Big Data Analytics and Visualization Lab

Subject Code: MCAL31

A Practical Journal Submitted in Fulfilment of the Degree

of

MASTER

In

COMPUTER APPLICATION

Year 2024-2025

By

Mr. Agrawal Yash Gopal

(Application Id: - 53715)

Semester- III (CBCS)

Institute of Distance and Open Learning

Vidya Nagari, Kalina, Santacruz East – 400098.

University of Mumbai

PCP Centre

[Vidyavardhini's College of Technology – Vasai Road, Palghar 401202]

Institute of Distance and Open Learning,

Vidya Nagari, Kalina, Santacruz (E) -400098

CERTIFICATE

This to certify that, **Mr. Agrawal Yash Gopal** appearing **Master in Computer Application (Semester III - CBCS) Application ID: 53715** has satisfactorily completed the prescribed practical of **MCAL31 - Big Data Analytics and Visualization Lab** as laid down by the University of Mumbai for the academic year 2024-25.

Teacher in charge | Examiners | Coordinator IDOL, MCA
University of Mumbai

Date: - 08/01/2025

Place: - Vasai

Index

Practical 1

AIM: Installation, Configuration of Hadoop on Windows 11 and Executing HDFS Commands (Any 5)

Steps:

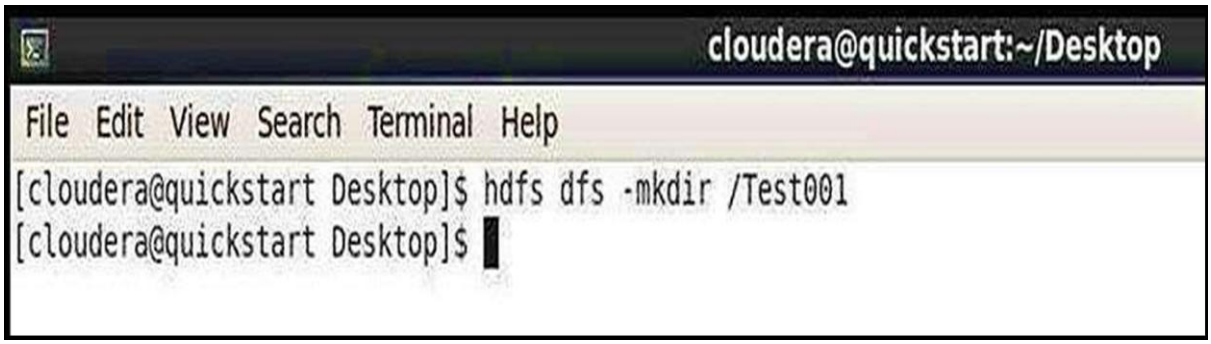
To install and configure Hadoop on Windows 11:

1. **Install Java**:
   - Download and install the latest JDK.
   - Set JAVA_HOME in system environment variables.
2. **Download Hadoop**:
   - Download the latest Hadoop binary from the Apache Hadoop website.
3. **Set Environment Variables**:
   - Set HADOOP_HOME to the Hadoop installation directory.
   - Add %HADOOP_HOME%\bin to the PATH environment variable.
4. **Configure Hadoop**:
   - In $HADOOP_HOME/etc/hadoop, edit core-site.xml, hdfs-site.xml, and mapred-site.xml with appropriate settings (use local filesystem).
5. **WinUtils**:
   - Download winutils.exe from GitHub and place it in the bin directory of Hadoop.
6. **Format the HDFS**:
   - Run hdfs namenode -format in the command prompt.
7. **Start Hadoop**:
   - Start Hadoop services (NameNode, DataNode, ResourceManager) using the following commands:
     sh
     Copy code
     start-dfs.cmd
     start-yarn.cmd
8. **Verify**:
   - Access the web UI at http://localhost:50070 for HDFS and http://localhost:8088 for YARN.

SOURCE CODE:

1. **mkdir**

   Command: hdfs dfs -mkdir



2. **touchz**

Command: hdfs dfs -touchz hello.txt

```
[cloudera@quickstart Desktop]$ hdfs dfs -touchz hello.txt
[cloudera@quickstart Desktop]$ hdfs dfs -ls
Found 2 items
-rw-r--r--   1 cloudera cloudera          0 2021-12-25 05:31 hello.txt
drwxr-xr-x   - cloudera cloudera          0 2021-12-18 10:24 mumbai
[cloudera@quickstart Desktop]$
```

3. **rmr**

Command: hdfs dfs –rm -r

```
[cloudera@quickstart ~]$ hdfs dfs -rm -r /Rmr
Deleted /Rmr
```

4. **stat**

Command: hdfs dfs -stat

```
[cloudera@quickstart ~]$ hdfs dfs -stat /Test001
2021-12-25 13:10:19
[cloudera@quickstart ~]$
```

5. **cp**

Command: hdfs dfs –cp

```
[cloudera@quickstart ~]$ hdfs dfs -cp /geek /mumbai
[cloudera@quickstart ~]$ hdfs dfs -ls /mumbai
Found 3 items
-rw-r--r--   1 cloudera supergroup         13 2021-12-13 07:45 /mumbai/Darshana.txt
-rw-r--r--   1 cloudera supergroup         47 2021-12-13 07:28 /mumbai/Rote.txt
drwxr-xr-x   - cloudera supergroup          0 2021-12-25 06:58 /mumbai/geek
```

Practical 2

AIM: WAP in Map Reduce for Word Count Operation

SOURCE CODE:

WordCountDriver.java

```
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.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

public class WordCountDriver {
   public static void main(String[] args) throws Exception {
      Job j1 = Job.getInstance(new Configuration());

      j1.setJarByClass(WordCountDriver.class);
      j1.setJobName("AverageWord    Count");
      FileInputFormat.addInputPath(j1, new Path(args[0]));
      FileOutputFormat.setOutputPath(j1, new j1.setMapperClass(WordCountMapper.class);
j1.setReducerClass(WordCountReducer.class); j1.setOutputKeyClass(Text.class);
j1.setOutputValueClass(IntWritable.class); System.exit(j1.waitForCompletion(true) ? 0 : 1); Path(args[1]));
   }
}
```

WordCountMapper.java

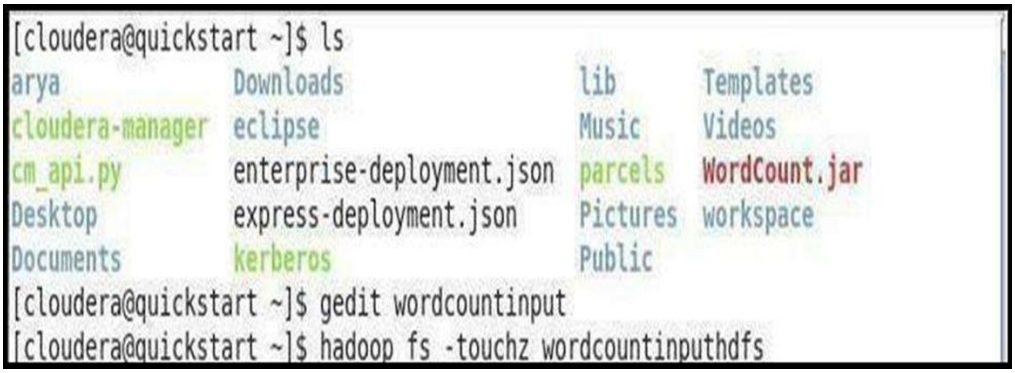
```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper < LongWritable, Text, Text, IntWritable > {
   private final static
   IntWritable one = new IntWritable(1);private Text word = new Text();@Override
   protected void map(LongWritable key, Text value, Mapper < LongWritable, Text, Text, IntWritable >
.Context context) throws IOException,
   InterruptedException {
      String line = value.toString();
      StringTokenizer itr = new StringTokenizer(line);
      while (itr.hasMoreTokens()) {
         word.set(itr.n extToken());
```

```
        context.write(word, one);
      }
    }
}
```

WordCountReducer.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer < Text, IntWritable, Text, IntWritable > {
   @Override
   protected void reduce(Text key, Iterable < IntWritable > values, Reducer < Text, IntWritable, Text,
IntWritable > .Context context) throws
   IOException,
   InterruptedException {
      int sum = 0;
      for (IntWritable value: values) {
         sum += value.get();
      }
      context.write(key, new IntWritable(sum));
   }
}
```

OUTPUT:



Practical 3

AIM: WAP in Map Reduce for Union and Intersection Operation

SOURCE CODE:

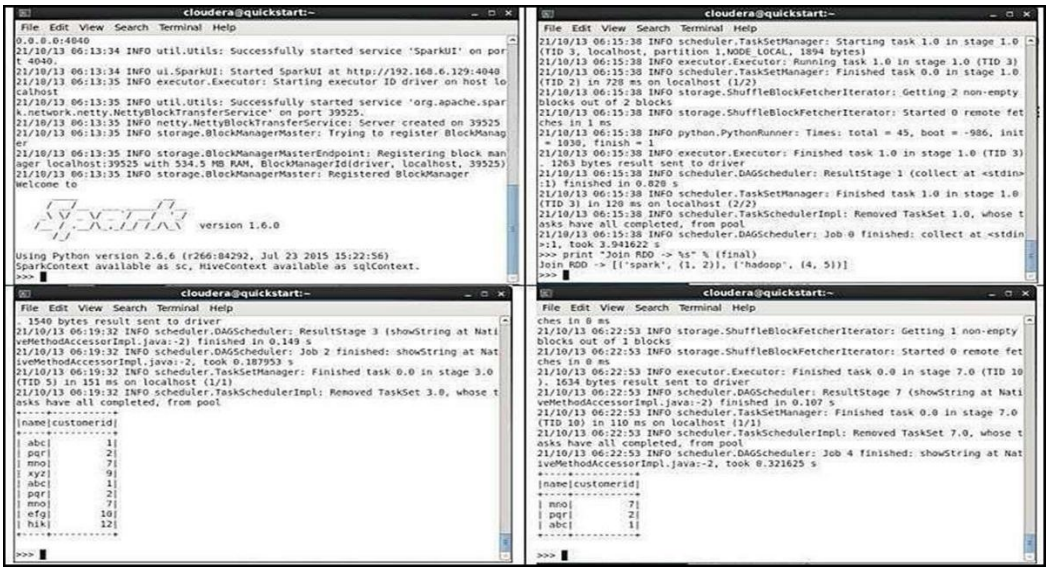
1. Union

```
>>>from pyspark import SparkContext
>>>sc = SparkContext()
>>>sqlContext = SQLContext(sc)
>>> valuesB = [('abc',1),('pqr',2),('mno',7),('xyz',9)]
>>> TableB = sqlContext.createDataFrame(valuesB,['name','customerid'])
>>> valuesC = [('abc',1),('pqr',2),('mno',7),('efg',10),('hik',12)]
>>> TableC = sqlContext.createDataFrame(valuesC,['name','customerid']
>> result= TableB.unionAll(TableC) >>> result.show()
>>>s =sc.parallelize([1,2,3,4,5,6])
>>>r = sc.parallelize([4,5,6,7,8,9,10])
>>>uni = s.union(r)
>>>uni.collect()
```

2. Intersection

```
>>>from pyspark import SparkContext
>>>sc = SparkContext()
>>>sqlContext = SQLContext(sc)
>>> valuesB = [('abc',1),('pqr',2),('mno',7),('xyz',9)]
>>> TableB = sqlContext.createDataFrame(valuesB,['name','customerid'])
>>> valuesC = [('abc',1),('pqr',2),('mno',7),('efg',10),('hik',12)]
>>> TableC = sqlContext.createDataFrame(valuesC,['name','customerid']
>> result= TableB.intersect(TableC)
>>> result.show()
```

OUTPUT:



Practical 4

AIM: WAP in Map Reduce for Matrix Multiplication

Source Code:
MatrixMultiplicationMapper.java

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MatrixMultiplicationMapper extends Mapper<Object, Text, Text, Text> {
   private Text outputKey = new Text();
   private Text outputValue = new Text();

   @Override
   protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
      String[] parts = value.toString().split(",");
      String matrixName = parts[0];
      int row = Integer.parseInt(parts[1]);
      int col = Integer.parseInt(parts[2]);
      String val = parts[3];

      if (matrixName.equals("A")) {
         for (int k = 0; k < context.getConfiguration().getInt("numColumnsB", 0); k++) {
            outputKey.set(row + "," + k);
            outputValue.set("A," + col + "," + val);
            context.write(outputKey, outputValue);
         }
      } else if (matrixName.equals("B")) {
         for (int i = 0; i < context.getConfiguration().getInt("numRowsA", 0); i++) {
            outputKey.set(i + "," + col);
            outputValue.set("B," + row + "," + val);
            context.write(outputKey, outputValue);
         }
      }
   }
}
```

MatrixMultiplicationReducer.java

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
```

```
import java.util.HashMap;

public class MatrixMultiplicationReducer extends Reducer<Text, Text, Text, Text> {
   private Text result = new Text();

   @Override
   protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
      HashMap<Integer, Double> mapA = new HashMap<>();
      HashMap<Integer, Double> mapB = new HashMap<>();

      for (Text value : values) {
         String[] parts = value.toString().split(",");
         String matrixName = parts[0];
         int index = Integer.parseInt(parts[1]);
         double val = Double.parseDouble(parts[2]);

         if (matrixName.equals("A")) {
            mapA.put(index, val);
         } else if (matrixName.equals("B")) {
            mapB.put(index, val);
         }
      }

      double sum = 0.0;
      for (int k : mapA.keySet()) {
         if (mapB.containsKey(k)) {
            sum += mapA.get(k) * mapB.get(k);
         }
      }

      result.set(String.valueOf(sum));
      context.write(key, result);
   }
}
```

MatrixMultiplicationDriver.java

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MatrixMultiplicationDriver {
   public static void main(String[] args) throws Exception {
```

```
        if (args.length != 4) {
            System.err.println("Usage: MatrixMultiplicationDriver <input path> <output path> <numRowsA>
<numColumnsB>");
            System.exit(-1);
        }

        Configuration conf = new Configuration();
        conf.setInt("numRowsA", Integer.parseInt(args[2]));
        conf.setInt("numColumnsB", Integer.parseInt(args[3]));

        Job job = Job.getInstance(conf, "Matrix Multiplication");
        job.setJarByClass(MatrixMultiplicationDriver.class);
        job.setMapperClass(MatrixMultiplicationMapper.class);
        job.setReducerClass(MatrixMultiplicationReducer.class);

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

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

OUTPUT:

```
0,0     7.0
0,1     10.0
1,0     15.0
1,1     22.0
```

Practical 5

AIM: Create Sample Database using MongoDB

Theory:

Creating a sample database in MongoDB is straightforward due to its flexible schema model. MongoDB uses the concept of databases, collections (like tables in RDBMS), and documents (like rows in RDBMS).

Step 1: Open the MongoDB shell.

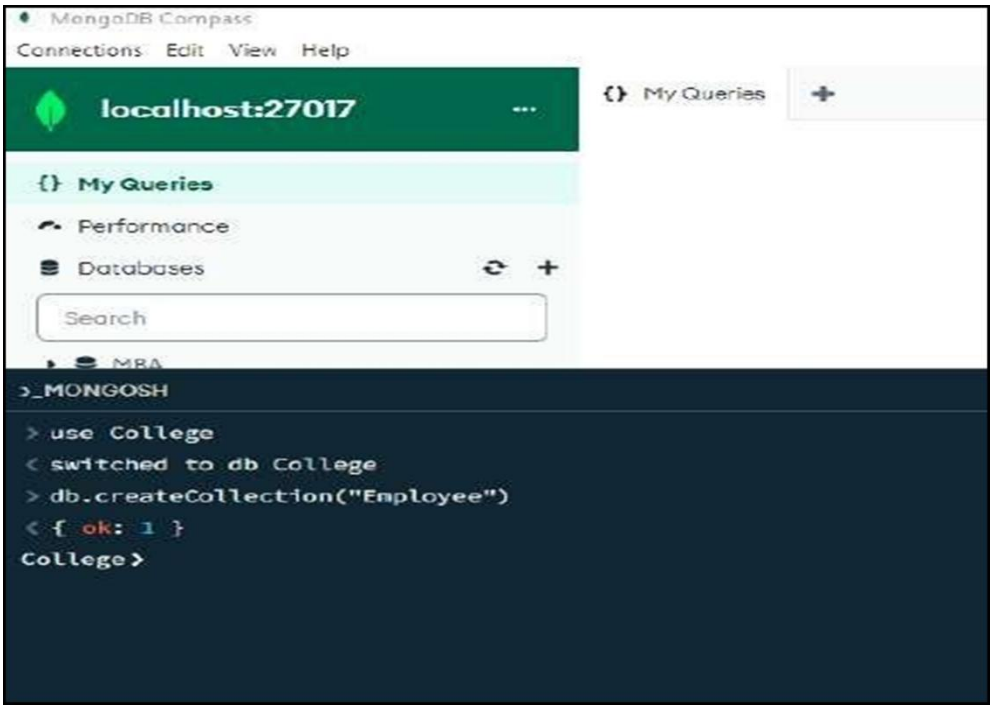
Step 2: Create a new database using the command:

Step 3: The database will be created and switched to SampleDB.

SOURCE CODE:

use College

OUTPUT:



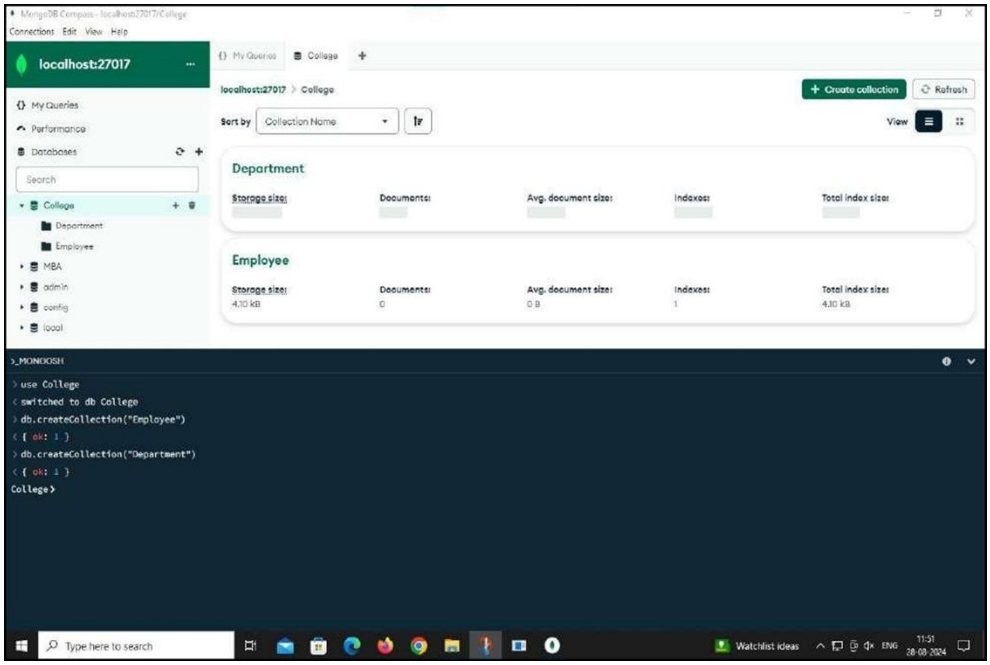
Practical 6

AIM: Query the Sample Database using MongoDB quering commands.

A. Create Collection

Source Code:

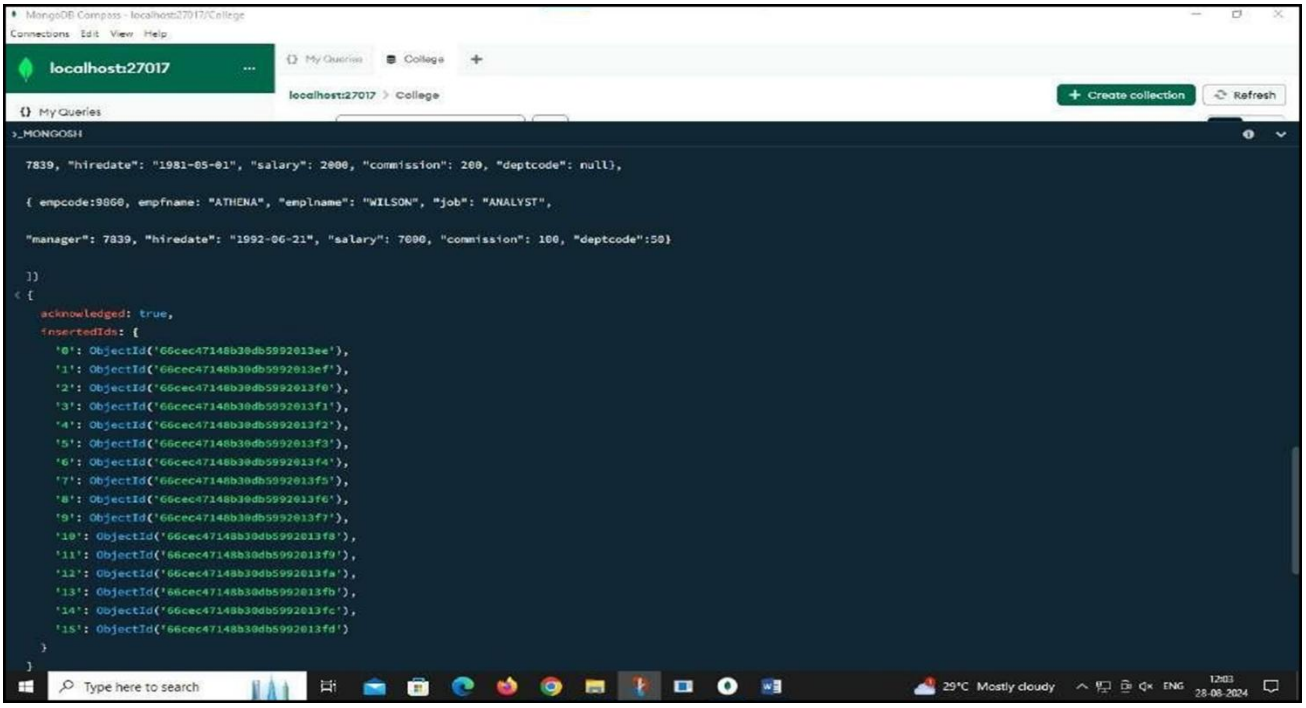
```
db.createCollection("Employee")
db.createCollection("Deparment")
```



B. Insert Document

Source Code:

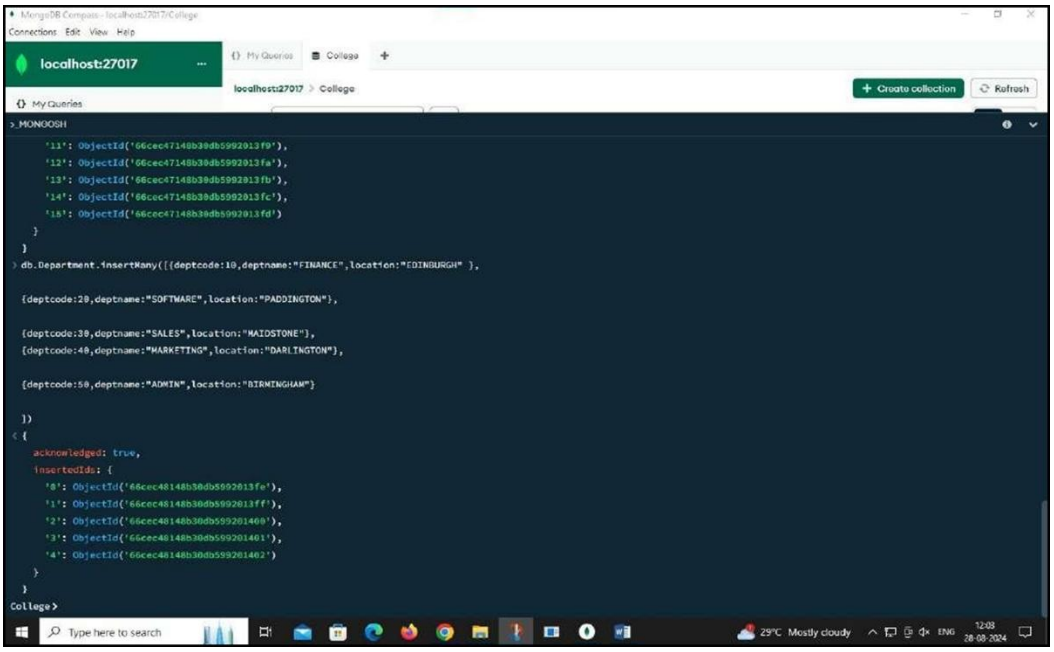
```
db.Employee.insertOne({
  name: "John Doe",
  age: 30,
  position: "Software Engineer",
  department_id: ObjectId("your_department_id_here"),  // Reference to Department collection
  salary: 75000
});
db.Employee.insertOne({
  name: "Alice Smith",
  age: 28,
  position: "Product Manager",
  department_id: dept.insertedId,  // Use the ID from the previous insert
  salary: 85000
});
```



C. Update Document

Source Code:

```
db.Employee.updateOne(
  { name: "John Doe" },
  {
    $set: {
      position: "Senior Software Engineer",
      salary: 80000
    }
  }
);
```



D. Delete Document

Source Code:

db.Employee.deleteMany({job:”SALESMAN”})

```
> db.Employee.deleteMany({job:"SALESMAN"})
< {
    acknowledged: true,
    deletedCount: 3
  }
> db.Employee.find()
< {
    _id: ObjectId('66cec47148b30db5992013ee'),
    empcode: 9369,
    empfname: 'TONY',
    emplname: 'STARK',
    job: 'SOFTWAREENGINEER',
    manager: 7902,
    hiredate: '1980-12-17',
    salary: 2800,
    commission: 0,
    deptcode: 20
  }
  {
    _id: ObjectId('66cec47148b30db5992013f0'),
    empcode: 9566,
    empfname: 'KIM',
    emplname: 'JARVIS',
    job: 'MANAGER',
    manager: 7839,
```

Practical 7

AIM: Create DB and table using Hive, execute basic HiveQL queries.

Source Code:

1. Creating Database

CREATE DATABASE SalesDB;

USE SalesDB;

```
hive> load data local inpath '/home/cloudera/student2.txt' into table sakshi50.student partition(course="hadoop");
Loading data to table sakshi50.student partition (course=hadoop)
Partition sakshi50.student{course=hadoop} stats: [numFiles=1, numRows=0, totalSize=65, rawDataSize=0]
OK
Time taken: 0.929 seconds
hive> load data local inpath '/home/cloudera/student1.txt' into table sakshi50.student partition(course="java");
Loading data to table sakshi50.student partition (course=java)
Partition sakshi50.student{course=java} stats: [numFiles=2, numRows=0, totalSize=314, rawDataSize=0]
OK
Time taken: 0.468 seconds
hive> select*from sakshi50.student;
OK
1       utkarsha        25      met     hadoop
2       omkar   24      met     hadoop
3       akshay  22      vjti    hadoop
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
Time taken: 0.376 seconds, Fetched: 17 row(s)
hive> select*from sakshi50.student where course="java";
OK
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
```

2. Creating Table

```
CREATE TABLE Customers (
   id INT,
   name STRING,
   age INT,
   city STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

```
hive> load data local inpath '/home/cloudera/student2.txt' into table sakshi50.student partition(course="hadoop");
Loading data to table sakshi50.student partition (course=hadoop)
Partition sakshi50.student{course=hadoop} stats: [numFiles=1, numRows=0, totalSize=65, rawDataSize=0]
OK
Time taken: 0.929 seconds
hive> load data local inpath '/home/cloudera/student1.txt' into table sakshi50.student partition(course="java");
Loading data to table sakshi50.student partition (course=java)
Partition sakshi50.student{course=java} stats: [numFiles=2, numRows=0, totalSize=314, rawDataSize=0]
OK
Time taken: 0.468 seconds
hive> select*from sakshi50.student;
OK
1       utkarsha        25      met     hadoop
2       omkar   24      met     hadoop
3       akshay  22      vjti    hadoop
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
Time taken: 0.376 seconds, Fetched: 17 row(s)
hive> select*from sakshi50.student where course="java";
OK
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
1       sakshi  25      nmitd   java
2       pooja   24      ruparel java
3       megha   22      tilak   java
NULL    NULL    NULL    NULL    java
1       utkarsha        26      met     java
2       omkar   27      met     java
3       akshay  28      vjti    java
```

3. HiveQL Queries

SELECT name FROM Customers WHERE city = 'Mumbai';

```
2024-10-23 22:51:19,935 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2024-10-23 22:51:19,937 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
2024-10-23 22:51:19,937 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.addr
2024-10-23 22:51:19,937 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2024-10-23 22:51:19,937 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2024-10-23 22:51:19,946 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2024-10-23 22:51:19,946 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(001,sakshi,kanase,21,723375287398,dombivli)
(002,pooja,agre,23,3728758732,kalyan)
(003,megha,patil,34,832787649,thane)
(004,utkarsha,jain,32,386873547,dadar)
(005,akshay,gunjal,21,6726489374,mumbai)
(006,komal,bhakre,33,7268237984,mulund)
(007,omkar,nayak,32,83246735,satara)
(008,sneha,jadhv,25,236875521,pune)
grunt> describe student_info;
student_info: {id: chararray,fname: chararray,lname: chararray,age: int,phone: chararray,city: chararray}
```

SELECT name, salary FROM Employees ORDER BY salary DESC;

```
2024-10-23 22:55:10,150 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher
2024-10-23 22:55:10,150 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecate
2024-10-23 22:55:10,150 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is depre
2024-10-23 22:55:10,150 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is dep
2024-10-23 22:55:10,150 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been in
2024-10-23 22:55:10,155 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to pr
2024-10-23 22:55:10,155 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input pat
(002,pooja,agre,23,3728758732,kalyan)
(003,megha,patil,34,832787649,thane)
(004,utkarsha,jain,32,386873547,dadar)
(006,komal,bhakre,33,7268237984,mulund)
(007,omkar,nayak,32,83246735,satara)
(008,sneha,jadhv,25,236875521,pune)
```

SELECT department, COUNT(*) FROM Employees GROUP BY department;

```
2024-10-23 22:58:27,955 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapR
2024-10-23 22:58:27,956 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
2024-10-23 22:58:27,956 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
2024-10-23 22:58:27,956 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation -
2024-10-23 22:58:27,957 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTup
2024-10-23 22:58:27,962 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFor
2024-10-23 22:58:27,962 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util
(002,pooja,23,kalyan)
(003,megha,34,thane)
(004,utkarsha,32,dadar)
(006,komal,33,mulund)
(007,omkar,32,satara)
(008,sneha,25,pune)
```

Practical 8

AIM: Perform Word Count in Apache Spark

Source Code:

```
from pyspark import SparkContext

# Step 1: Initialize Spark Context
sc = SparkContext("local", "WordCount")

# Step 2: Load Data from a Text File
text_file = sc.textFile("/path/to/textfile.txt")

# Step 3: Transformations for Word Count
word_counts = (
    text_file
    .flatMap(lambda line: line.lower().split())  # Convert to lowercase and split into words
    .filter(lambda word: word.isalpha())       # Filter out non-alphabetic characters
    .map(lambda word: (word, 1))              # Create (word, 1) pairs
    .reduceByKey(lambda a, b: a + b)          # Sum counts for each word
)

# Step 4: Collect and Print the Results
for word, count in word_counts.collect():
    print(f"{word}: {count}")

# Stop the Spark Context
sc.stop()
```

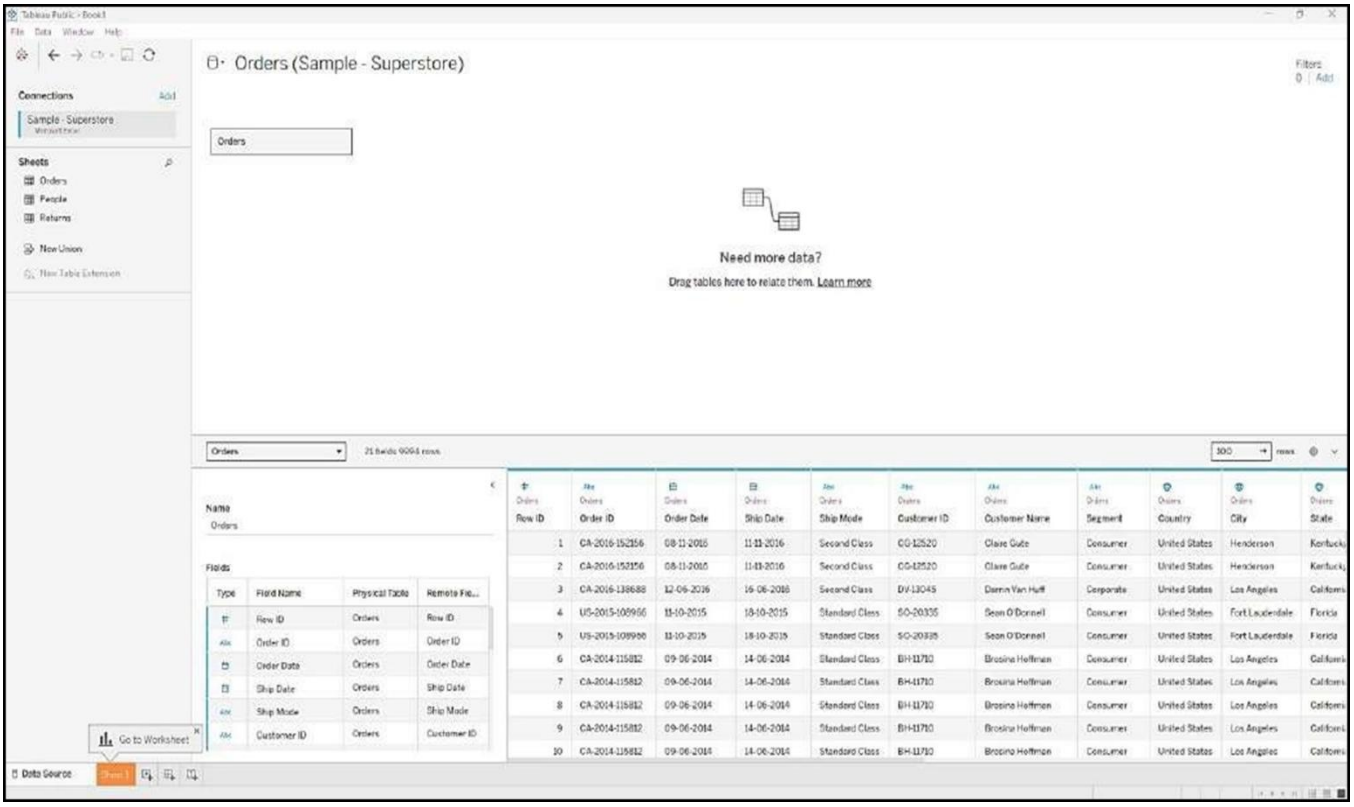
OUTPUT:

```
scala> val reducedata = mapdata.reduceByKey(_+_);
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[15] at reduceByKey at <console>:33

scala> reducedata.collect()
res14: Array[(String, Int)] = Array((d,1), (e,1), (M,1), (a,5), (t,2), (k,1), (u,1), (" ",2), ("",3), (o,2), (n,1), (!,1), (H,2), (r,1), (l,3), (W,1))

scala> val mapdata = RDD1.map(word =>(word,1));
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[16] at map at <console>:29

scala> mapdata.collect
res15: Array[(String, Int)] = Array((Hello World!,1), (Hakuna Matata,1), ("",1))

scala> val splitdata = RDD1.flatMap(line => line.split(" "));
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[17] at flatMap at <console>:29

scala> splitdata.collect
res16: Array[String] = Array(Hello, World!, Hakuna, Matata, "")
```

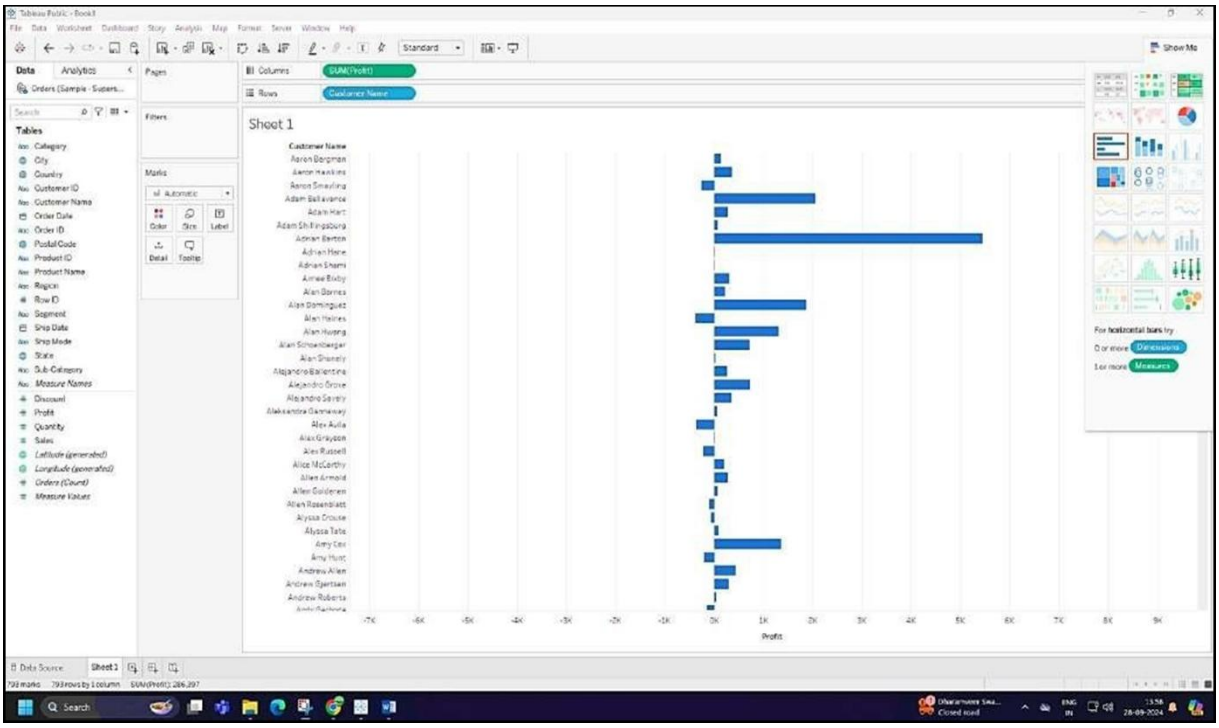
Practical 9

AIM: Data Visualization with Tableau: Import a Dataset and analyze the data with charts

IMPORTING DATASET:



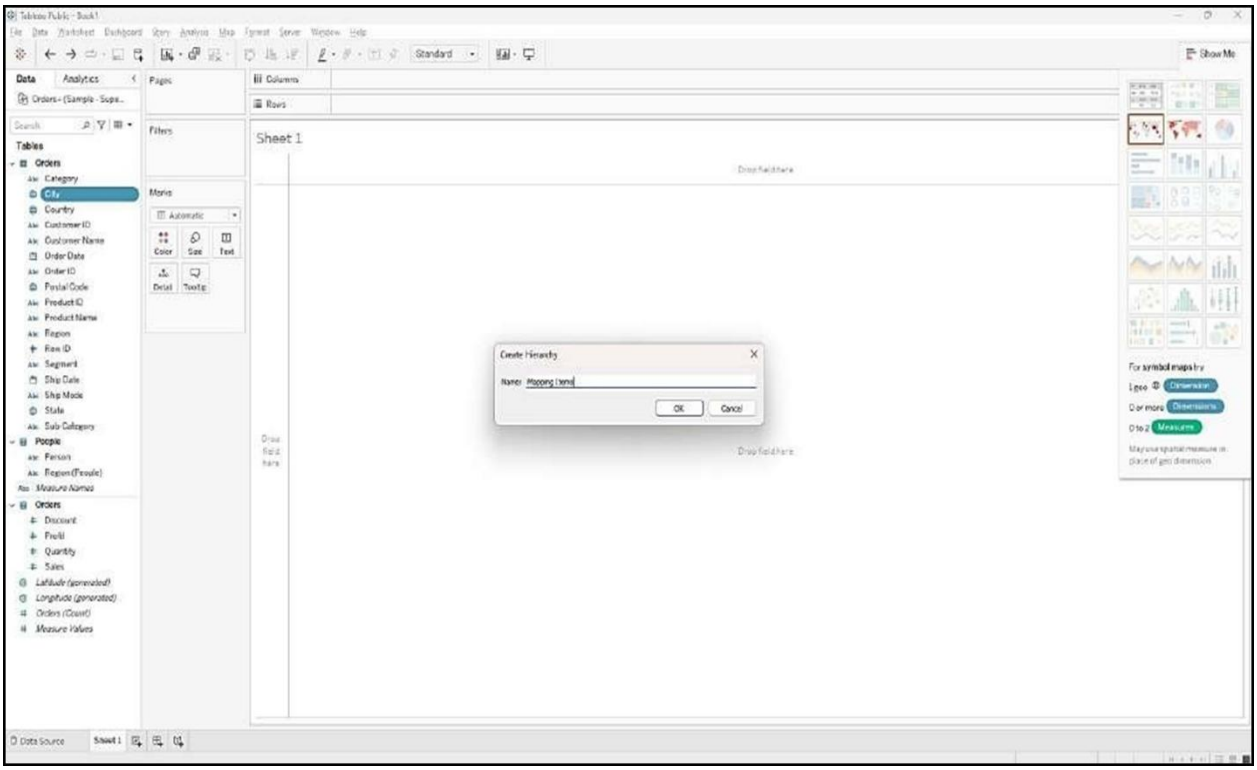
ANALYSING DATA WITH CHART:



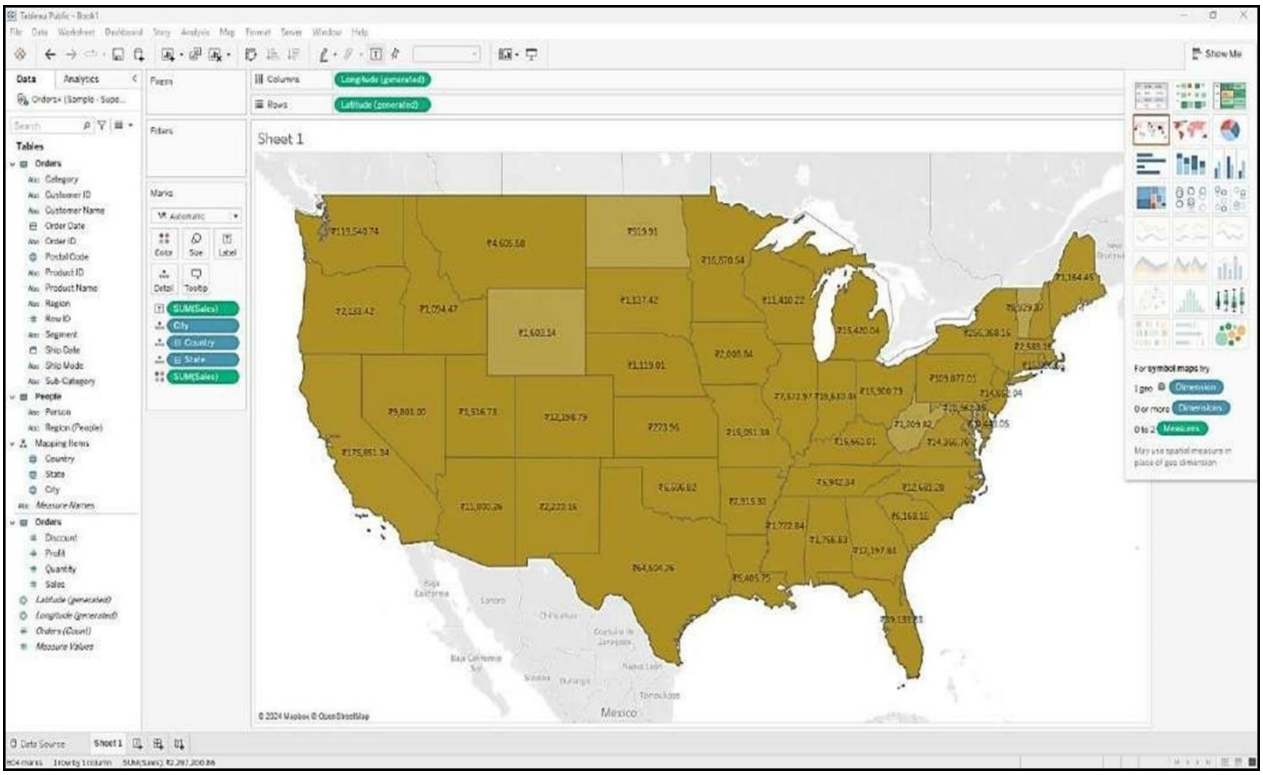
Practical 10

AIM: Create a Dashboards, Maps and Stories with Tableau

CREATING DASHBOARD:



CREATIMG MAPS:



CREATING STORIES: