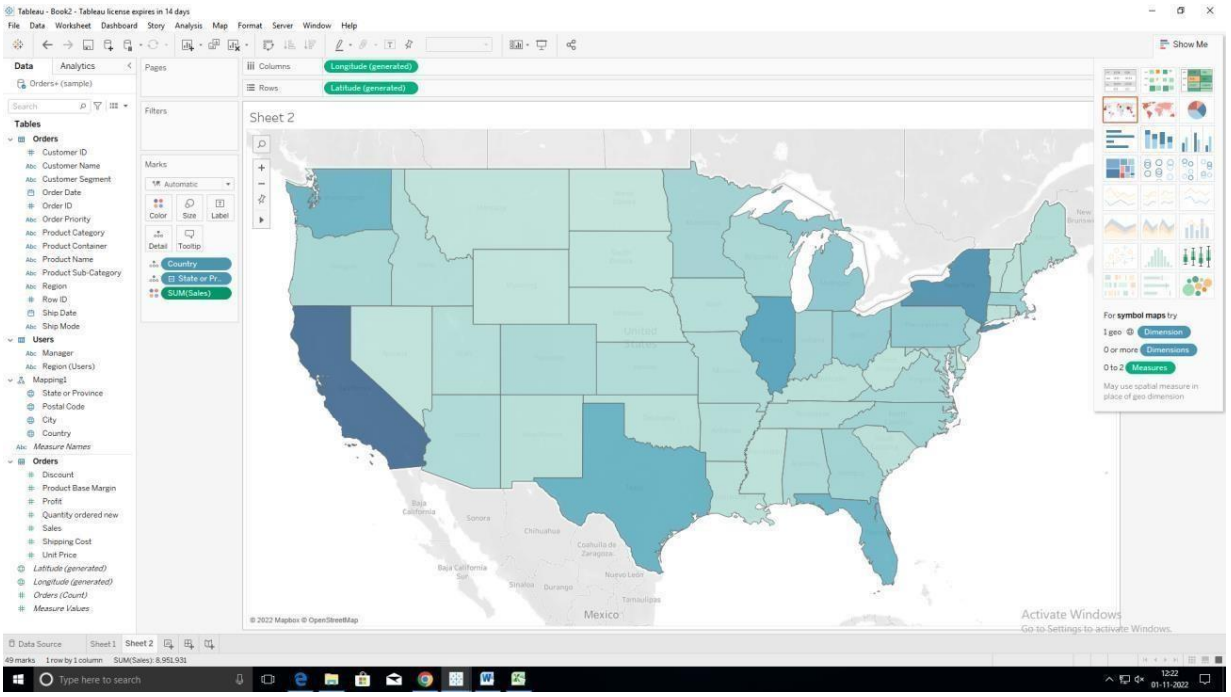
2) On the Marks card, click the + icon on the Country field.



e. Add visual

detailAdd color

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

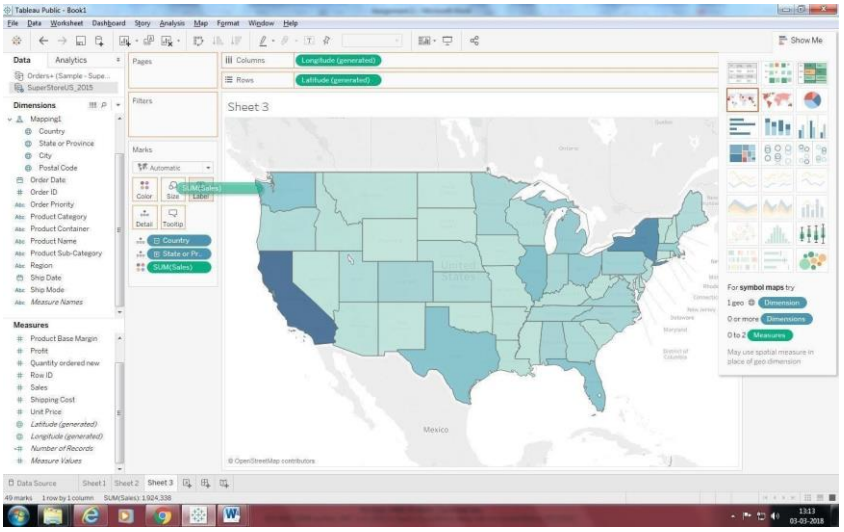


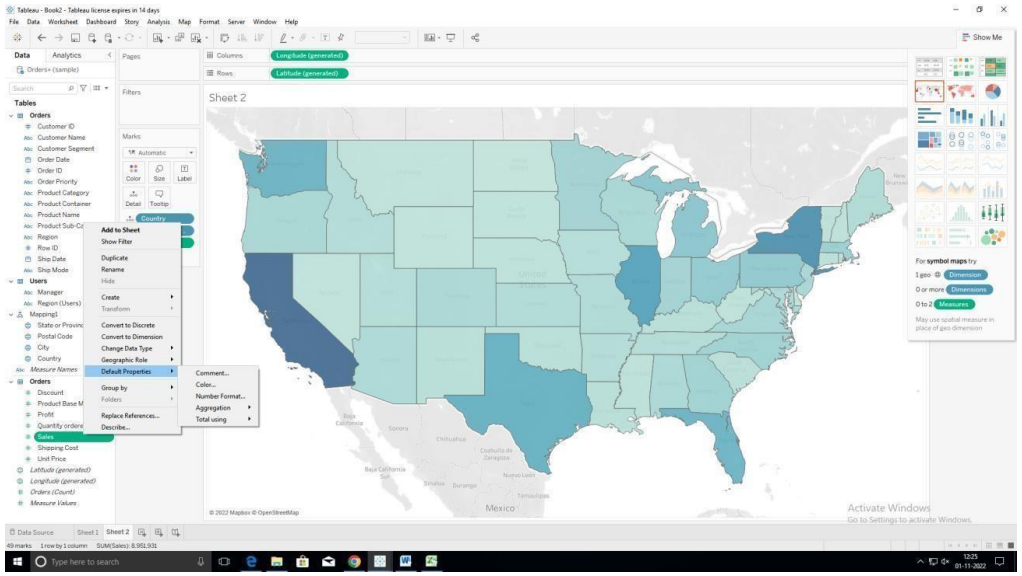
Add labels

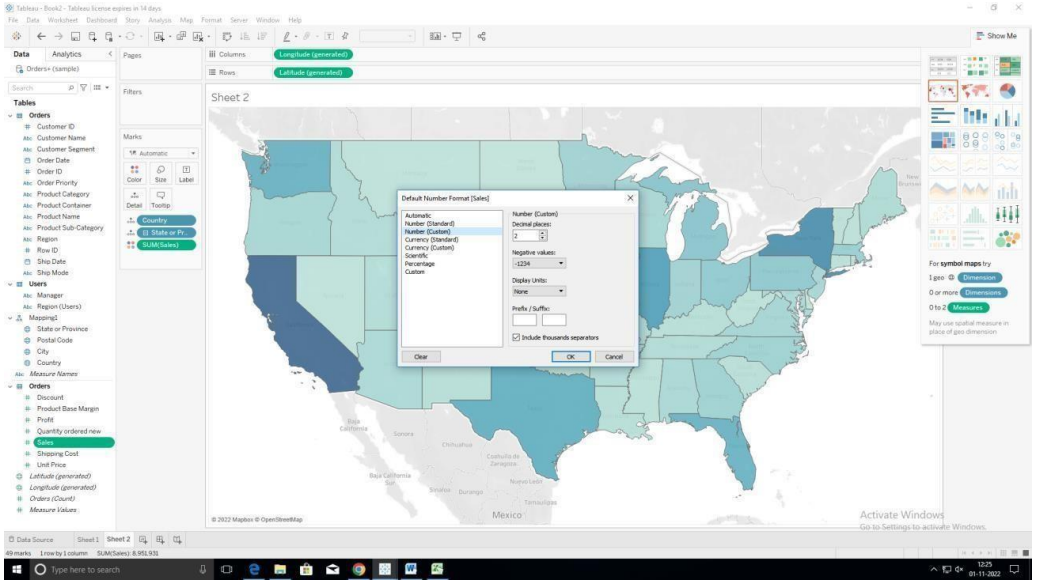
1. From Measures, drag Sales to Label on the Marks card.

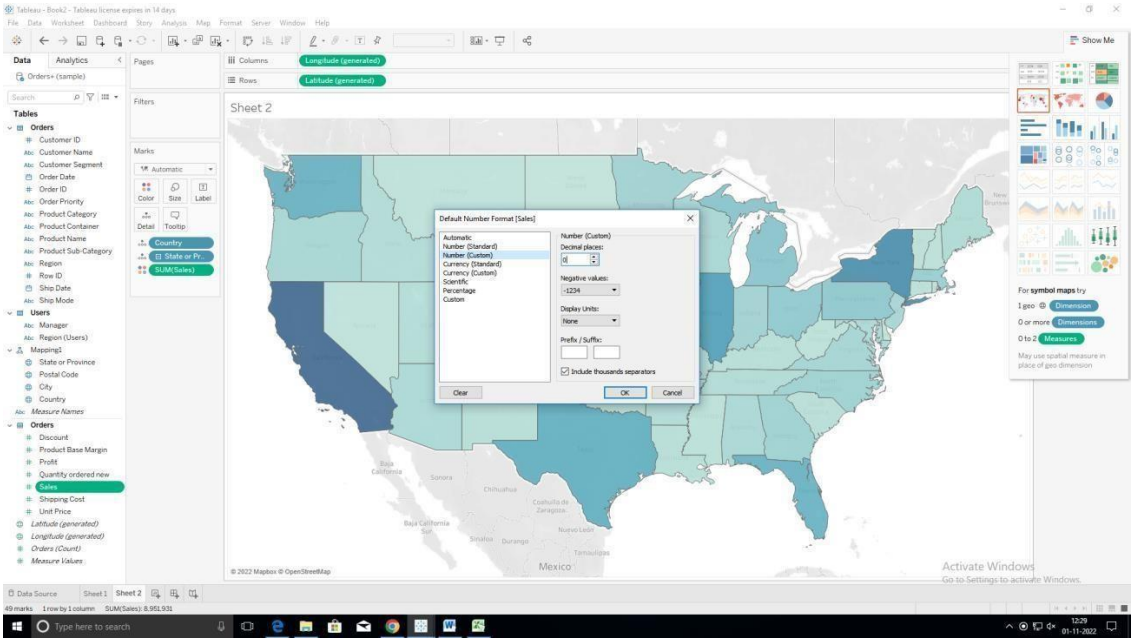
   Each state is labeled with sum of sales. The numbers need a little bit of formatting, however.

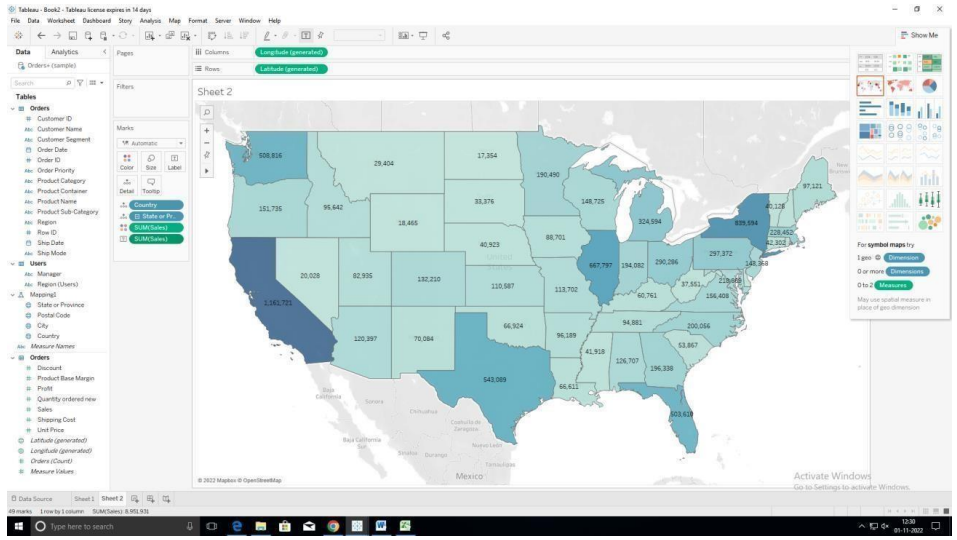
2. In the Data pane, right-click Sales and select Default Properties > Number Format.
3. In the Default Number Format dialog box that opens, select Number (Custom), and then do the following:
   - For Decimal Places, enter 0.
   - For Units, select Thousands (K).
   - Click OK.



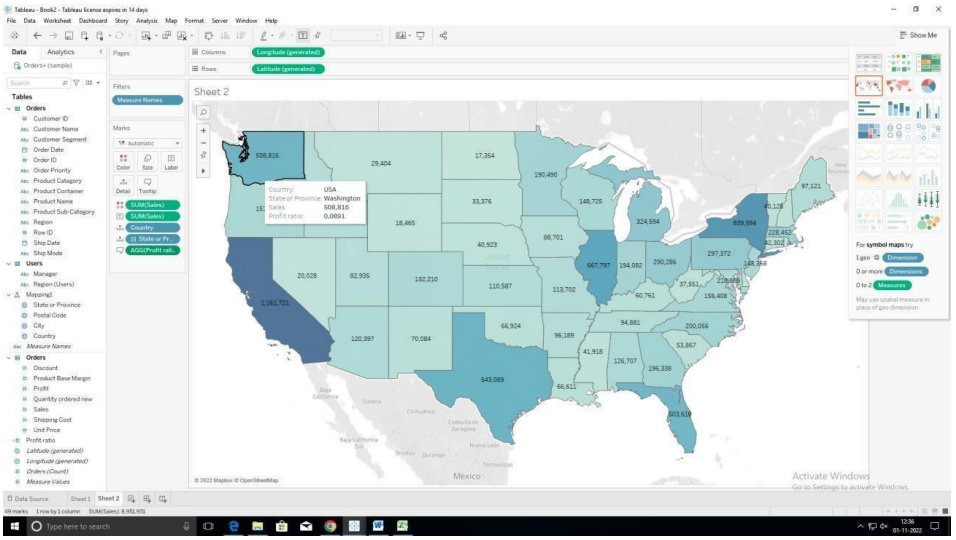

Sheet 2
Mexico
Activate Windows


Sheet 2
Default Number Format [Sales]
Mexico
Activate Windows

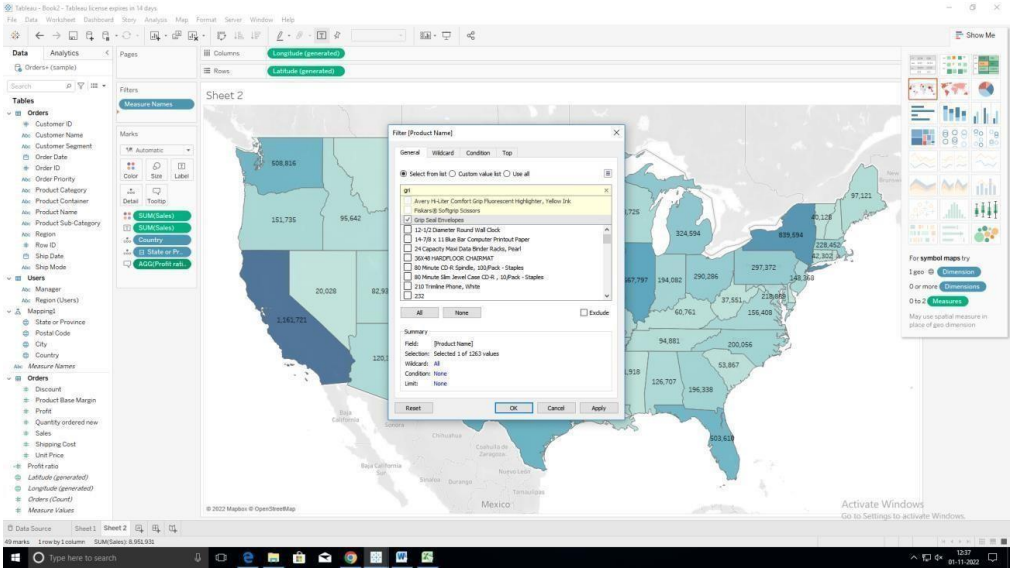

Sheet 2
Default Number Format [Sales]
Mexico
Activate Windows



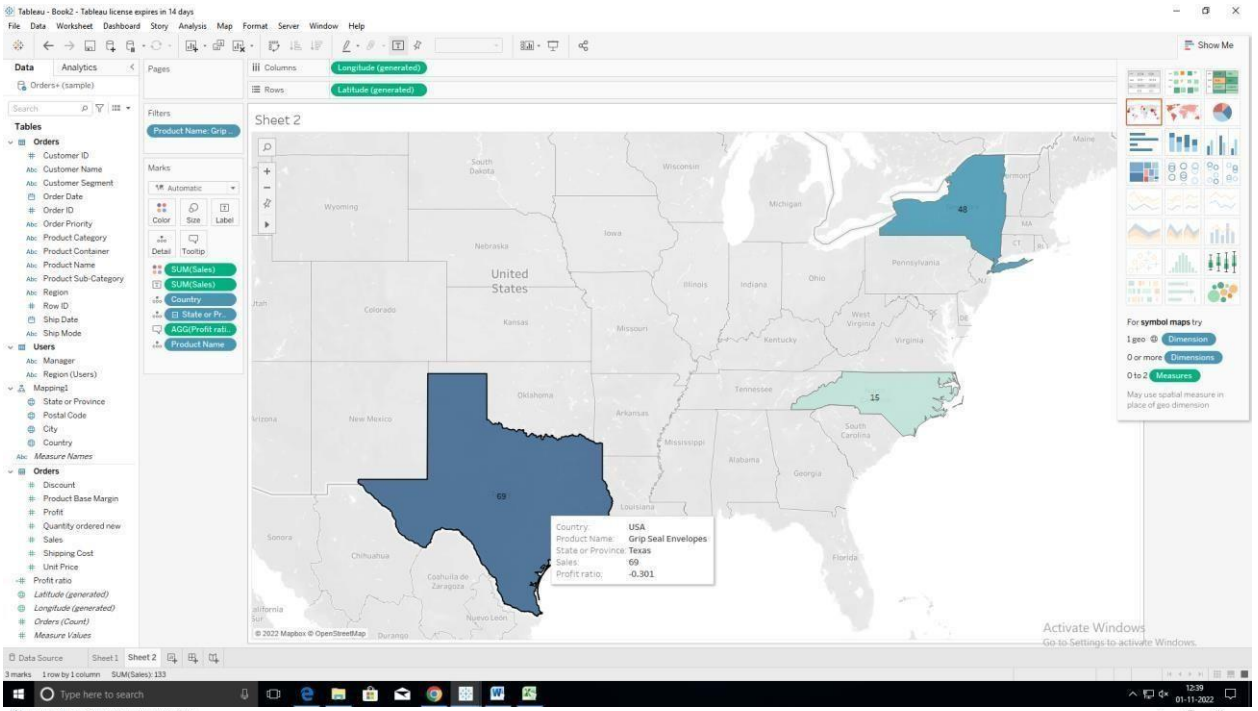
Q 2: Show Profit Ratio of each state as tooltip on map

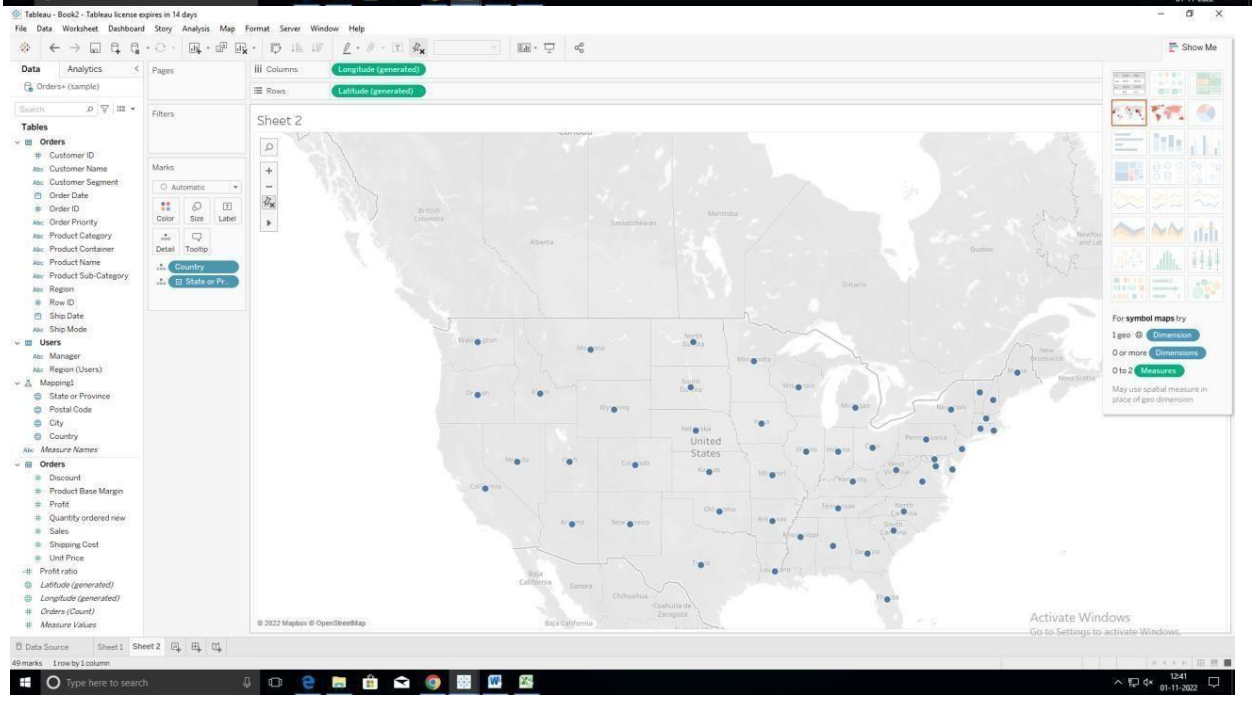


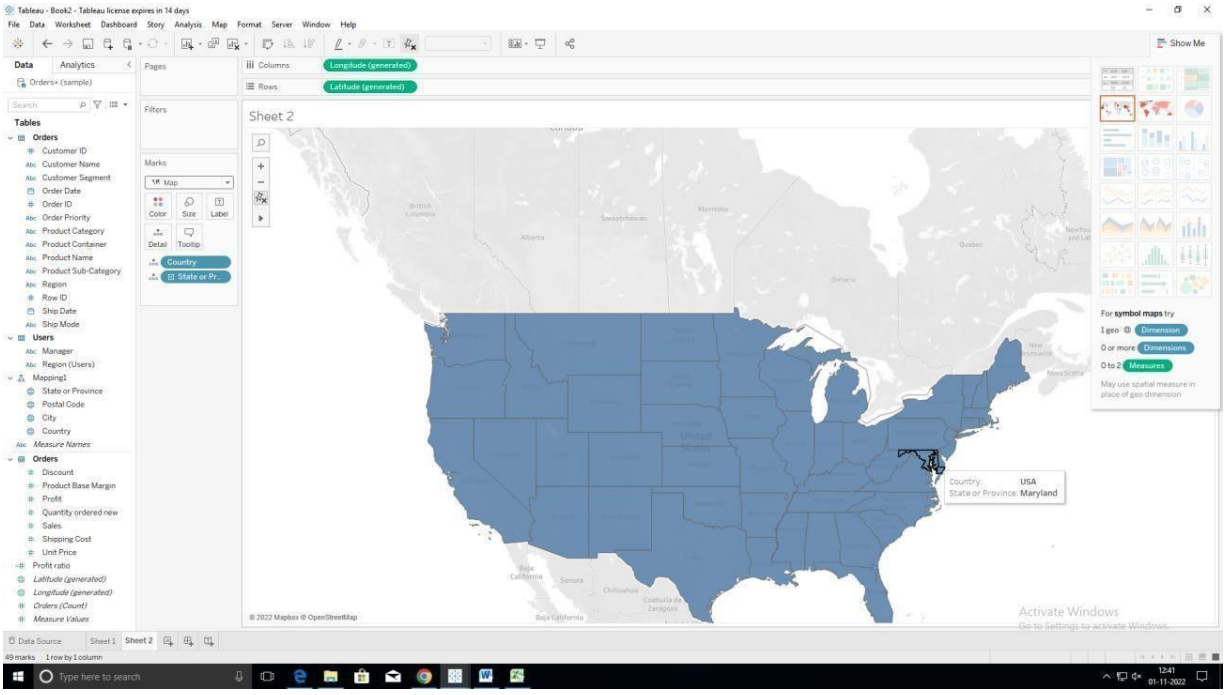
Q 3: Show Profit ratio for Grip Envelop products



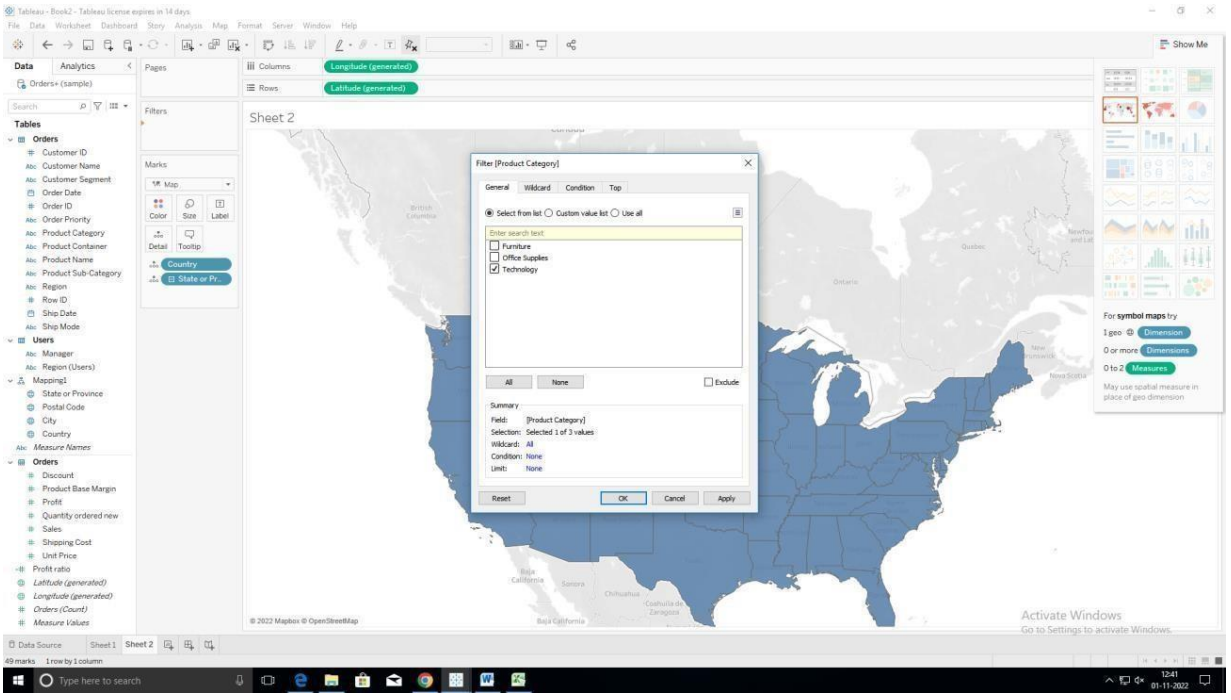
Q 4: In the technology product category which unprofitable state is surrounded by only profitable states.

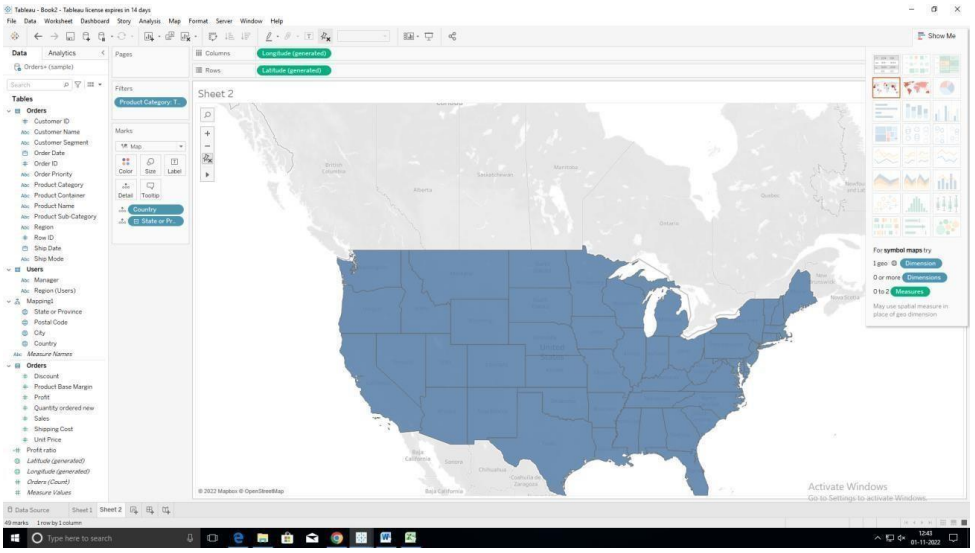




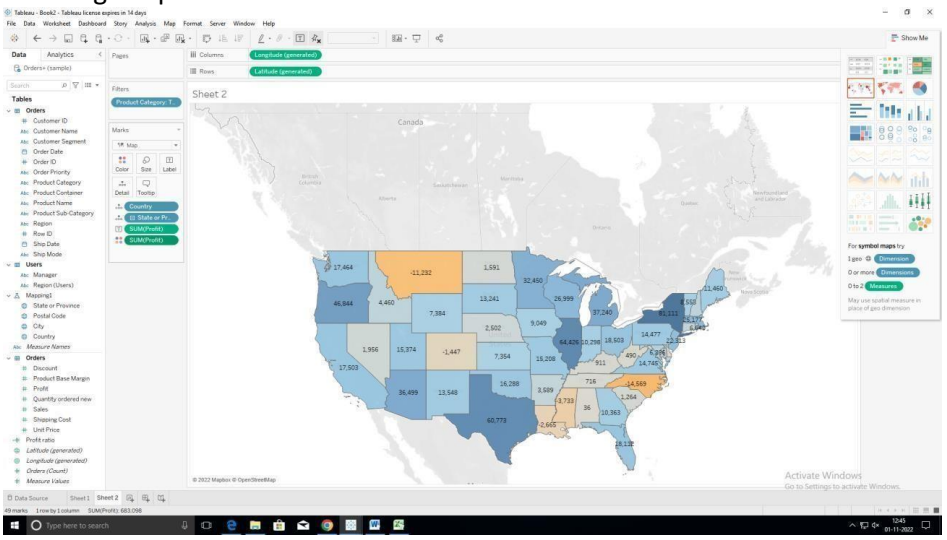


Drag the product category on the filter shelf and select in technology.





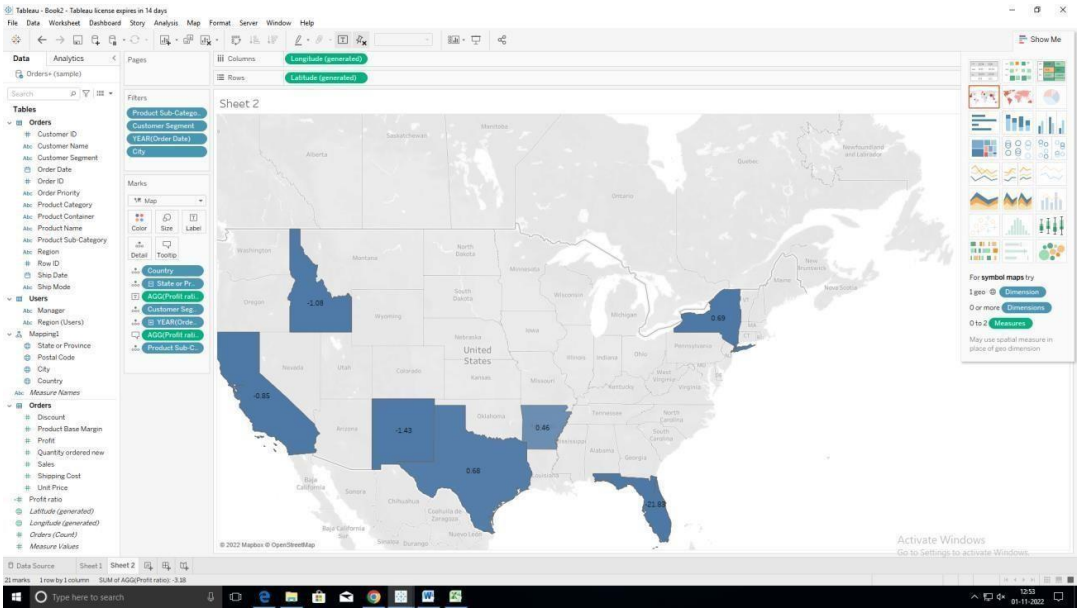
Now Drag the profit measure to color mark



Add profit Label.

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

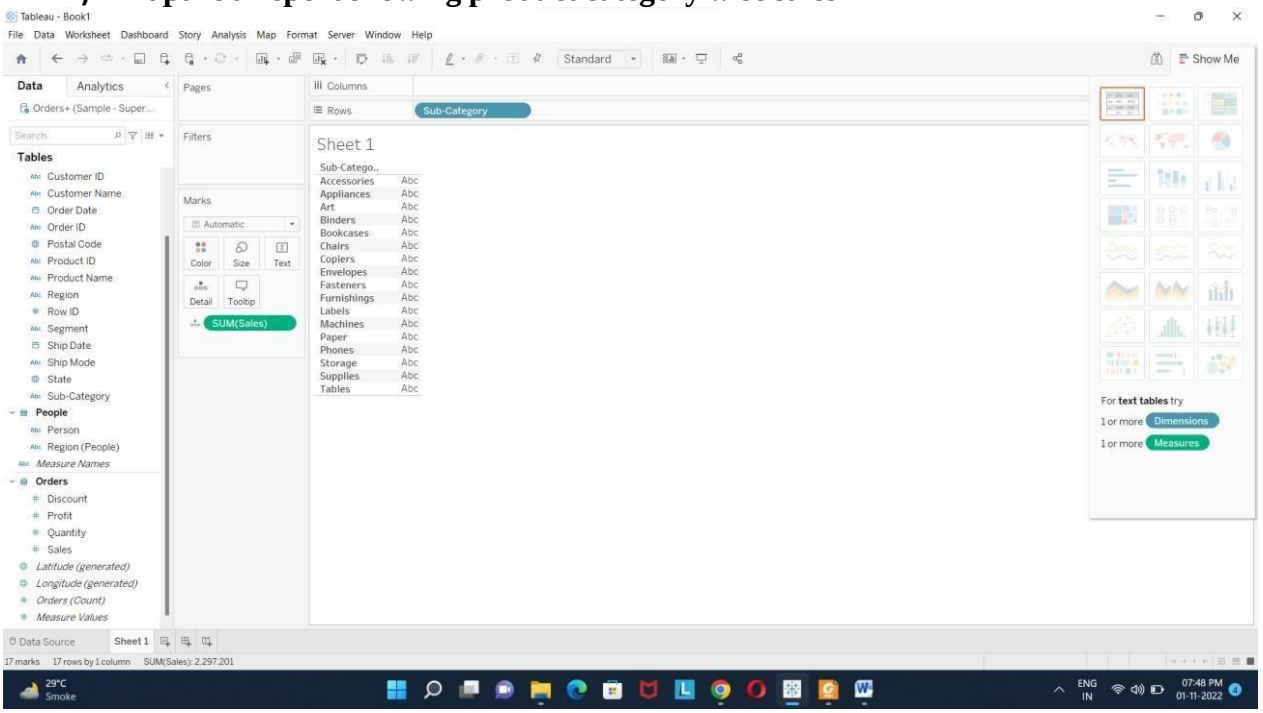
Q 5: Which state has the worst Gross Profit Ratio on Envelopes in the Corporate Customer Segment that were Shipped in 2015?



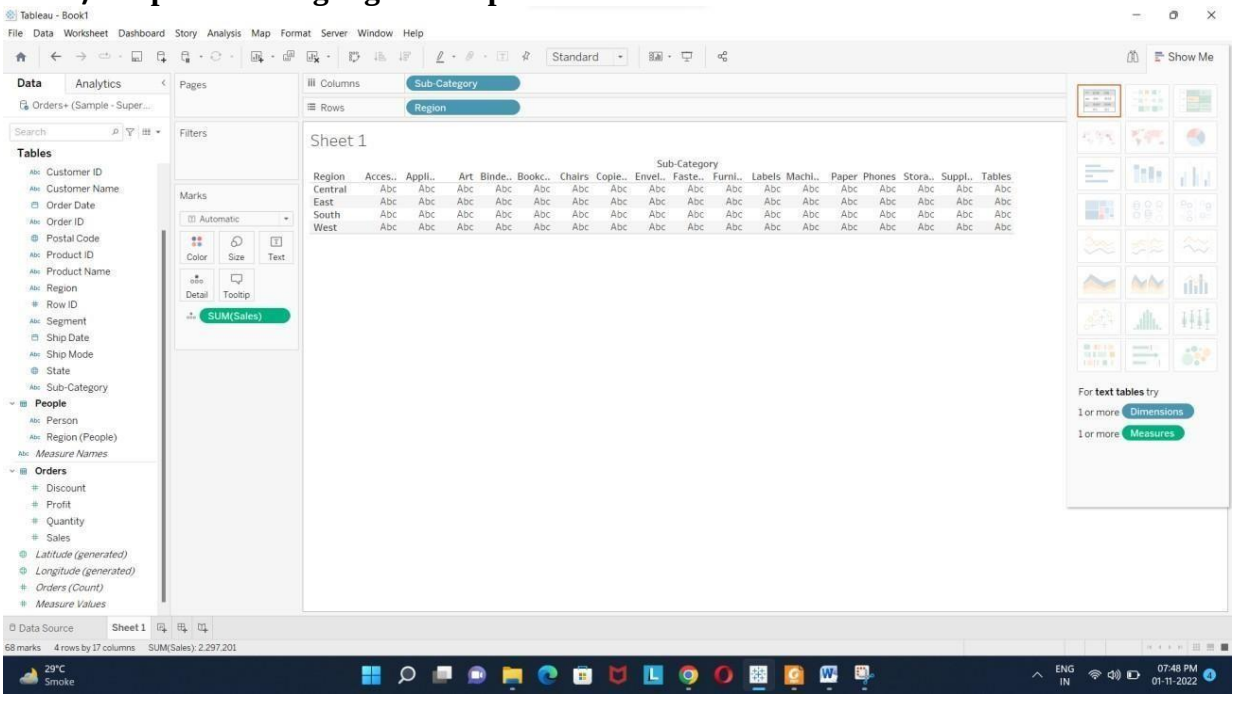
Assignment 3: Preparing Reports

Data Set: Super store

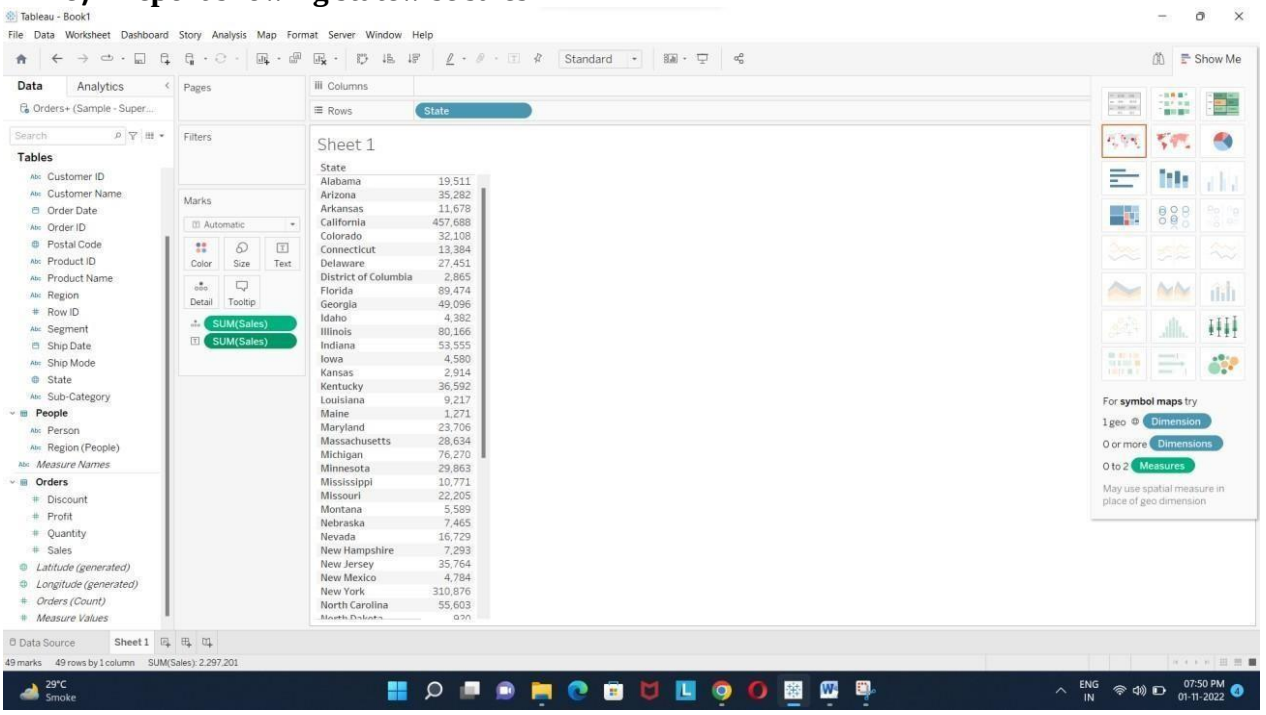
1) Prapare a report showing product category wise sales



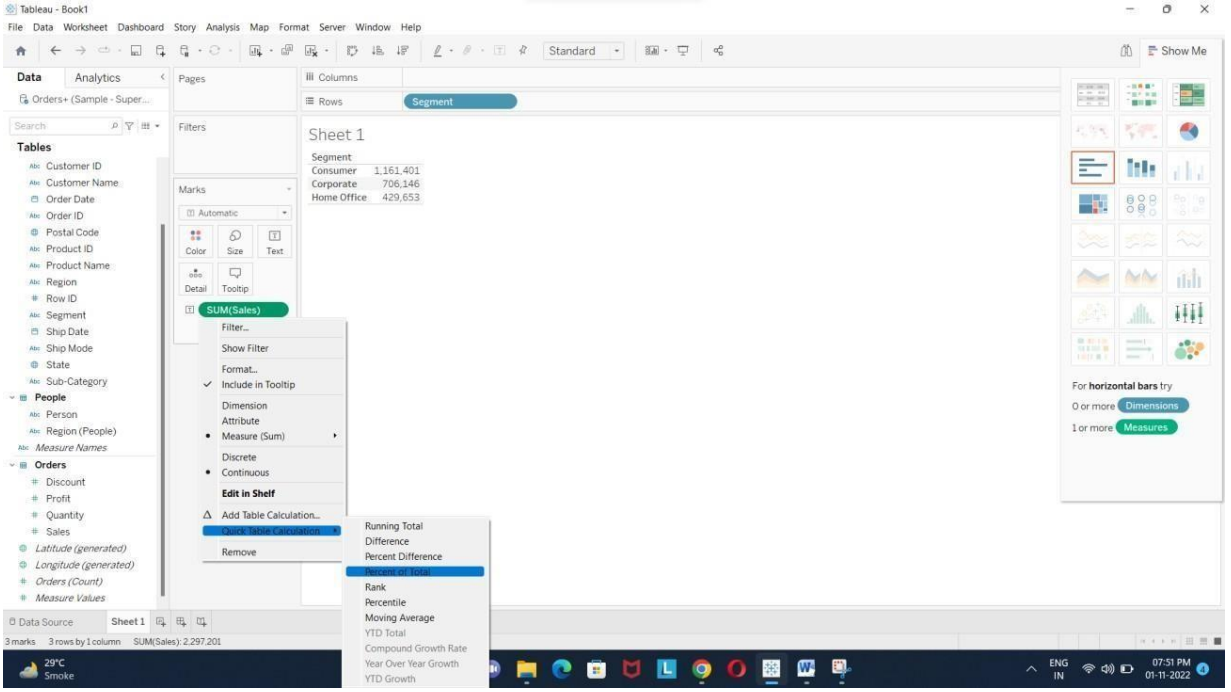
2) Report showing regionwise productwise sales

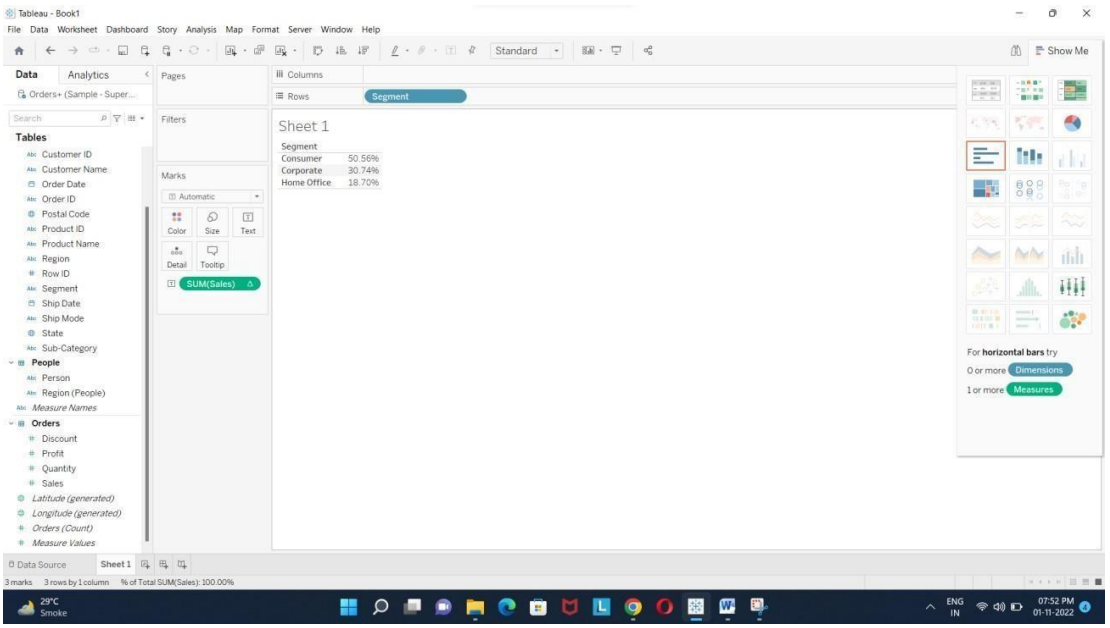


3) Report showing statewise sales



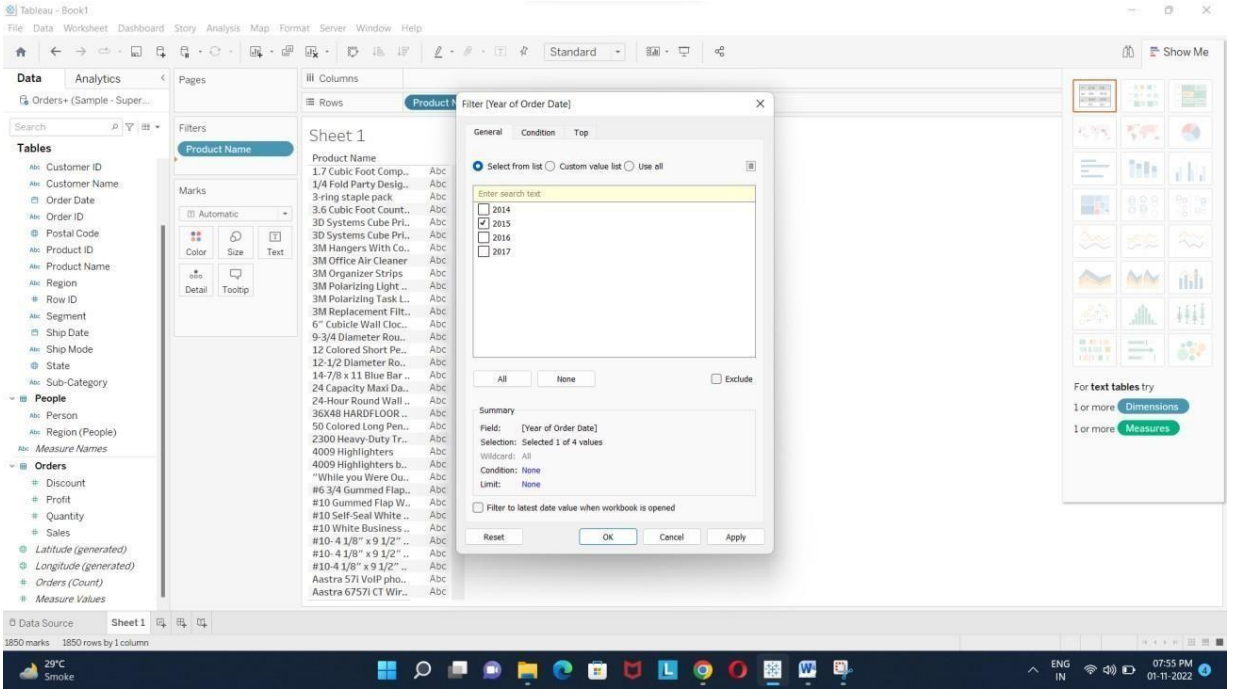
4) What is the percent of total Sales for the 'Home Office' Customer Segment in Julyof 2014?

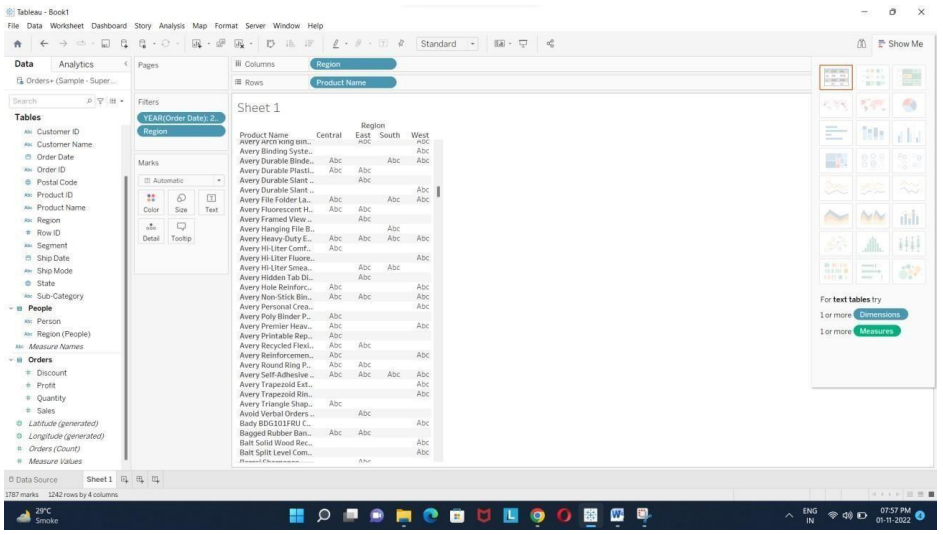




5) Find the top 10 Product Names by Sales within each region. Which product isranked #2 in both the Central & West regions in 2015?

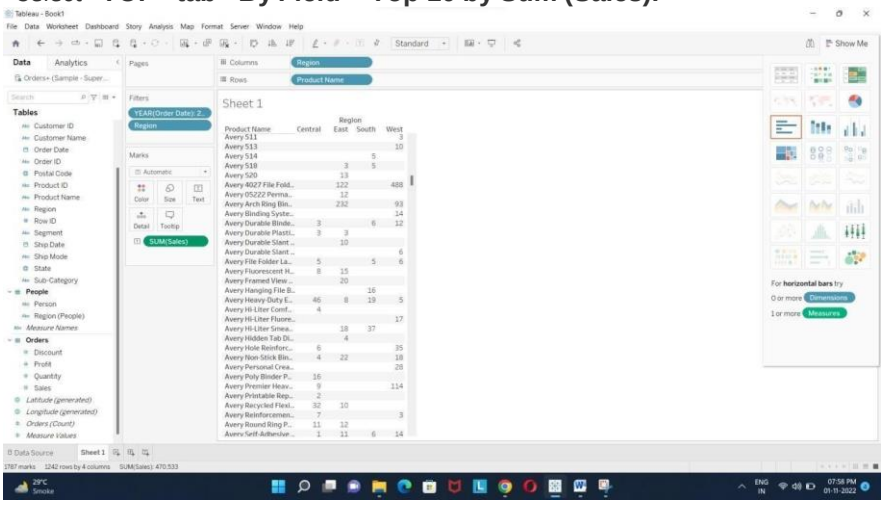
Drag "**Product Name**" dimension from data pane window to Row Shelf and then add an "order Date" on Filter shelf and select "Year" of Order date as **2015**.After that put region onFilter shelf and select "**Central**" and "**West**" checkbox. Also, put a copy of region to the Column shelf as well.

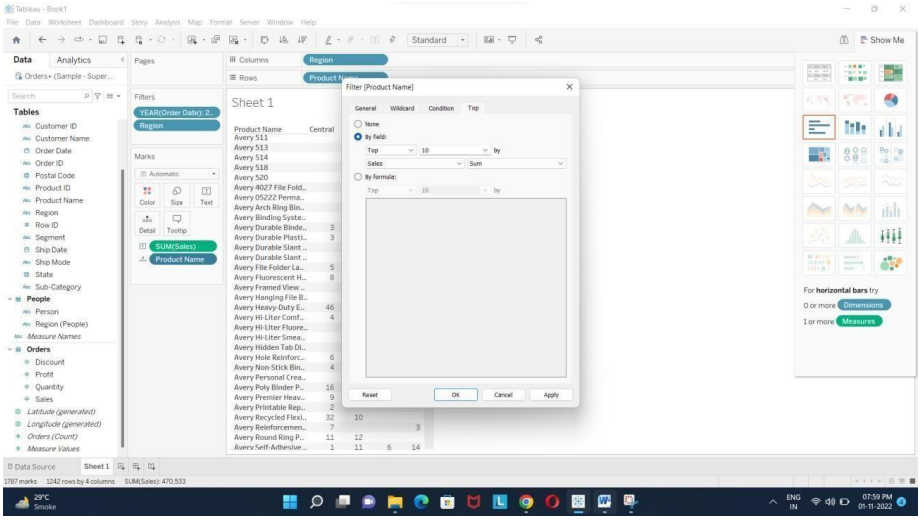


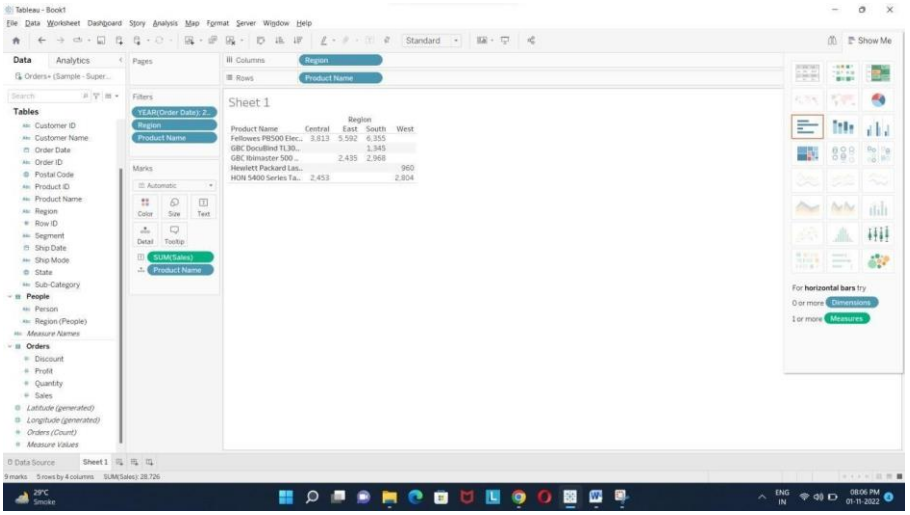


Drag a Sales measure to the text label. So for getting the Top 10 “product name” by sales, we need to add the “Product name” on Filter shelf. Once the Filter Pop up is open,

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







Calculation > Rank. As the default addressing is Table across, please change it intoTable Down(**Compute using -> Table Down**).