QUESTION BANK FOR II Semester (Term: June-September 2024)

Cloud Computing and Big Data Laboratory (MCSL26/MCNL26)

Part - A

1. Write a MapReduce program using Java, to analyze the given Weather Report Data and to generate a report with cities having maximum and minimum temperature for a particular year.

driver.java

```
package weather;

import java.util.*;

import java.io.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.fs.Path;


public class driver

{

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

   {

         JobConf conf=new JobConf(driver.class);

         conf.setMapperClass(mapper.class);

         conf.setReducerClass(reducer.class);

         conf.setOutputKeyClass(Text.class);

         conf.setOutputValueClass(DoubleWritable.class);

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

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

         JobClient.runJob(conf);

   }

}
```

gmailmapper.java

```
package weather;

import java.util.*;

import java.io.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;


public class mapper extends MapReduceBase implements Mapper<LongWritable,
Text,Text,DoubleWritable>{

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

   {

         String line=value.toString();

         String year=line.substring(15,19);

         Double temp=Double.parseDouble(line.substring(87,92));

         output.collect(new Text(year), new DoubleWritable(temp));

   }

}
```

reducer.jav

```
package weather;

import java.util.*;

import java.io.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

class reducer extends MapReduceBase implements
Reducer<Text,DoubleWritable,Text,DoubleWritable> {

   public void reduce(Text key, Iterator<DoubleWritable> value,
OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException{

         Double max=-9999.0;

         Double min=9999.0;

         while(value.hasNext()){

               Double temp=value.next().get();

               max=Math.max(max,temp);
```

```
                min=Math.min(min,temp);
            }
            output.collect(new Text("Max temp at "+ key), new DoubleWritable(max));
            output.collect(new Text("Min temp at "+ key), new DoubleWritable(min));
    }
}
```

input.txt

0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9+00001+99999999999

0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999999

0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999999

0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999999

0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+00781+99999999999

Steps to run

1. Create a New File named Bash.sh
2. Copy the Below code and Paste inside Bash.sh and save that File.

```
        export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
        export PATH=$(echo $PATH):$(pwd)/bin
        export CLASSPATH=$(hadoop classpath)
```

3. Execute the bash.sh File using following command source Bash.sh.
4. Verify JAVA_HOME variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc file.
5. Verify Hadoop is Installed or not by executing hadoop command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files (driver.java mapper.java reducer.java)

```
        javac -d . *.java
```

9. Set driver class in manifest

```
        echo Main-Class: weather.driver > Manifest.txt
```

10. Create an executable jar file

```
        jar cfm weather.jar Manifest.txt weather/*.class
```

11. input.txt is input file for Weather create Input File

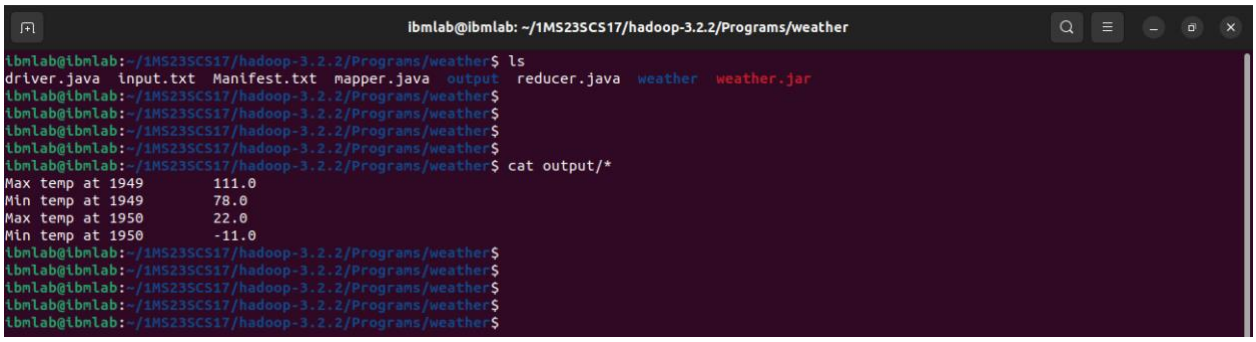
12. Run the jar file

hadoop jar weather.jar input.txt output

13. To see the Output

cat output/*

Output Screenshot



2. Write a MapReduce program using Java, to analyze the given Earthquake Data and generate statistics with region and magnitude/ region and depth/ region and latitude/ region and longitude

driver.java

```
package earthquake;

import java.util.*;

import java.io.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.fs.Path;


public class driver

{

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

   {

         JobConf conf=new JobConf(driver.class);

         conf.setMapperClass(mapper.class);

         conf.setReducerClass(reducer.class);

         conf.setOutputKeyClass(Text.class);

         conf.setOutputValueClass(DoubleWritable.class);

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

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

         JobClient.runJob(conf);

   }
```

mapper.java

```
package earthquake;

import java.util.*;

import java.io.*;

import org.apache.hadoop.mapred.*;

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

```
public class mapper extends MapReduceBase implements Mapper<LongWritable,
Text,Text,DoubleWritable>

{

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

   {

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

         Double longi=Double.parseDouble(line[7]);

         output.collect(new Text(line[11]), new DoubleWritable(longi));

   }

}
```

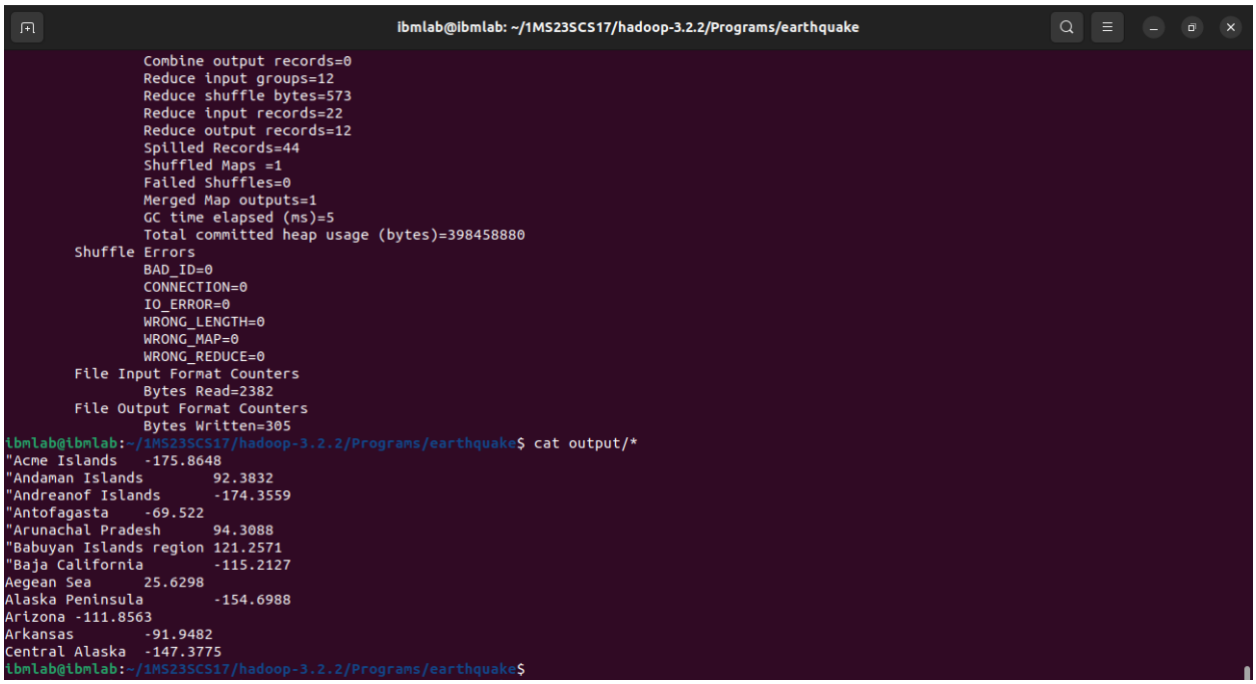
reducer.java

```
package earthquake;

import java.util.*;

import java.io.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

class reducer extends MapReduceBase implements
Reducer<Text,DoubleWritable,Text,DoubleWritable> {


   public void reduce(Text key, Iterator<DoubleWritable> value,
OutputCollector<Text,DoubleWritable> output, Reporter r) throws IOException

   {

         Double max=-9999.0;

         while(value.hasNext())

         {

               Double temp=value.next().get();

               max=Math.max(max,temp);

         }

         output.collect(new Text(key), new DoubleWritable(max));

   }


}
```

Steps to run

1. Create a New File named Bash.sh
2. Copy the Below code and Paste inside Bash.sh and save that File.
    export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
    export PATH=$(echo $PATH):$(pwd)/bin
    export CLASSPATH=$(hadoop classpath)
3. Execute the bash.sh File using following command source Bash.sh.
4. Verify JAVA_HOME variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc file.
5. Verify Hadoop is Installed or not by executing hadoop command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files (driver.java mapper.java reducer.java)
    javac -d . *.java
9. Set driver class in manifest
    echo Main-Class: earthquake.driver > Manifest.txt
10. Create an executable jar file
    jar cfm earthquake.jar Manifest.txt earthquake/*.class
11. input.csv is input file for earthquake create Input File
12. Run the jar file
    hadoop jar earthquake.jar input.csv output
13. To see the Output
    cat output/*

Output Screenshots



3. Write a MapReduce program using Java, to analyze the given natural numbers and generate statistics for the number as Odd or Even and print their sum.

driver.java

```
package oddeven;

import java.io.*;

import java.util.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.fs.Path;


public class driver

{

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

   {

          JobConf conf=new JobConf(driver.class);

          conf.setMapperClass(mapper.class);

          conf.setReducerClass(reducer.class);

          conf.setOutputKeyClass(Text.class);

          conf.setOutputValueClass(IntWritable.class);

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

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

          JobClient.runJob(conf);

   }

}
```

mapper.java

```
package oddeven;

import java.io.*;

import java.util.*;

import org.apache.hadoop.mapred.*;

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

```
public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , IntWritable>
{
   public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable> output,Reporter r) throws IOException
   {
         String[] line=value.toString().split(" ");
         for(String num:line){
               int number=Integer.parseInt(num);
               if(number%2==0) {
                     output.collect(new Text("even"),new IntWritable(number));
               }
               else{
                     output.collect(new Text("odd"),new IntWritable(number));
               }
         }
   }
}
```

reducer.java

```
package oddeven;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
public class reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>
{
   public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable> output ,Reporter r) throws IOException
   {
         int sum=0,count=0;
         while(value.hasNext()){
               sum+=value.next().get();
```

```
                count++;
            }
            output.collect(new Text("Sum of "+key+" Numbers"),new IntWritable(sum));
            output.collect(new Text(key+" Number count"),new IntWritable(count));
    }
}
```

input.txt

1 2 3 4 5 6 7 8 9 10

Steps to run

1. Create a New File named Bash.sh
2. Copy the Below code and Paste inside Bash.sh and save that File.

```
export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
export PATH=$(echo $PATH):$(pwd)/bin
export CLASSPATH=$(hadoop classpath)
```

3. Execute the bash.sh File using following command source Bash.sh.
4. Verify JAVA_HOME variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc file.
5. Verify Hadoop is Installed or not by executing hadoop command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files (driver.java mapper.java reducer.java)

```
javac -d . *.javagmail
```

9. Set driver class in manifest

```
echo Main-Class: oddeven.driver > Manifest.txt
```

10. Create an executable jar file

```
jar cfm oddeven.jar Manifest.txt oddeven/*.class
```

11. oe.txt is input file for Oddeven create Input File

```
echo 1 2 3 4 5 6 7 8 9 10 > input.txt
```

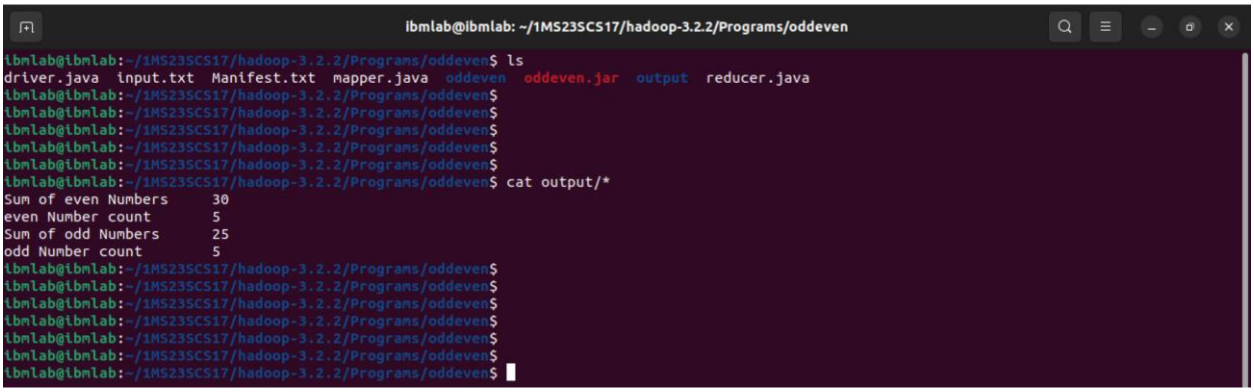
12. Run the jar file

```
hadoop jar oddeven.jar input.txt output
```

13. To see the Output

```
cat output/*
```

Output Screenshot



4. Write a MapReduce program using Java, to analyze the given Insurance Data and generate a statistics report with the construction building name and the count of building/ county name and its frequency.

driver.java

```
package insurance;

import java.io.*;

import java.util.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.fs.Path;


public class driver

{

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

   {

          JobConf conf=new JobConf(driver.class);

          conf.setMapperClass(mapper.class);

          conf.setReducerClass(reducer.class);

          conf.setOutputKeyClass(Text.class);

          conf.setOutputValueClass(IntWritable.class);

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

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

          JobClient.runJob(conf);

   }

}
```

mapper.java

```
package insurance;

import java.io.*;gmail

import java.util.*;

import org.apache.hadoop.mapred.*;

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

```
public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , IntWritable>

{

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

   {

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

         output.collect(new Text(line[2]),new IntWritable(1));

   }

}
```

reducer.java

```
package insurance;

import java.io.*;

import java.util.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

public class reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>

{

   public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable> output ,Reporter r) throws IOException

   {

         int sum=0;

         while(value.hasNext())

         {

               sum+=value.next().get();

         }

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

  }

}
```

Steps to run

1. Create a New File named Bash.sh
2. Copy the Below code and Paste inside Bash.sh and save that File.

```
export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
export PATH=$(echo $PATH):$(pwd)/bin
export CLASSPATH=$(hadoop classpath)
```

3. Execute the bash.sh File using following command source Bash.sh.
4. Verify JAVA_HOME variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc file.
5. Verify Hadoop is Installed or not by executing hadoop command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files (driver.java mapper.java reducer.java)

```
javac -d . *.java
```

9. Set driver class in manifest

```
echo Main-Class: insurance.driver > Manifest.txt
```

10. Create an executable jar file

```
jar cfm insurance.jar Manifest.txt insurance/*.class
```

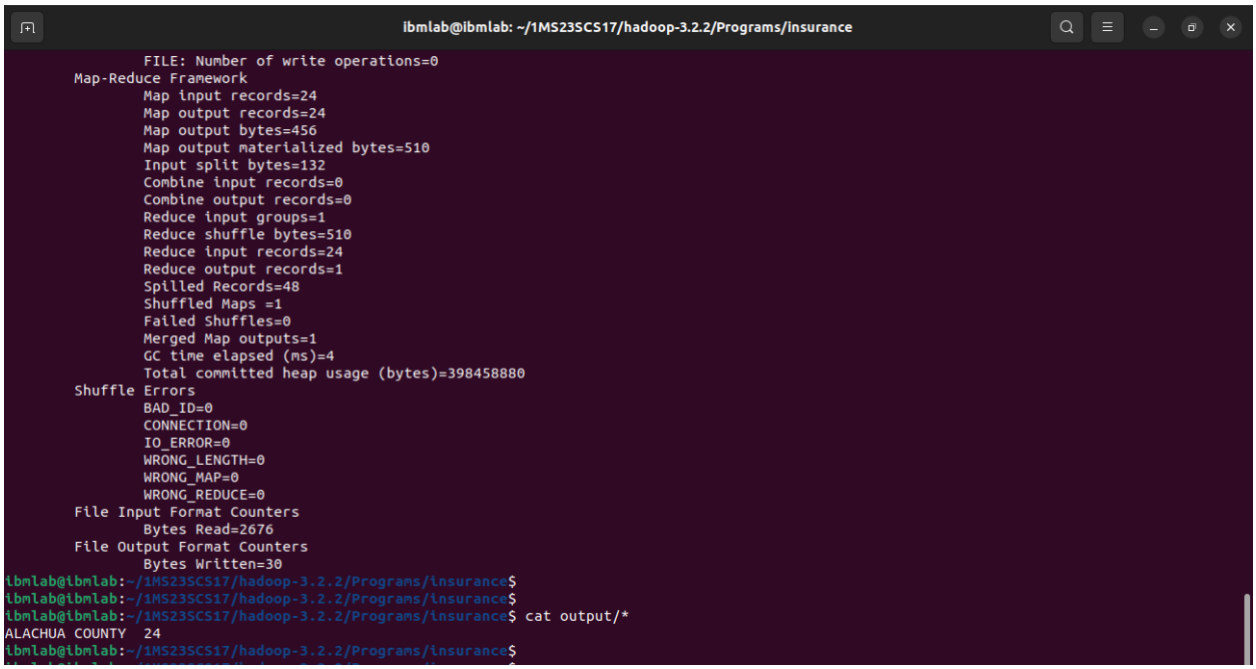
11. input-insurance.csv is input file for Insurance create Input File
12. Run the jar file

```
hadoop jar insurance.jar input-insurance.csv output
```

13. To see the Output

```
cat output/*
```

Output Screenshots



5. Write a MapReduce program using Java, to analyze the given employee record data and generate a statistics report with the total number of Female and Male Employees and their average salary.

driver.java

```
package employee;

import java.io.*;

import java.util.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.fs.Path;


public class driver

{

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

   {

          JobConf conf=new JobConf(driver.class);

          conf.setMapperClass(mapper.class);

          conf.setReducerClass(reducer.class);

          conf.setOutputKeyClass(Text.class);

          conf.setOutputValueClass(DoubleWritable.class);

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

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

          JobClient.runJob(conf);

   }

}
```

mapper.java

```
package employee;

import java.io.*;

import java.util.*;

import org.apache.hadoop.mapred.*;

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

```
class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text ,
DoubleWritable> {


   public void map(LongWritable key, Text value, OutputCollector<Text,DoubleWritable> output
,Reporter r) throws IOException
   {
         String[] line=value.toString().split("\\t");
         Double salary=Double.parseDouble(line[8]);
         output.collect(new Text(line[3]), new DoubleWritable(salary));
   }


}
```

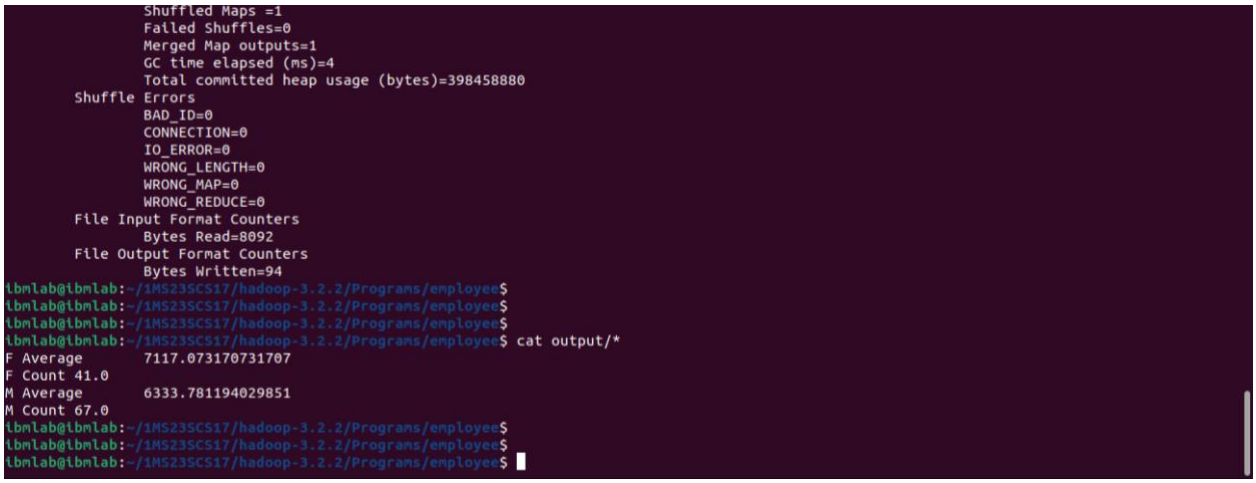
reducer.java

```
package employee;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
class reducer extends MapReduceBase implements
Reducer<Text,DoubleWritable,Text,DoubleWritable> {
public void reduce(Text key,Iterator<DoubleWritable> value ,
OutputCollector<Text,DoubleWritable> output ,Reporter r) throws IOException
   {
         int count=0;
         Double sum=0.0;
         while(value.hasNext()){
               sum+=value.next().get();
               count+=1;
         }
         output.collect(new Text(key+" Average"), new DoubleWritable(sum/count));
         output.collect(new Text(key+" Count"), new DoubleWritable(count));
   }
}
```

Steps to run

1. Create a New File named Bash.sh
2. Copy the Below code and Paste inside Bash.sh and save that File.
   export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
   export PATH=$(echo $PATH):$(pwd)/bin
   export CLASSPATH=$(hadoop classpath)
3. Execute the bash.sh File using following command source Bash.sh.
4. Verify JAVA_HOME variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc file.
5. Verify Hadoop is Installed or not by executing hadoop command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files (driver.java mapper.java reducer.java)
   javac -d . *.java
9. Set driver class in manifest
   echo Main-Class: employee.driver > Manifest.txt
10. Create an executable jar file
    jar cfm employee.jar Manifest.txt employee/*.class
11. input.csv is input file for employee create Input File
12. Run the jar file
    hadoop jar employee.jar input.csv output
13. To see the Output
    cat output/*

Output Screenshots



6. Write a MapReduce program using Java, to analyze the given Sales Records over a period of time and generate data about the country's total sales, and the total number of the products. / Country's total sales and the frequency of the payment mode.

driver.java

```
package sales;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.Path;

public class driver
{
   public static void main(String args[]) throws IOException
   {
         JobConf conf=new JobConf(driver.class);
         conf.setMapperClass(mapper.class);
         conf.setReducerClass(reducer.class);
         conf.setOutputKeyClass(Text.class);
         conf.setOutputValueClass(IntWritable.class);
         FileInputFormat.addInputPath(conf, new Path(args[0]));
         FileOutputFormat.setOutputPath(conf,new Path(args[1]));
         JobClient.runJob(conf);
   }
}
```

mapper.java

```
package sales;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
```

```
public class mapper extends MapReduceBase implements Mapper<LongWritable , Text , Text , IntWritable>
{
   public void map(LongWritable key,Text value,OutputCollector<Text,IntWritable> output,Reporter r) throws IOException
   {
         String[] line=value.toString().split(",");
         int price=Integer.parseInt(line[2]);
         String cardtype=line[3];
         String Country=line[7];
         output.collect(new Text("Country "+Country),new IntWritable(price));
         output.collect(new Text("CardType "+cardtype),new IntWritable(1));
   }
}
```

reducer.java

```
package sales;

import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;

public class reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable>
{
   public void reduce(Text key,Iterator<IntWritable> value,OutputCollector<Text,IntWritable> output ,Reporter r) throws IOException
   {
         int sum=0;
         while(value.hasNext())
         {
               sum+=value.next().get();
```

```
        }
        output.collect(new Text(key),new IntWritable(sum));
    }
}
```

Steps to run

1. Create a New File named Bash.sh
2. Copy the Below code and Paste inside Bash.sh and save that File.

```
export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
export PATH=$(echo $PATH):$(pwd)/bin
export CLASSPATH=$(hadoop classpath)
```

3. Execute the bash.sh File using following command source Bash.sh.
4. Verify JAVA_HOME variable to be set to Java Path and PATH variable has your USN Hadoop Folder.If any previous PATH set to Hadoop Folder remove that inside .bashrc file.
5. Verify Hadoop is Installed or not by executing hadoop command.if command gives Information about Hadoop command then Hadoop is Successfully Installed.
6. Create a folder oddeven and move to that folder
7. Make the driver.java , mapper.java and reducer.java files
8. Compile all java files (driver.java mapper.java reducer.java)

```
javac -d . *.java
```

9. Set driver class in manifest

```
echo Main-Class: sales.driver > Manifest.txt
```

10. Create an executable jar file

```
jar cfm sales.jar Manifest.txt sales/*.class
```

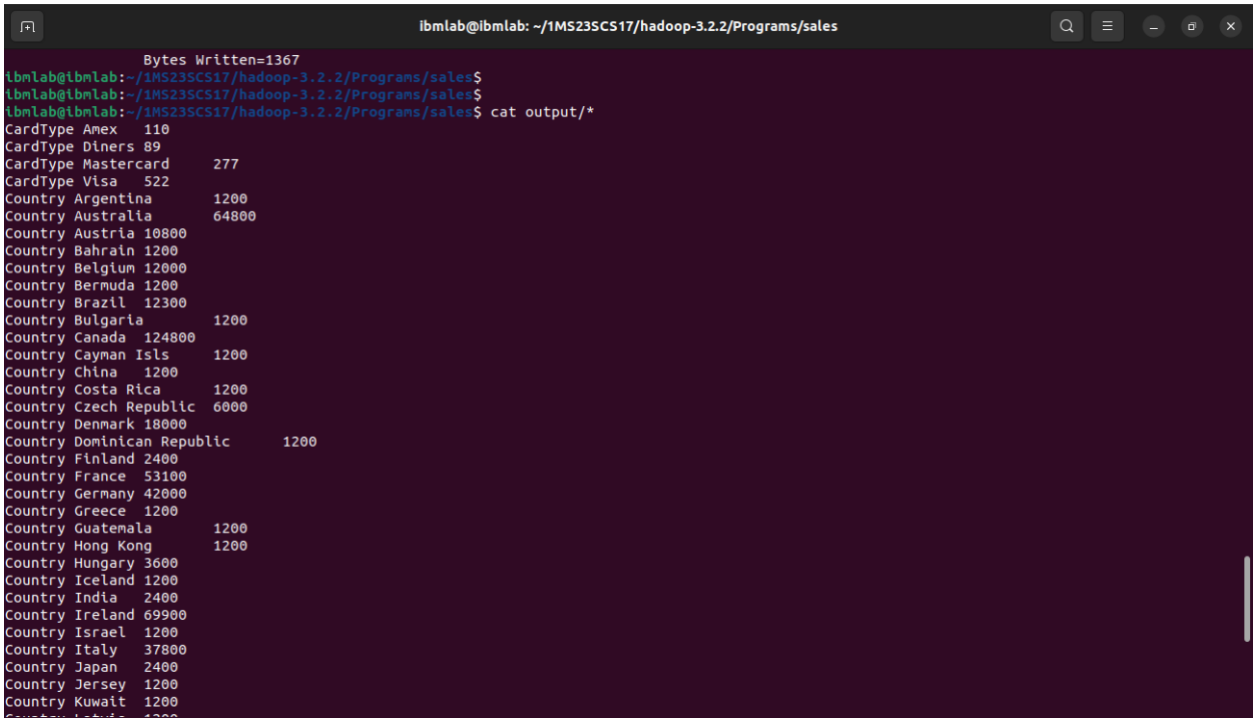
11. sales.txt is input file for Sales create Input File
12. Run the jar file

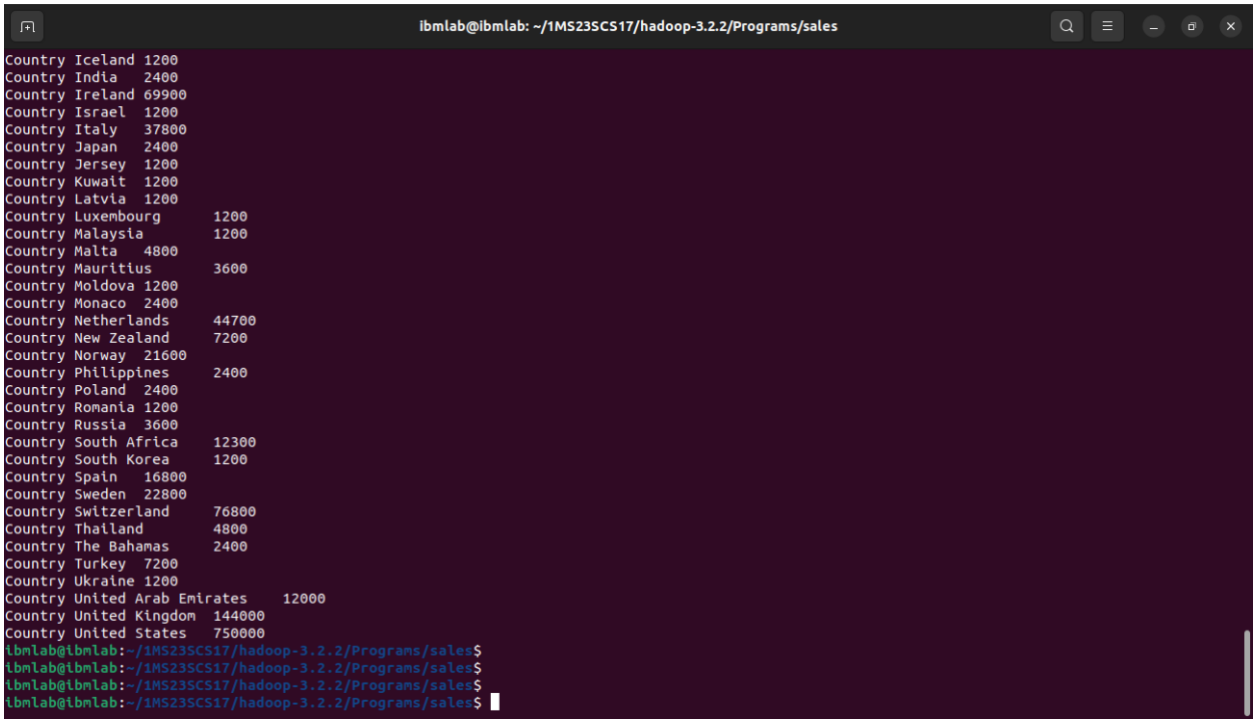
```
hadoop jar sales.jar sales.txt output
```

13. To see the Output

```
cat output/*
```

Output Screenshots





Part-B

1. A. Write a spark program using Python, to analyze the given Weather Report Data and to generate a report with cities having maximum and minimum temperature for a particular year.

Code: weather.py

```
import sys
if(len(sys.argv)!=4):
    print("Provide Input File and Output Directory")
    sys.exit(0)

from pyspark import SparkContext
sc =SparkContext()
f = sc.textFile(sys.argv[1])

temp=f.map(lambda x: (int(x[15:19]),int(x[87:92])))

mini=temp.reduceByKey(lambda a,b:a if a<b else b)
mini.saveAsTextFile(sys.argv[2])

maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[3])
```

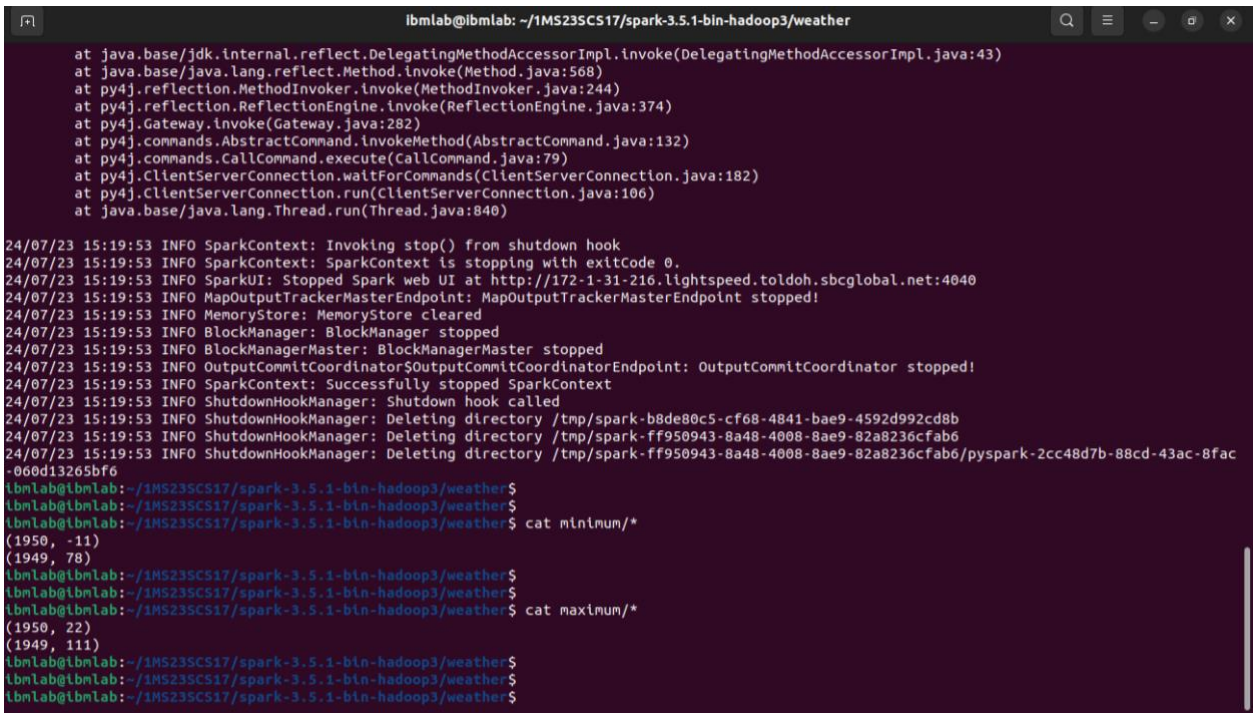
Execution:

spark-submit weather.py input.txt minimum maximum

Output:

$ cat minimum/*

$ cat maximum/*



B. Create/Launch an EC2 instance at Amazon Web Service (AWS) with the following configuration:

- Name: <USN>
- Region: ap-south-1
- Image: Ubuntu Server 22.04 (HVM), SSD Volume Type
- Architecture: 64-bit (x86)
- Instance Type: t2.micro, 1v CPU, 1GB Memory
- Key-pair: RSA type
- Network Settings: Allow SSH traffic from anywhere.

2. A. Write a spark program using Python, to analyze the given Earthquake Data and generate statistics with region and magnitude/ region and depth/ region and latitude/ region and longitude

Code: earthquake.py

```
import sys
from pyspark import SparkContext

if(len(sys.argv)!=6):
    print("Provide Input File and Output Directory")
    sys.exit(0)

sc =SparkContext()
f = sc.textFile(sys.argv[1])

# Region with Magnitude
temp=f.map(lambda x: (x.split(',')[11],float(x.split(',')[8])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])

# Region with Depth
temp=f.map(lambda x: (x.split(',')[11],float(x.split(',')[9])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[3])

# Region with latitude
temp=f.map(lambda x: (x.split(',')[11],float(x.split(',')[6])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[4])

# Region with longitude
temp=f.map(lambda x: (x.split(',')[11],float(x.split(',')[7])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[5])
```

Execution:

spark-submit earthquake.py earthquake-input.csv magnitude depth latitude longitude

Output:

$ cat magnitude/*

$ cat depth/*

```
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$ cat magnitude/*
('Aegean Sea', 5.7)
('Alaska Peninsula', 3.1)
('"Andreanof Islands', 2.7)
('Arizona', 3.1)
('Arkansas', 1.8)
('"Arunachal Pradesh', 4.2)
('"Babuyan Islands region', 4.5)
('"Baja California', 2.8)
('"Acme Islands', 8.9)
('"Andaman Islands', 5.0)
('"Antofagasta', 5.1)
('Central Alaska', 1.5)
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$ cat depth/*
('Aegean Sea', 10.3)
('Alaska Peninsula', 201.5)
('"Andreanof Islands', 96.6)
('Arizona', 6.4)
('Arkansas', 2.1)
('"Arunachal Pradesh', 18.1)
('"Babuyan Islands region', 10.0)
('"Baja California', 41.8)
('"Acme Islands', 59.4)
('"Andaman Islands', 34.2)
('"Antofagasta', 115.3)
('Central Alaska', 0.0)
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
```

$ cat latitude/*

$ cat longitude/*

```
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$ cat magnitude/*
('Aegean Sea', 5.7)
('Alaska Peninsula', 3.1)
('"Andreanof Islands', 2.7)
('Arizona', 3.1)
('Arkansas', 1.8)
('"Arunachal Pradesh', 4.2)
('"Babuyan Islands region', 4.5)
('"Baja California', 2.8)
('"Acme Islands', 8.9)
('"Andaman Islands', 5.0)
('"Antofagasta', 5.1)
('Central Alaska', 1.5)
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$ cat depth/*
('Aegean Sea', 10.3)
('Alaska Peninsula', 201.5)
('"Andreanof Islands', 96.6)
('Arizona', 6.4)
('Arkansas', 2.1)
('"Arunachal Pradesh', 18.1)
('"Babuyan Islands region', 10.0)
('"Baja California', 41.8)
('"Acme Islands', 59.4)
('"Andaman Islands', 34.2)
('"Antofagasta', 115.3)
('Central Alaska', 0.0)
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
ibmlab@ibmlab:~/1MS23SCS17/spark-3.5.1-bin-hadoop3/earthquake$
```

B. Write a python script to start, stop and reboot AWS EC2 Instances using Boto3

```
import boto3

access_key_id = ""
secret_access_key = ""
region = ""
instance_id = ""

ec2 = boto3.client('ec2',
            aws_access_key_id=access_key_id,
            aws_secret_access_key=secret_access_key,
            region_name=region
          )

# start instances
try:
   response = ec2.start_instances(InstanceIds=[instance_id], DryRun=False)
   print(response)
except Exception as e:
   print(e)

# stop instances
try:
   response = ec2.stop_instances(InstanceIds=[instance_id], DryRun=False)
   print(response)
except Exception as e:
   print(e)

# reboot instances
try:
   response = ec2.reboot_instances(InstanceIds=[instance_id], DryRun=False)
   print('Success', response)
except Exception as e:
   print('Error', e)
```

3. A. Write a spark program using Python, to analyze the given Insurance Data and generate a statistics report with the construction building name and the count of building/ county name and its frequency

Code: insurance.py

```
import sys
from pyspark import SparkContext

if(len(sys.argv)!=4):
    print("Provide Input File and Output Directory")
    sys.exit(0)

sc =SparkContext()
f = sc.textFile(sys.argv[1])

# Construction building or Count of building
temp=f.map(lambda x: (x.split(',')[16],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])

# County name and its frequency
temp=f.map(lambda x: (x.split(',')[2],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[3])
```

Execution:

spark-submit insurance.py input-insurance.csv construction county

Output:

$ cat construction/*

$ cat county/*

B. Write a python script to upload any sample file to AWS S3 Bucket using Boto3.

```
import boto3

access_key_id = ""
secret_access_key = ""
region = ""

s3 = boto3.client('s3',
            aws_access_key_id=access_key_id,
            aws_secret_access_key=secret_access_key,
            region_name=region
          )

file_path = "sample_file.txt"
bucket_name = “bucket-name”
object_name = 'sample_file.txt'

try:
   s3.upload_file(file_path, bucket_name, object_name)
   print(f"File '{file_path}' uploaded to bucket '{bucket_name}' as '{object_name}'.")
except Exception as e:
   print(f"An error occurred: {e}")
```

4. A. Write a spark program using Python, to analyze the given Sales Records over a period of time and generate data about the country's total sales, and the total number of the products. / Country's total sales and the frequency of the payment mode.

Code: sales.py

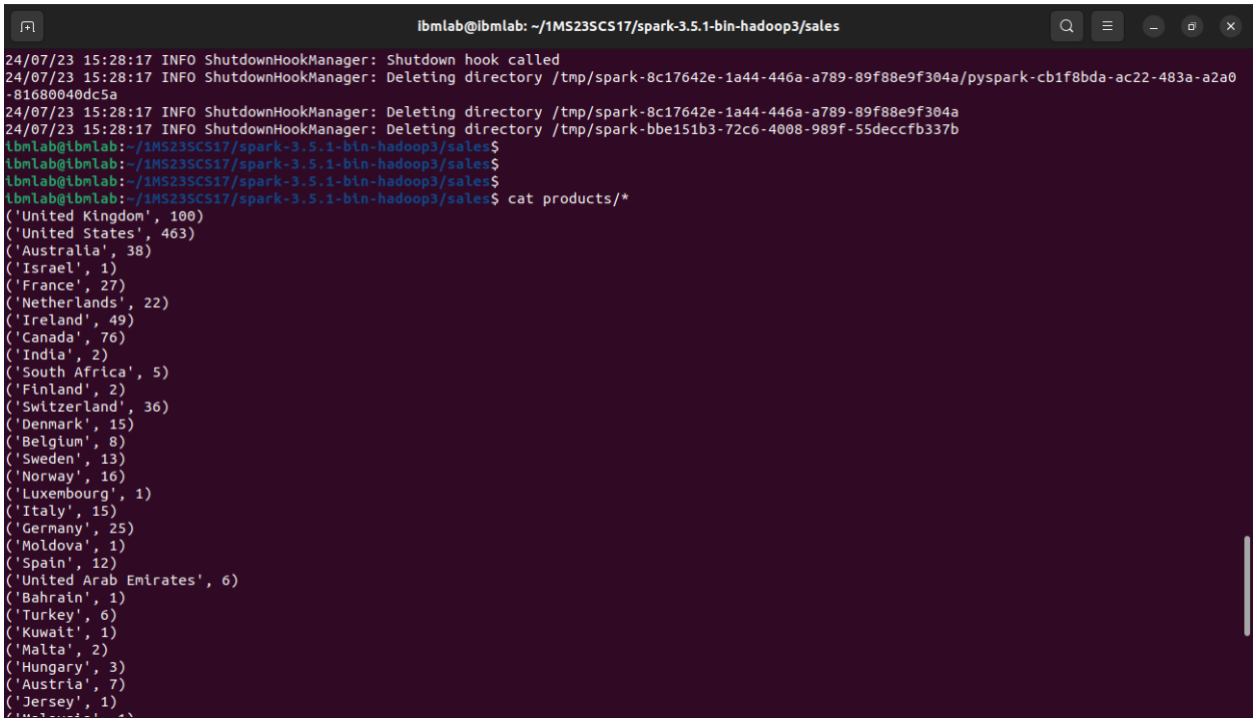
```
import sys
from pyspark import SparkContext

if(len(sys.argv)!=4):
    print("Provide Input File and Output Directory")
    sys.exit(0)

sc =SparkContext()
f = sc.textFile(sys.argv[1])

# Total products
temp=f.map(lambda x: (x.split(',')[7],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])

# Frequency
temp=f.map(lambda x: (x.split(',')[3],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[3])
```

Execution:

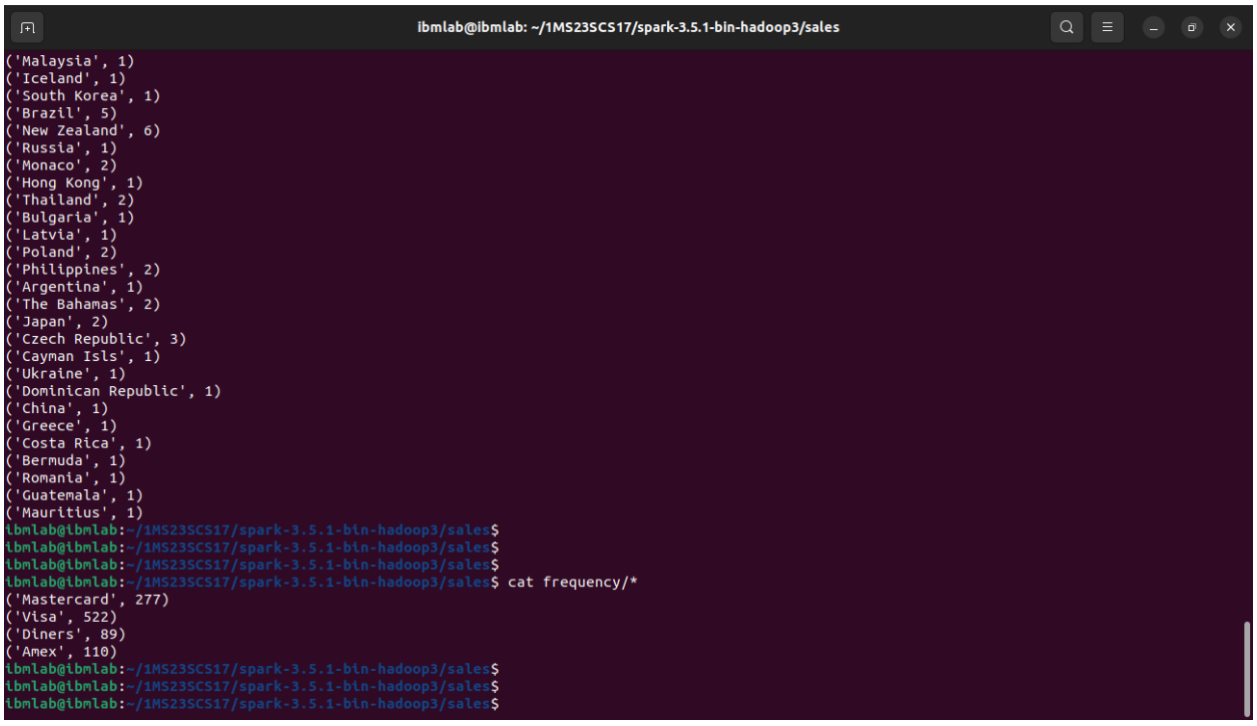
spark-submit sales.py input-sales.csv products frequency

Output:

$ cat products/*

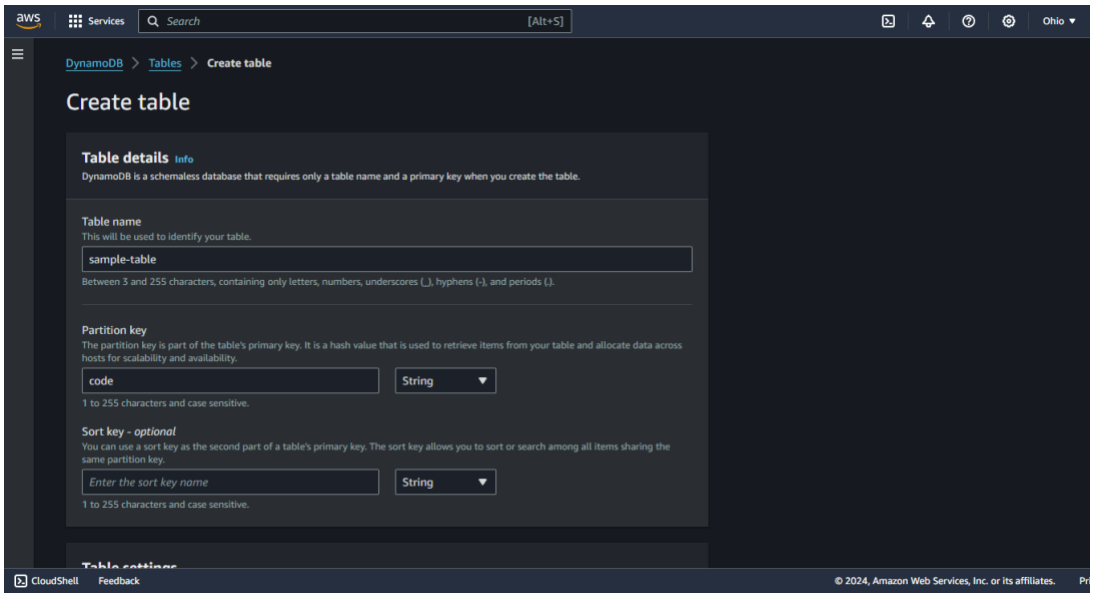


$ cat frequency/*



B. Write a Python script to add data items to AWS DynamoDB using Boto3

```
import boto3

access_key_id = ""
secret_access_key = ""
region = ""
table_name = ""

dynamodb = boto3.resource('dynamodb',
                  aws_access_key_id=access_key_id,
                  aws_secret_access_key=secret_access_key,
                  region_name=region
                 )

data_item = {
   "code": "MCSL26",
   "course": "Cloud Computing and Big Data Laboratory",
   "credits": 1,
}

try:
   table = dynamodb.Table(table_name)
   table.put_item(Item=data_item)
   print(f"Item added to table '{table_name}': {data_item}")
except Exception as e:
   print(f"An error occurred: {e}")
```



5. Write Pig Latin scripts for Crop Production Dataset.

```
crop_prod = LOAD 'Datasets/crop_production.csv' USING PigStorage(',') AS
(State_Name:chararray, District_Name:chararray, Crop_Year:int, Season:chararray,
Crop:chararray, Area:float, Production:float);

DESCRIBE crop_prod;
```

a. Calculate total production of each crop.

```
total_production = GROUP crop_prod BY Crop;

sum_production = FOREACH total_production GENERATE group AS Crop,
SUM(crop_prod.Production) AS Total_Production;

DUMP sum_production;
```

b. Find the average production per year for each crop.

```
grouped_by_crop_year = GROUP crop_prod BY (Crop, Crop_Year);

average_production = FOREACH grouped_by_crop_year GENERATE group.Crop AS Crop,
group.Crop_Year AS Crop_Year, AVG(crop_prod.Production) AS Avg_Production;

DUMP average_production;
```

c. Filter all crops grown in ‘Karnataka’

```
specific_state = FILTER crop_prod BY State_Name == ‘Karnataka’;

unique_crops = GROUP specific_state BY Crop;

DUMP unique_crops;
```

d. Calculate the total area used for each crop in the year 2010.

```
specific_year = FILTER crop_prod BY Crop_Year == 2010;

total_area = GROUP specific_year BY Crop;

sum_area = FOREACH total_area GENERATE group AS Crop, SUM(specific_year.Area)
AS Total_Area;

DUMP sum_area;
```

6. Write Pig Latin scripts for **Olympic Athletes and Hosts Datasets**.

```
athletes = LOAD 'olympic_athletes.csv' USING PigStorage(',') AS (athlete_url: chararray, athlete_full_name: chararray, games_participations: int, first_game: chararray, athlete_year_birth: float, athlete_medals: chararray, bio: chararray);

hosts = LOAD 'olympic_hosts.csv' USING PigStorage(',') AS (game_slug: chararray, game_end_date: chararray, game_start_date: chararray, game_location: chararray, game_name: chararray, game_season: chararray, game_year: int);

DESCRIBE athletes;

DESCRIBE hosts;
```

a. Filter athletes participated in the “Tokyo 2020” games.

```
tokyo_2020_athletes = FILTER athletes BY first_game == ‘Tokyo 2020’;

DUMP tokyo_2020_athletes;
```

b. Filter the games held in “China”.

```
games_in_china = FILTER hosts BY game_location == 'China';

DUMP games_in_china;
```

c. Group games by season and count the number of games in each session.

```
grouped_by_season = GROUP hosts BY game_season;

counted_by_season = FOREACH grouped_by_season GENERATE group AS game_season, COUNT(hosts) AS num_games;

DUMP counted_by_season;
```

d. Filter games that occurred after the year 2000.

```
games_after_2000 = FILTER hosts BY game_year > 2000;

DUMP games_after_2000;
```