RAMAIAH
Institute of Technology

Department of Computer Science And Engineering

Big Data Lab Record

CSL76

NAME: DANISH MAHAJAN
USN: 1MS20CS037
SECTION: A

Faculty Signature

HADOOP PROGRAMS

Bash file:

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

Command to run it: **source bash.sh**

Programs:

1. **Write a MapReduce program to analyse 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));
                 }
          }
   }
}
```

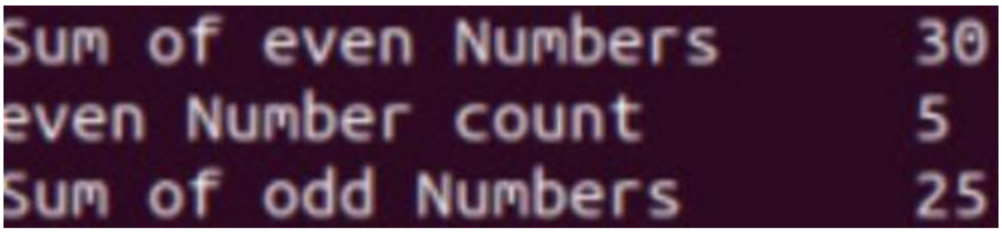
reducer.java

```
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));
   }
}
```

oe.txt

```
1 2 3 4 5 6 7 8 9 10
```

Output:



```
Sum of even Numbers     30
even Number count       5
Sum of odd Numbers      25
```

2. **Write a MapReduce program 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);
   }
}
```

mapper.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));
   }
}
```

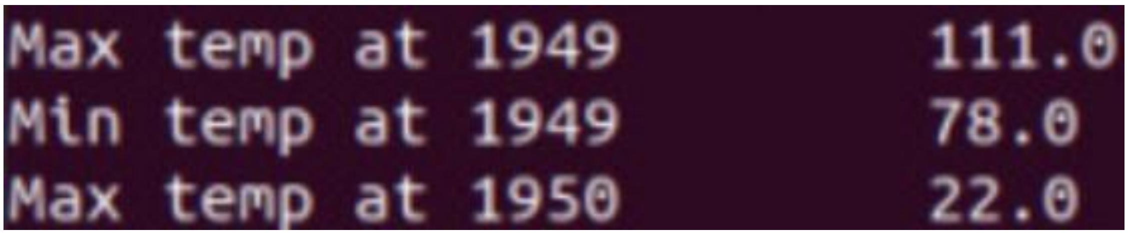
reducer.java

```
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+038299999V0203301N00671220
001CN9999999N9+00001+99999999999
0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220
001CN9999999N9+00221+99999999999
0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220
001CN9999999N9-00111+99999999999
0043012650999991949032412004+62300+010750FM-12+048599999V0202701N0046122
0001CN0500001N9+01111+99999999999
0043012650999991949032418004+62300+010750FM-12+048599999V0202701N0046122
0001CN0500001N9+00781+99999999999
```

Output:



```
Max temp at 1949        111.0
Min temp at 1949        78.0
Max temp at 1950        22.0
```

3. **Write a MapReduce program 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);
   }
```

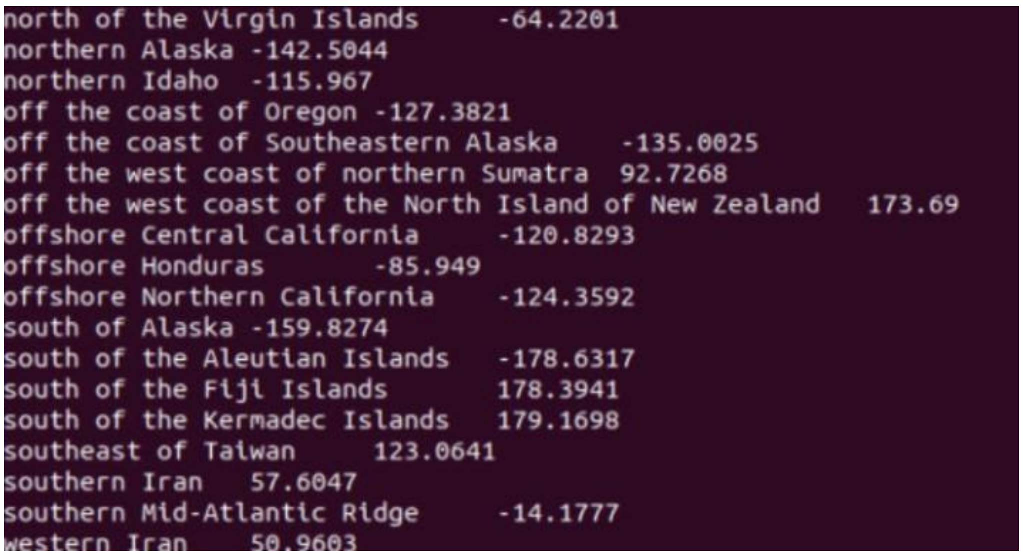
mapper.java

```
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));
  }
}
```

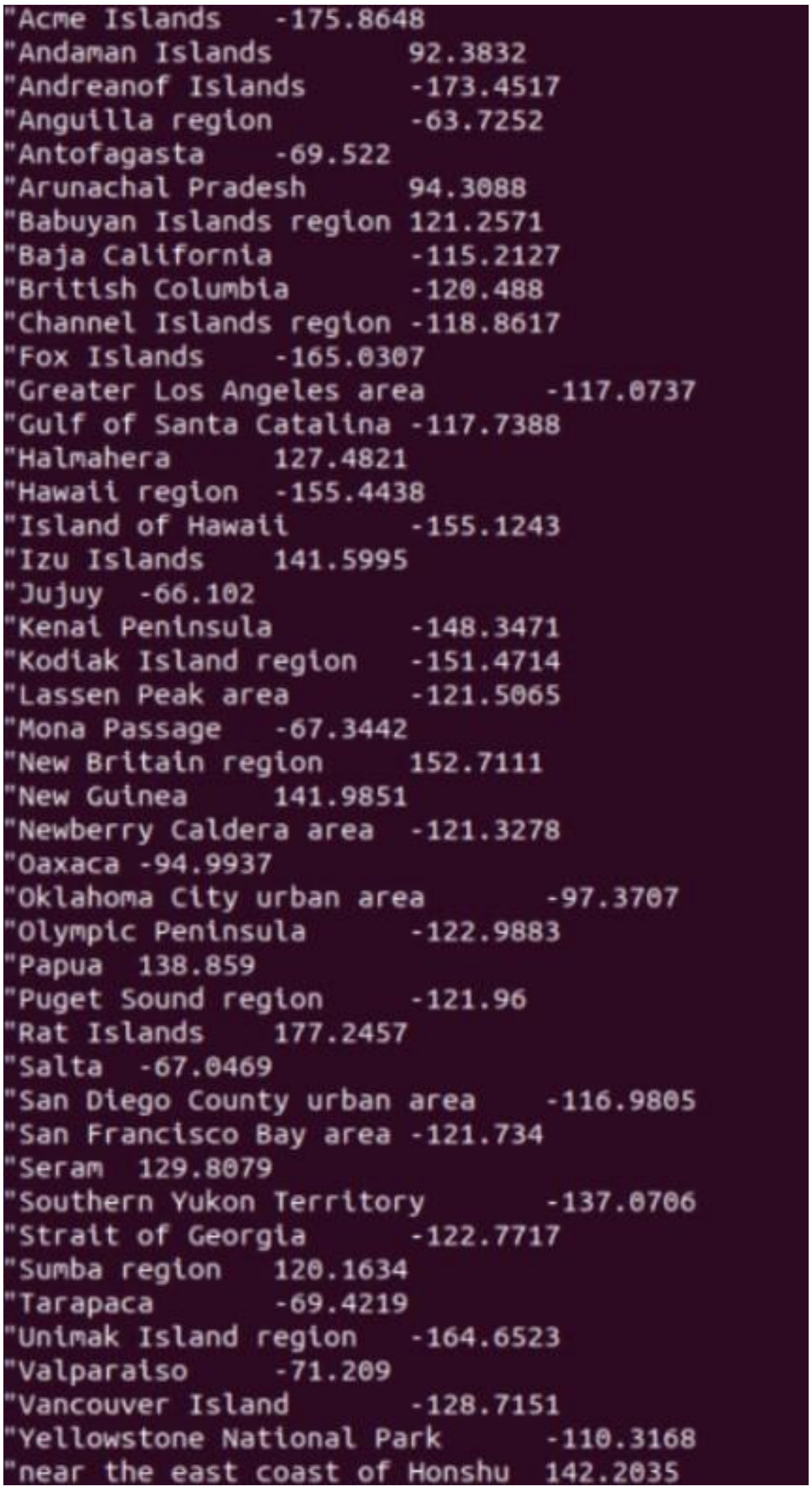
reducer.java

```
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));
  }
```

Output:



```
north of the Virgin Islands     -64.2201
northern Alaska -142.5044
northern Idaho  -115.967
off the coast of Oregon -127.3821
off the coast of Southeastern Alaska    -135.0025
off the west coast of northern Sumatra  92.7268
off the west coast of the North Island of New Zealand   173.69
offshore Central California     -120.8293
offshore Honduras       -85.949
offshore Northern California    -124.3592
south of Alaska -159.8274
south of the Aleutian Islands   -178.6317
south of the Fiji Islands       178.3941
south of the Kermadec Islands   179.1698
southeast of Taiwan     123.0641
southern Iran   57.6047
southern Mid-Atlantic Ridge     -14.1777
western Iran    50.9603
```



```
"Acme Islands   -175.8648
"Andaman Islands        92.3832
"Andreanof Islands      -173.4517
"Anguilla region        -63.7252
"Antofagasta    -69.522
"Arunachal Pradesh      94.3088
"Babuyan Islands region 121.2571
"Baja California        -115.2127
"British Columbia       -120.488
"Channel Islands region -118.8617
"Fox Islands    -165.0307
"Greater Los Angeles area       -117.0737
"Gulf of Santa Catalina -117.7388
"Halmahera      127.4821
"Hawaii region  -155.4438
"Island of Hawaii       -155.1243
"Izu Islands    141.5995
"Jujuy  -66.102
"Kenai Peninsula        -148.3471
"Kodiak Island region   -151.4714
"Lassen Peak area       -121.5065
"Mona Passage   -67.3442
"New Britain region     152.7111
"New Guinea     141.9851
"Newberry Caldera area  -121.3278
"Oaxaca -94.9937
"Oklahoma City urban area       -97.3707
"Olympic Peninsula      -122.9883
"Papua  138.859
"Puget Sound region     -121.96
"Rat Islands    177.2457
"Salta  -67.0469
"San Diego County urban area    -116.9805
"San Francisco Bay area -121.734
"Seram  129.8079
"Southern Yukon Territory       -137.0706
"Strait of Georgia      -122.7717
"Sumba region   120.1634
"Tarapaca       -69.4219
"Unimak Island region   -164.6523
"Valparaiso     -71.209
"Vancouver Island       -128.7151
"Yellowstone National Park      -110.3168
"near the east coast of Honshu  142.2035
```

4. **Write a MapReduce program 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);
   }
}
```

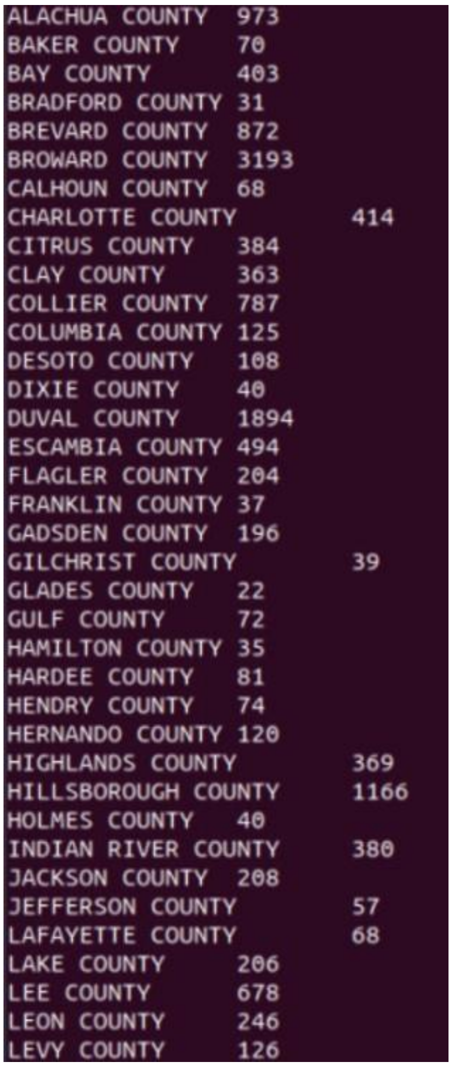
mapper.java

```
package insurance;
import java.io.*;
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));
   }
}
```

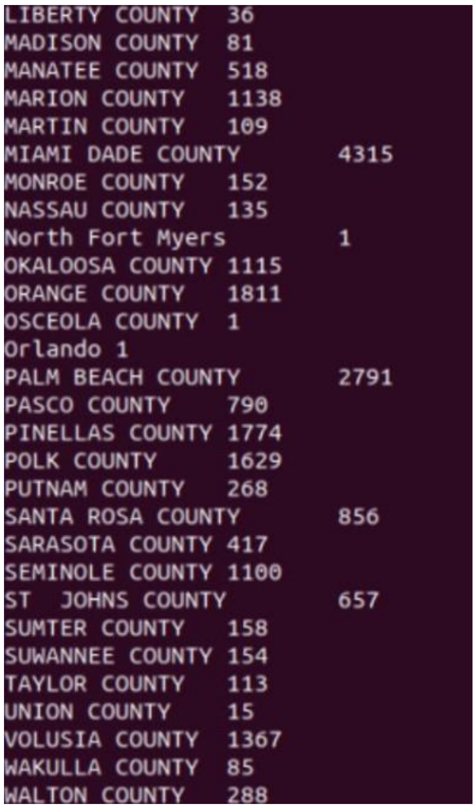
reducer.java

```
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));
   }
}
```

Output:



```
ALACHUA COUNTY  973
BAKER COUNTY    70
BAY COUNTY      403
BRADFORD COUNTY 31
BREVARD COUNTY  872
BROWARD COUNTY  3193
CALHOUN COUNTY  68
CHARLOTTE COUNTY        414
CITRUS COUNTY   384
CLAY COUNTY     363
COLLIER COUNTY  787
COLUMBIA COUNTY 125
DESOTO COUNTY   108
DIXIE COUNTY    40
DUVAL COUNTY    1894
ESCAMBIA COUNTY 494
FLAGLER COUNTY  204
FRANKLIN COUNTY 37
GADSDEN COUNTY  196
GILCHRIST COUNTY        39
GLADES COUNTY   22
GULF COUNTY     72
HAMILTON COUNTY 35
HARDEE COUNTY   81
HENDRY COUNTY   74
HERNANDO COUNTY 120
HIGHLANDS COUNTY        369
HILLSBOROUGH COUNTY     1166
HOLMES COUNTY   40
INDIAN RIVER COUNTY     380
JACKSON COUNTY  208
JEFFERSON COUNTY        57
LAFAYETTE COUNTY        68
LAKE COUNTY     206
LEE COUNTY      678
LEON COUNTY     246
LEVY COUNTY     126
```



```
LIBERTY COUNTY  36
MADISON COUNTY  81
MANATEE COUNTY  518
MARION COUNTY   1138
MARTIN COUNTY   109
MIAMI DADE COUNTY       4315
MONROE COUNTY   152
NASSAU COUNTY   135
North Fort Myers        1
OKALOOSA COUNTY 1115
ORANGE COUNTY   1811
OSCEOLA COUNTY  1
Orlando 1
PALM BEACH COUNTY       2791
PASCO COUNTY    790
PINELLAS COUNTY 1774
POLK COUNTY     1629
PUTNAM COUNTY   268
SANTA ROSA COUNTY       856
SARASOTA COUNTY 417
SEMINOLE COUNTY 1100
ST  JOHNS COUNTY        657
SUMTER COUNTY   158
SUWANNEE COUNTY 154
TAYLOR COUNTY   113
UNION COUNTY    15
VOLUSIA COUNTY  1367
WAKULLA COUNTY  85
WALTON COUNTY   288
```

5. **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);
    }
}
```

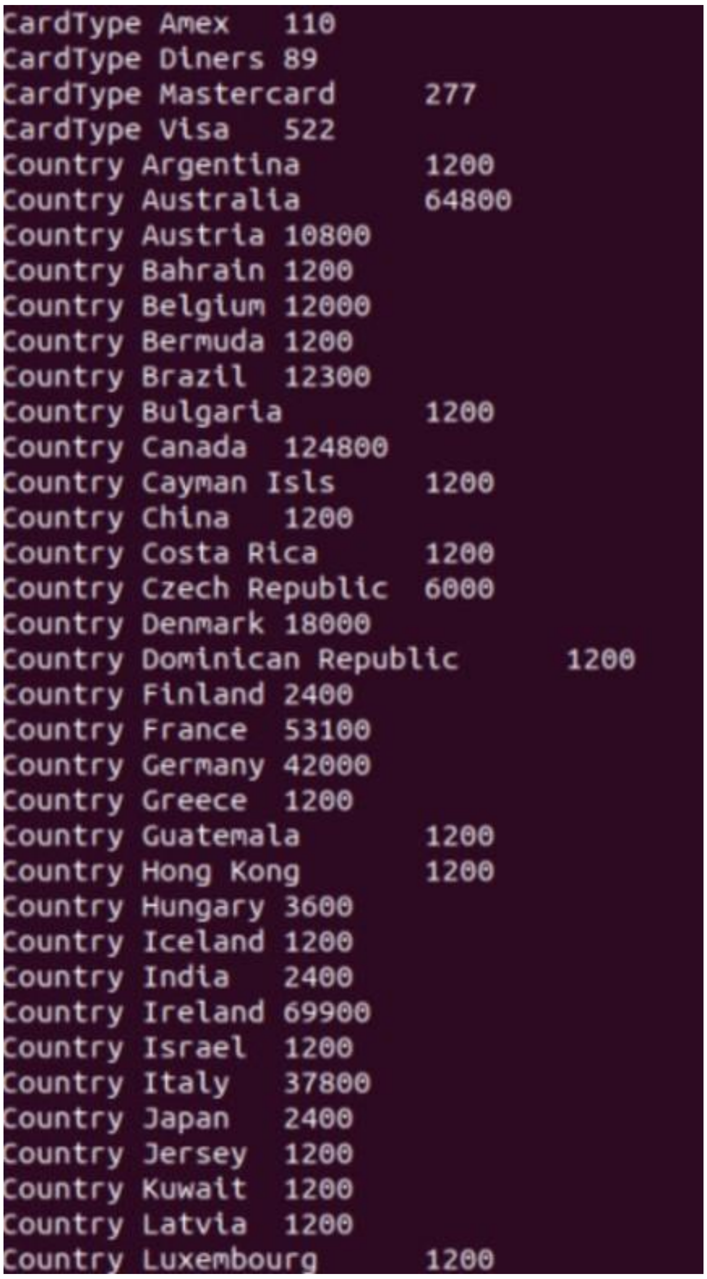
mapper.java

```
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));
    }
}
```

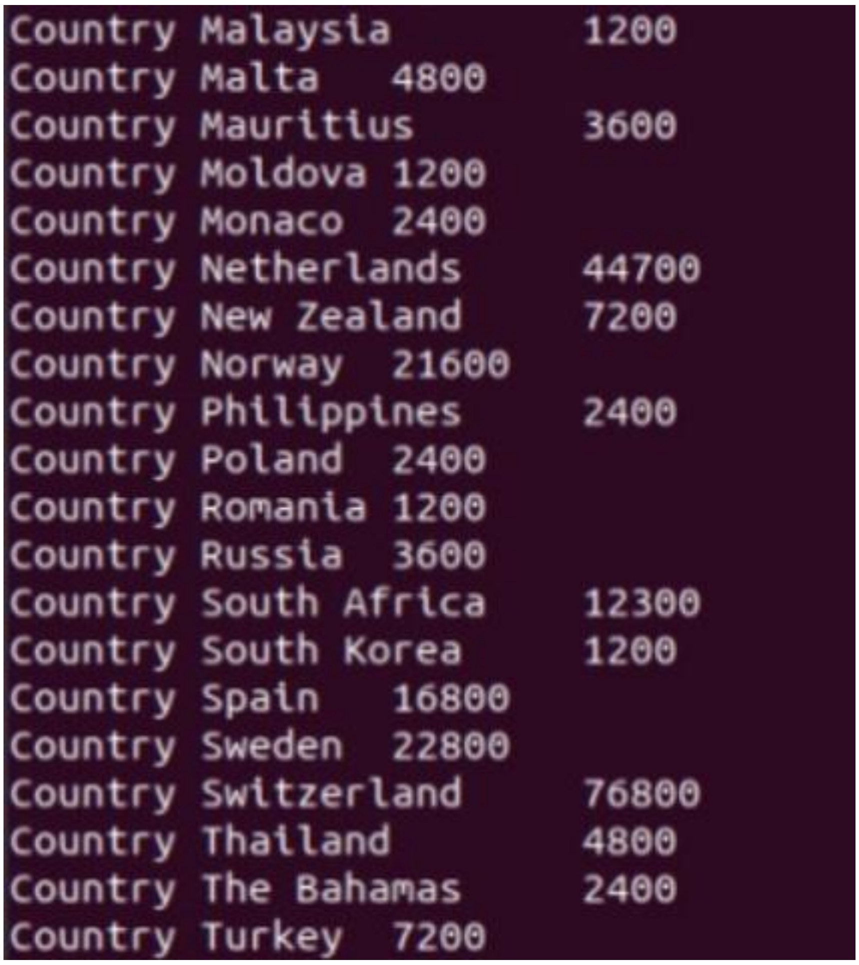
reducer.java

```
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));
    }
}
```

Output:



```
CardType Amex   110
CardType Diners 89
CardType Mastercard     277
CardType Visa   522
Country Argentina       1200
Country Australia       64800
Country Austria 10800
Country Bahrain 1200
Country Belgium 12000
Country Bermuda 1200
Country Brazil  12300
Country Bulgaria        1200
Country Canada  124800
Country Cayman Isls     1200
Country China   1200
Country Costa Rica      1200
Country Czech Republic  6000
Country Denmark 18000
Country Dominican Republic      1200
Country Finland 2400
Country France  53100
Country Germany 42000
Country Greece  1200
Country Guatemala       1200
Country Hong Kong       1200
Country Hungary 3600
Country Iceland 1200
Country India   2400
Country Ireland 69900
Country Israel  1200
Country Italy   37800
Country Japan   2400
Country Jersey  1200
Country Kuwait  1200
Country Latvia  1200
Country Luxembourg      1200
```



```
Country Malaysia        1200
Country Malta   4800
Country Mauritius       3600
Country Moldova 1200
Country Monaco  2400
Country Netherlands     44700
Country New Zealand     7200
Country Norway  21600
Country Philippines     2400
Country Poland  2400
Country Romania 1200
Country Russia  3600
Country South Africa    12300
Country South Korea     1200
Country Spain   16800
Country Sweden  22800
Country Switzerland     76800
Country Thailand        4800
Country The Bahamas     2400
Country Turkey  7200
```

6. **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);
    }
}
```

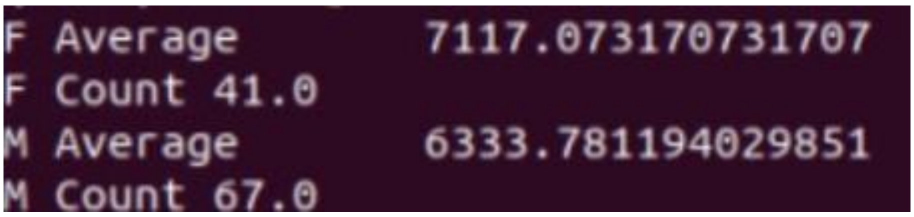
mapper.java

```
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");
                        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));
    }
}
```

Output:



```
F Average       7117.073170731707
F Count 41.0
M Average       6333.781194029851
M Count 67.0
```

7. **Write a MapReduce program using java, to demonstrate matrix multiplication.**

driver.java

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

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(Text.class);
           FileInputFormat.addInputPath(conf,new Path(args[0]));
           FileOutputFormat.setOutputPath(conf,new Path(args[1]));
           JobClient.runJob(conf);

   }
}
```

mapper.java

```
package matrix;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
class mapper extends MapReduceBase implements Mapper<LongWritable, Text, Text,Text>
{
   public void map(LongWritable key, Text value, OutputCollector<Text,Text> output,
Reporter r) throws IOException
   {
          String line[]=value.toString().split(",");
          Text OutputKey=new Text();
          Text OutputValue=new Text();
          if(line[0].equals("A"))
          {

                 for(int i=0;i<3;i++)
                 {
                        OutputKey.set(line[1]+","+i);
                        OutputValue.set("A,"+line[2]+","+line[3]);
                        output.collect(OutputKey,OutputValue);

                 }
          }
          else
          {
                 for(int i=0;i<2;i++)
                 {
                        OutputKey.set(i+","+line[2]);
                        OutputValue.set("B,"+line[1]+","+line[3]);
                        output.collect(OutputKey,OutputValue);

                 }
          }
   }
}
```

reducer.java

```
package matrix;
import java.util.*;
import java.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
public class reducer extends MapReduceBase implements Reducer<Text,Text,Text,Text>
{
   public void reduce(Text key ,Iterator<Text> value , OutputCollector<Text,Text>
output,Reporter r) throws IOException
   {
          HashMap<Integer,Float> a=new HashMap<Integer,Float>();
          HashMap<Integer,Float> b=new HashMap<Integer,Float>();
          String[] v;
          while(value.hasNext())
          {
                 v=value.next().toString().split(",");
```

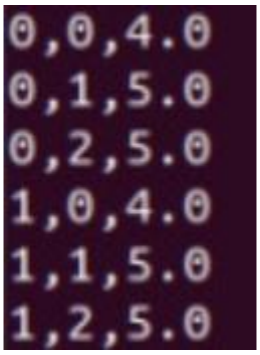
```
                    if(v[0].equals("A"))
                    {
                              a.put(Integer.parseInt(v[1]),Float.parseFloat(v[2]));
                    }
                    else
                    {
                              b.put(Integer.parseInt(v[1]),Float.parseFloat(v[2]));
                    }
          }
          float aij,bij, result=0.0f;
          for(int i=0;i<5;i++)
          {
                    aij=a.containsKey(i) ? a.get(i): 0.0f;
                    bij=b.containsKey(i) ? b.get(i): 0.0f;
                    result+=aij*bij;
          }
          if(result!=0.0f)
          {
                    output.collect(null,new Text(key+","+Float.toString(result)));
          }
   }

}
```

input.txt

```
A,0,0,1.0
A,0,1,1.0
A,0,2,1.0
A,0,3,1.0
A,0,4,1.0
A,1,0,2.0
A,1,1,2.0
A,1,2,2.0
A,1,3,2.0
A,1,4,2.0
B,0,0,1.0
B,0,1,1.0
B,0,2,1.0
B,1,0,1.0
B,1,1,1.0
B,1,2,1.0
B,2,0,1.0
B,2,1,1.0
B,2,2,1.0
B,3,0,1.0
B,3,1,1.0
B,3,2,1.0
```

```
B,4,0,1.0
B,4,1,1.0
B,4,2,1.0
```

Output:



```
0,0,4.0
0,1,5.0
0,2,5.0
1,0,4.0
1,1,5.0
1,2,5.0
```

8. **Write a MapReduce program using java, to find out the word count from a given input file.**

driver.java

```
package wordcount;

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 Exception
   {
          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 wordcount;
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 a:line){
                 output.collect(new Text(a),new IntWritable(1));
          }
   }
}
```

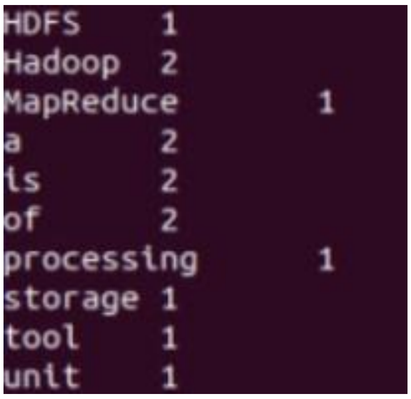
reducer.java

```
package wordcount;
import java.io.*;
import java.util.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.io.*;
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 count=0;
          while(value.hasNext())
          {
                 count+=value.next().get();
          }
          output.collect(new Text(key),new IntWritable(count));
   }

}
```

input.txt

```
HDFS is a storage unit of Hadoop
MapReduce is a processing tool of Hadoop
```

Output:



```
HDFS    1
Hadoop  2
MapReduce       1
a       2
is      2
of      2
processing      1
storage 1
tool    1
unit    1
```

SPARK PROGRAMS

Bash file:

```
export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
if ! command -v spark-shell --version &> /dev/null
then
   export PATH=$(echo $PATH):$(pwd)/bin
fi
```

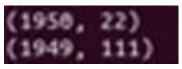
Command to run it: **source bash.sh**

Programs:

1. **Write a spark to analyze the given weather report data and to generate a report with cities having maximum temperature for a particular year**

```
import sys
if(len(sys.argv)!=3):
        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])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Output:

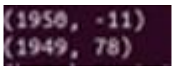


```
(1950, 22)
(1949, 111)
```

2. **Write a spark to analyze the given weather report data and to generate a report with cities having minimum temperature for a particular year**

```
import sys
if(len(sys.argv)!=3):
        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])
```

Output:

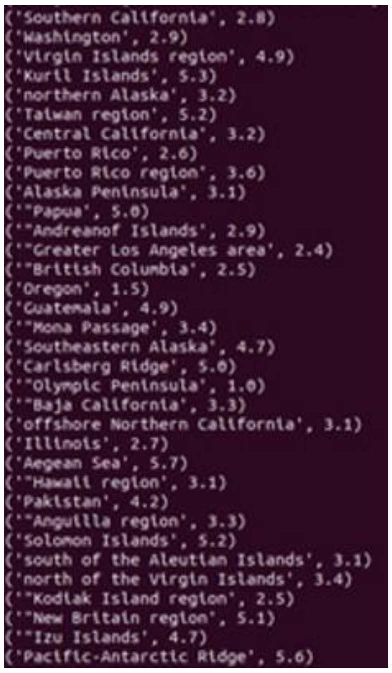


```
(1950, -11)
(1949, 78)
```

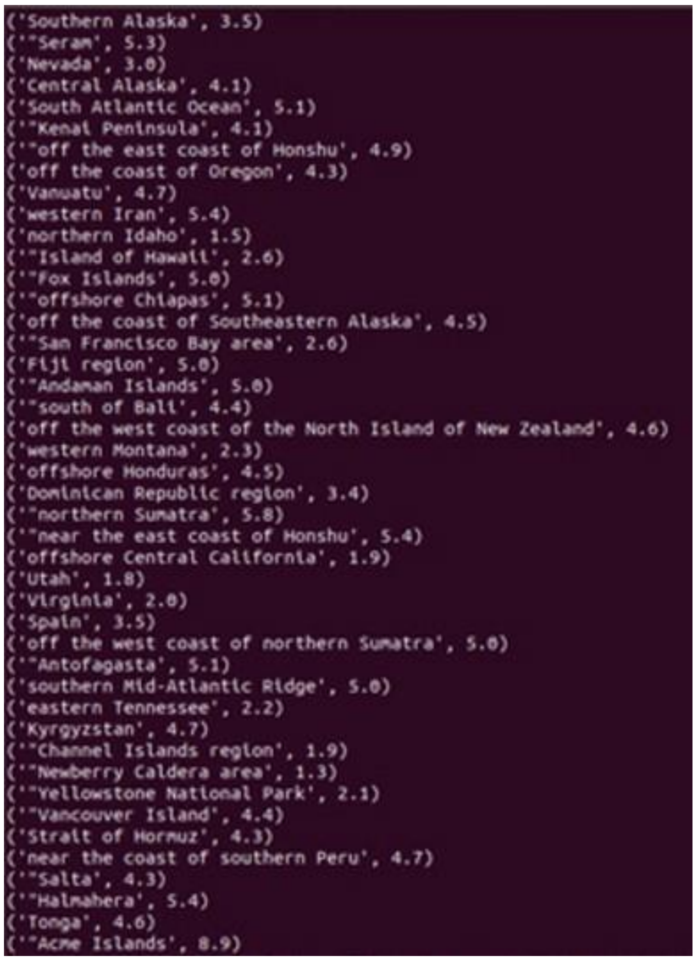
3. **Write a spark program to analyze the given Earthquake data and generate statistics with region and magnitude**

```
import sys
if(len(sys.argv)!=3):
        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: (x.split(',')[11],float(x.split(',')[8])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Output:



```
('Southern California', 2.8)
('Washington', 2.9)
('Virgin Islands region', 4.9)
('Kuril Islands', 5.3)
('northern Alaska', 3.2)
('Taiwan region', 5.2)
('Central California', 3.2)
('Puerto Rico', 2.6)
('Puerto Rico region', 3.6)
('Alaska Peninsula', 3.1)
('"Papua', 5.0)
('"Andreanof Islands', 2.9)
('"Greater Los Angeles area', 2.4)
('"British Columbia', 2.5)
('Oregon', 1.5)
('Guatemala', 4.9)
('"Mona Passage', 3.4)
('Southeastern Alaska', 4.7)
('Carlsberg Ridge', 5.0)
('"Olympic Peninsula', 1.0)
('"Baja California', 3.3)
('offshore Northern California', 3.1)
('Illinois', 2.7)
('Aegean Sea', 5.7)
('"Hawaii region', 3.1)
('Pakistan', 4.2)
('"Anguilla region', 3.3)
('Solomon Islands', 5.2)
('south of the Aleutian Islands', 3.1)
('north of the Virgin Islands', 3.4)
('"Kodiak Island region', 2.5)
('"New Britain region', 5.1)
('"Izu Islands', 4.7)
('Pacific-Antarctic Ridge', 5.6)
```



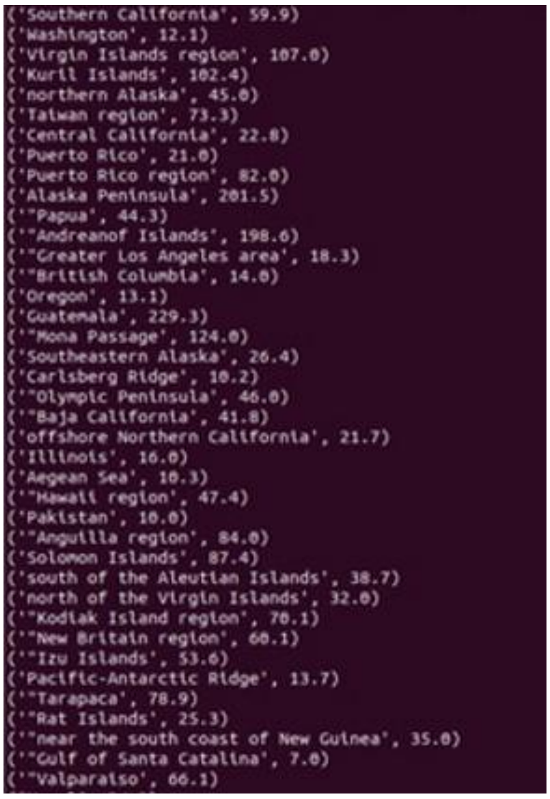
```
('Southern Alaska', 3.5)
('"Seram', 5.3)
('Nevada', 3.0)
('Central Alaska', 4.1)
('South Atlantic Ocean', 5.1)
('"Kenai Peninsula', 4.1)
('"off the east coast of Honshu', 4.9)
('off the coast of Oregon', 4.3)
('Vanuatu', 4.7)
('western Iran', 5.4)
('northern Idaho', 1.5)
('"Island of Hawaii', 2.6)
('"Fox Islands', 5.0)
('"offshore Chiapas', 5.1)
('off the coast of Southeastern Alaska', 4.5)
('"San Francisco Bay area', 2.6)
('Fiji region', 5.0)
('"Andaman Islands', 5.0)
('"south of Bali', 4.4)
('off the west coast of the North Island of New Zealand', 4.6)
('western Montana', 2.3)
('offshore Honduras', 4.5)
('Dominican Republic region', 3.4)
('"northern Sumatra', 5.8)
('"near the east coast of Honshu', 5.4)
('offshore Central California', 1.9)
('Utah', 1.8)
('Virginia', 2.0)
('Spain', 3.5)
('off the west coast of northern Sumatra', 5.0)
('"Antofagasta', 5.1)
('southern Mid-Atlantic Ridge', 5.0)
('eastern Tennessee', 2.2)
('Kyrgyzstan', 4.7)
('"Channel Islands region', 1.9)
('"Newberry Caldera area', 1.3)
('"Yellowstone National Park', 2.1)
('"Vancouver Island', 4.4)
('Strait of Hormuz', 4.3)
('near the coast of southern Peru', 4.7)
('"Salta', 4.3)
('"Halmahera', 5.4)
('Tonga', 4.6)
('"Acme Islands', 8.9)
```

4. **Write a spark program to analyze the given Earthquake data and generate statistics with region and depth**

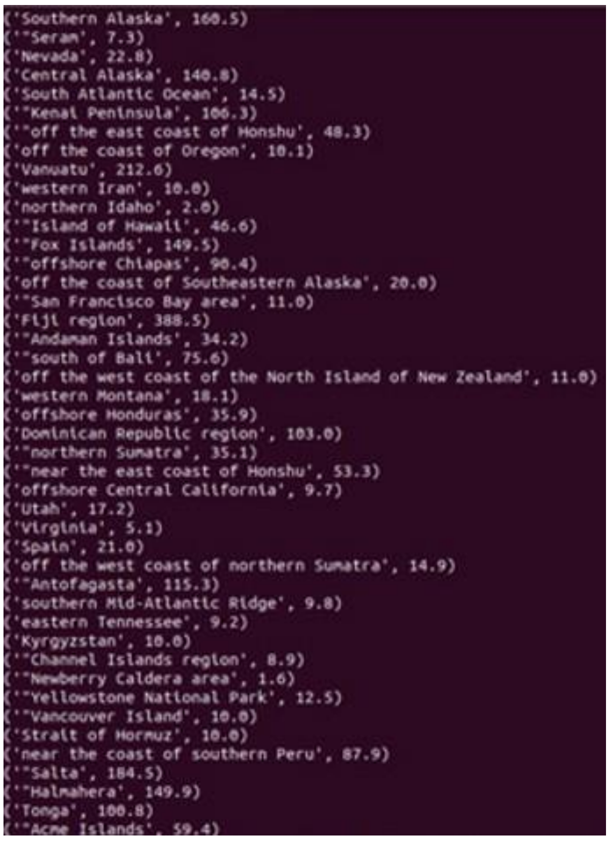
```
import sys
```

```
if(len(sys.argv)!=3):
        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: (x.split(',')[11],float(x.split(',')[9])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Output:



```
('Southern California', 59.9)
('Washington', 12.1)
('Virgin Islands region', 107.0)
('Kuril Islands', 102.4)
('northern Alaska', 45.0)
('Taiwan region', 73.3)
('Central California', 22.8)
('Puerto Rico', 21.0)
('Puerto Rico region', 82.0)
('Alaska Peninsula', 201.5)
('"Papua', 44.3)
('"Andreanof Islands', 198.6)
('"Greater Los Angeles area', 18.3)
('"British Columbia', 14.0)
('Oregon', 13.1)
('Guatemala', 229.3)
('"Mona Passage', 124.0)
('Southeastern Alaska', 26.4)
('Carlsberg Ridge', 10.2)
('"Olympic Peninsula', 46.0)
('"Baja California', 41.8)
('offshore Northern California', 21.7)
('Illinois', 16.0)
('Aegean Sea', 10.3)
('"Hawaii region', 47.4)
('Pakistan', 10.0)
('"Anguilla region', 84.0)
('Solomon Islands', 87.4)
('south of the Aleutian Islands', 38.7)
('north of the Virgin Islands', 32.0)
('"Kodiak Island region', 70.1)
('"New Britain region', 60.1)
('"Izu Islands', 53.6)
('Pacific-Antarctic Ridge', 13.7)
('"Tarapaca', 78.9)
('"Rat Islands', 25.3)
('"near the south coast of New Guinea', 35.0)
('"Gulf of Santa Catalina', 7.0)
('"Valparaiso', 66.1)
```

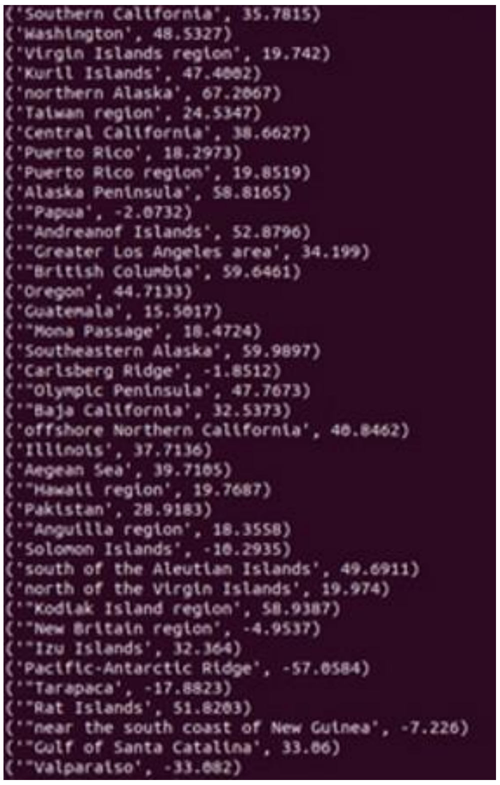


```
('Southern Alaska', 160.5)
('"Seram', 7.3)
('Nevada', 22.8)
('Central Alaska', 140.8)
('South Atlantic Ocean', 14.5)
('"Kenai Peninsula', 106.3)
('"off the east coast of Honshu', 48.3)
('off the coast of Oregon', 10.1)
('Vanuatu', 212.6)
('western Iran', 10.0)
('northern Idaho', 2.0)
('"Island of Hawaii', 46.6)
('"Fox Islands', 149.5)
('"offshore Chiapas', 90.4)
('off the coast of Southeastern Alaska', 20.0)
('"San Francisco Bay area', 11.0)
('Fiji region', 388.5)
('"Andaman Islands', 34.2)
('"south of Bali', 75.6)
('off the west coast of the North Island of New Zealand', 11.0)
('western Montana', 18.1)
('offshore Honduras', 35.9)
('Dominican Republic region', 103.0)
('"northern Sumatra', 35.1)
('"near the east coast of Honshu', 53.3)
('offshore Central California', 9.7)
('Utah', 17.2)
('Virginia', 5.1)
('Spain', 21.0)
('off the west coast of northern Sumatra', 14.9)
('"Antofagasta', 115.3)
('southern Mid-Atlantic Ridge', 9.8)
('eastern Tennessee', 9.2)
('Kyrgyzstan', 10.0)
('"Channel Islands region', 8.9)
('"Newberry Caldera area', 1.6)
('"Yellowstone National Park', 12.5)
('"Vancouver Island', 10.0)
('Strait of Hormuz', 10.0)
('near the coast of southern Peru', 87.9)
('"Salta', 184.5)
('"Halmahera', 149.9)
('Tonga', 100.8)
('"Acme Islands', 59.4)
```

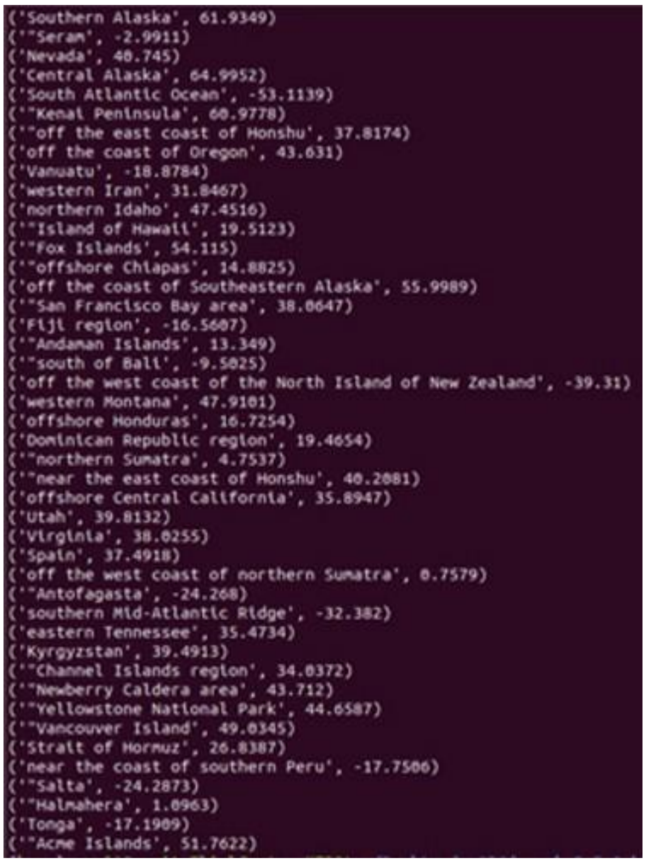
5. **Write a spark program to analyze the given Earthquake data and generate statistics with region and latitude**

```
import sys
if(len(sys.argv)!=3):
        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: (x.split(',')[11],float(x.split(',')[6])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Output:



```
('Southern California', 35.7815)
('Washington', 48.5327)
('Virgin Islands region', 19.742)
('Kuril Islands', 47.4082)
('northern Alaska', 67.2067)
('Taiwan region', 24.5347)
('Central California', 38.6627)
('Puerto Rico', 18.2973)
('Puerto Rico region', 19.8519)
('Alaska Peninsula', 58.8165)
('"Papua', -2.0732)
('"Andreanof Islands', 52.8796)
('"Greater Los Angeles area', 34.199)
('"British Columbia', 59.6461)
('Oregon', 44.7133)
('Guatemala', 15.5017)
('"Mona Passage', 18.4724)
('Southeastern Alaska', 59.9897)
('Carlsberg Ridge', -1.8512)
('"Olympic Peninsula', 47.7673)
('"Baja California', 32.5373)
('offshore Northern California', 40.8462)
('Illinois', 37.7136)
('Aegean Sea', 39.7105)
('"Hawaii region', 19.7687)
('Pakistan', 28.9183)
('"Anguilla region', 18.3558)
('Solomon Islands', -10.2935)
('south of the Aleutian Islands', 49.6911)
('north of the Virgin Islands', 19.974)
('"Kodiak Island region', 58.9387)
('"New Britain region', -4.9537)
('"Izu Islands', 32.364)
('Pacific-Antarctic Ridge', -57.0584)
('"Tarapaca', -17.8823)
('"Rat Islands', 51.8203)
('"near the south coast of New Guinea', -7.226)
('"Gulf of Santa Catalina', 33.06)
('"Valparaiso', -33.082)
```

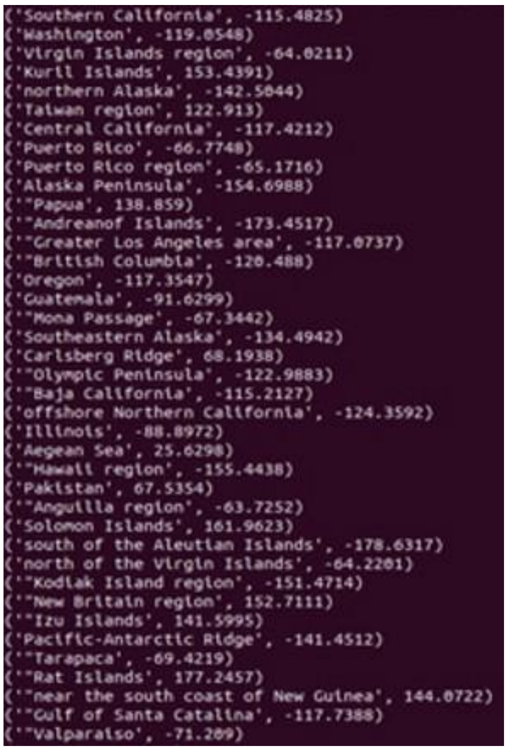


```
('Southern Alaska', 61.9349)
('"Seram', -2.9911)
('Nevada', 40.745)
('Central Alaska', 64.9952)
('South Atlantic Ocean', -53.1139)
('"Kenai Peninsula', 60.9778)
('"off the east coast of Honshu', 37.8174)
('off the coast of Oregon', 43.631)
('Vanuatu', -18.8784)
('western Iran', 31.8467)
('northern Idaho', 47.4516)
('"Island of Hawaii', 19.5123)
('"Fox Islands', 54.115)
('"offshore Chiapas', 14.8825)
('off the coast of Southeastern Alaska', 55.9989)
('"San Francisco Bay area', 38.0647)
('Fiji region', -16.5607)
('"Andaman Islands', 13.349)
('"south of Bali', -9.5025)
('off the west coast of the North Island of New Zealand', -39.31)
('western Montana', 47.9101)
('offshore Honduras', 16.7254)
('Dominican Republic region', 19.4654)
('"northern Sumatra', 4.7537)
('"near the east coast of Honshu', 40.2081)
('offshore Central California', 35.8947)
('Utah', 39.8132)
('Virginia', 38.0255)
('Spain', 37.4918)
('off the west coast of northern Sumatra', 0.7579)
('"Antofagasta', -24.268)
('southern Mid-Atlantic Ridge', -32.382)
('eastern Tennessee', 35.4734)
('Kyrgyzstan', 39.4913)
('"Channel Islands region', 34.0372)
('"Newberry Caldera area', 43.712)
('"Yellowstone National Park', 44.6587)
('"Vancouver Island', 49.0345)
('Strait of Hormuz', 26.8387)
('near the coast of southern Peru', -17.7506)
('"Salta', -24.2873)
('"Halmahera', 1.0963)
('Tonga', -17.1909)
('"Acme Islands', 51.7622)
```

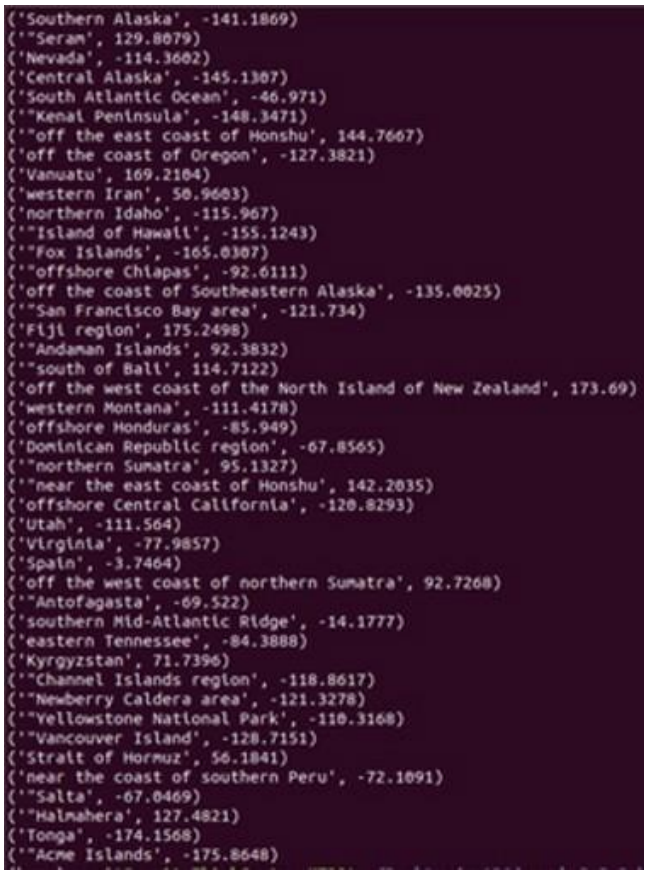
6. **Write a spark program to analyze the given Earthquake data and generate statistics with region and longitude**

```
import sys
if(len(sys.argv)!=3):
        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: (x.split(',')[11],float(x.split(',')[7])))
maxi=temp.reduceByKey(lambda a,b:a if a>b else b)
maxi.saveAsTextFile(sys.argv[2])
```

Output:



```
('Southern California', -115.4825)
('Washington', -119.0548)
('Virgin Islands region', -64.0211)
('Kuril Islands', 153.4391)
('northern Alaska', -142.5044)
('Taiwan region', 122.913)
('Central California', -117.4212)
('Puerto Rico', -66.7748)
('Puerto Rico region', -65.1716)
('Alaska Peninsula', -154.6988)
('"Papua', 138.859)
('"Andreanof Islands', -173.4517)
('"Greater Los Angeles area', -117.0737)
('"British Columbia', -120.488)
('Oregon', -117.3547)
('Guatemala', -91.6299)
('"Mona Passage', -67.3442)
('Southeastern Alaska', -134.4942)
('Carlsberg Ridge', 68.1938)
('"Olympic Peninsula', -122.9883)
('"Baja California', -115.2127)
('offshore Northern California', -124.3592)
('Illinois', -88.8972)
('Aegean Sea', 25.6298)
('"Hawaii region', -155.4438)
('Pakistan', 67.5354)
('"Anguilla region', -63.7252)
('Solomon Islands', 161.9623)
('south of the Aleutian Islands', -178.6317)
('north of the Virgin Islands', -64.2201)
('"Kodiak Island region', -151.4714)
('"New Britain region', 152.7111)
('"Izu Islands', 141.5995)
('Pacific-Antarctic Ridge', -141.4512)
('"Tarapaca', -69.4219)
('"Rat Islands', 177.2457)
('"near the south coast of New Guinea', 144.0722)
('"Gulf of Santa Catalina', -117.7388)
('"Valparaiso', -71.209)
```

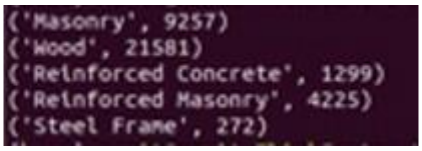


```
('Southern Alaska', -141.1869)
('"Seram', 129.8079)
('Nevada', -114.3602)
('Central Alaska', -145.1307)
('South Atlantic Ocean', -46.971)
('"Kenai Peninsula', -148.3471)
('"off the east coast of Honshu', 144.7667)
('off the coast of Oregon', -127.3821)
('Vanuatu', 169.2104)
('western Iran', 50.9603)
('northern Idaho', -115.967)
('"Island of Hawaii', -155.1243)
('"Fox Islands', -165.0307)
('"offshore Chiapas', -92.6111)
('off the coast of Southeastern Alaska', -135.0025)
('"San Francisco Bay area', -121.734)
('Fiji region', 175.2498)
('"Andaman Islands', 92.3832)
('"south of Bali', 114.7122)
('off the west coast of the North Island of New Zealand', 173.69)
('western Montana', -111.4178)
('offshore Honduras', -85.949)
('Dominican Republic region', -67.8565)
('"northern Sumatra', 95.1327)
('"near the east coast of Honshu', 142.2035)
('offshore Central California', -120.8293)
('Utah', -111.564)
('Virginia', -77.9857)
('Spain', -3.7464)
('off the west coast of northern Sumatra', 92.7268)
('"Antofagasta', -69.522)
('southern Mid-Atlantic Ridge', -14.1777)
('eastern Tennessee', -84.3888)
('Kyrgyzstan', 71.7396)
('"Channel Islands region', -118.8617)
('"Newberry Caldera area', -121.3278)
('"Yellowstone National Park', -110.3168)
('"Vancouver Island', -128.7151)
('Strait of Hormuz', 56.1841)
('near the coast of southern Peru', -72.1091)
('"Salta', -67.0469)
('"Halmahera', 127.4821)
('Tonga', -174.1568)
('"Acme Islands', -175.8648)
```

7. **Write a spark program to analyze the given Insurance data and generate a statistics report with the construction building name and the count of building.**

```
import sys
if(len(sys.argv)!=3):
        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: (x.split(',')[16],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

Output:

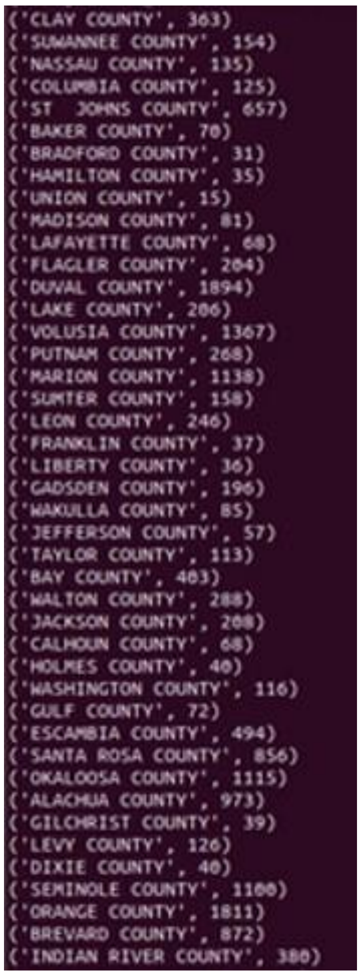


```
('Masonry', 9257)
('Wood', 21581)
('Reinforced Concrete', 1299)
('Reinforced Masonry', 4225)
('Steel Frame', 272)
```

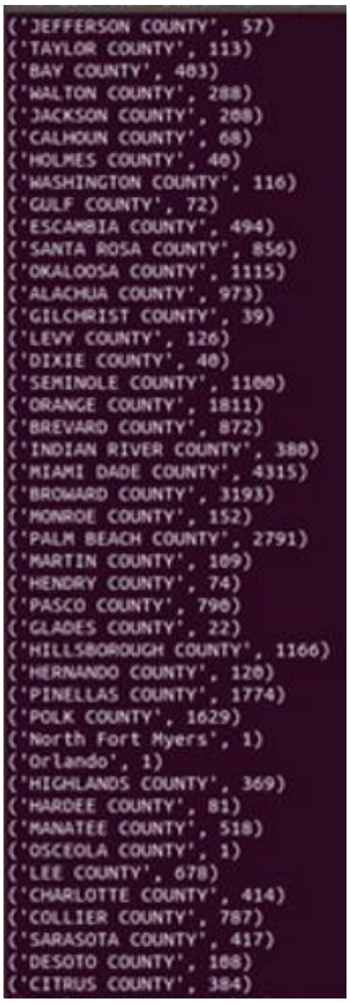
8. **Write a spark program to analyze the given Insurance data and generate a statistics report with the county name and its frequency.**

```
import sys
if(len(sys.argv)!=3):
	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: (x.split(',')[2],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

Output:



```
('CLAY COUNTY', 363)
('SUWANNEE COUNTY', 154)
('NASSAU COUNTY', 135)
('COLUMBIA COUNTY', 125)
('ST  JOHNS COUNTY', 657)
('BAKER COUNTY', 70)
('BRADFORD COUNTY', 31)
('HAMILTON COUNTY', 35)
('UNION COUNTY', 15)
('MADISON COUNTY', 81)
('LAFAYETTE COUNTY', 68)
('FLAGLER COUNTY', 204)
('DUVAL COUNTY', 1894)
('LAKE COUNTY', 206)
('VOLUSIA COUNTY', 1367)
('PUTNAM COUNTY', 268)
('MARION COUNTY', 1138)
('SUMTER COUNTY', 158)
('LEON COUNTY', 246)
('FRANKLIN COUNTY', 37)
('LIBERTY COUNTY', 36)
('GADSDEN COUNTY', 196)
('WAKULLA COUNTY', 85)
('JEFFERSON COUNTY', 57)
('TAYLOR COUNTY', 113)
('BAY COUNTY', 403)
('WALTON COUNTY', 288)
('JACKSON COUNTY', 208)
('CALHOUN COUNTY', 68)
('HOLMES COUNTY', 40)
('WASHINGTON COUNTY', 116)
('GULF COUNTY', 72)
('ESCAMBIA COUNTY', 494)
('SANTA ROSA COUNTY', 856)
('OKALOOSA COUNTY', 1115)
('ALACHUA COUNTY', 973)
('GILCHRIST COUNTY', 39)
('LEVY COUNTY', 126)
('DIXIE COUNTY', 40)
('SEMINOLE COUNTY', 1100)
('ORANGE COUNTY', 1811)
('BREVARD COUNTY', 872)
('INDIAN RIVER COUNTY', 380)
```



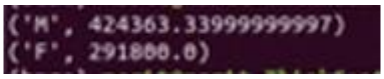
```
('JEFFERSON COUNTY', 57)
('TAYLOR COUNTY', 113)
('BAY COUNTY', 403)
('WALTON COUNTY', 288)
('JACKSON COUNTY', 208)
('CALHOUN COUNTY', 68)
('HOLMES COUNTY', 40)
('WASHINGTON COUNTY', 116)
('GULF COUNTY', 72)
('ESCAMBIA COUNTY', 494)
('SANTA ROSA COUNTY', 856)
('OKALOOSA COUNTY', 1115)
('ALACHUA COUNTY', 973)
('GILCHRIST COUNTY', 39)
('LEVY COUNTY', 126)
('DIXIE COUNTY', 40)
('SEMINOLE COUNTY', 1100)
('ORANGE COUNTY', 1811)
('BREVARD COUNTY', 872)
('INDIAN RIVER COUNTY', 380)
('MIAMI DADE COUNTY', 4315)
('BROWARD COUNTY', 3193)
('MONROE COUNTY', 152)
('PALM BEACH COUNTY', 2791)
('MARTIN COUNTY', 109)
('HENDRY COUNTY', 74)
('PASCO COUNTY', 790)
('GLADES COUNTY', 22)
('HILLSBOROUGH COUNTY', 1166)
('HERNANDO COUNTY', 120)
('PINELLAS COUNTY', 1774)
('POLK COUNTY', 1629)
('North Fort Myers', 1)
('Orlando', 1)
('HIGHLANDS COUNTY', 369)
('HARDEE COUNTY', 81)
('MANATEE COUNTY', 518)
('OSCEOLA COUNTY', 1)
('LEE COUNTY', 678)
('CHARLOTTE COUNTY', 414)
('COLLIER COUNTY', 787)
('SARASOTA COUNTY', 417)
('DESOTO COUNTY', 108)
('CITRUS COUNTY', 384)
```

9. **Write a map-reduce program to analyze the given employee record data and generate a statistics report with the total Sales for female and male employees**

```
import sys
if(len(sys.argv)!=3):
	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: (x.split('\t')[3],float(x.split('\t')[8])))
total=temp.reduceByKey(lambda a,b : a+b)
total.saveAsTextFile(sys.argv[2])
```

Output:

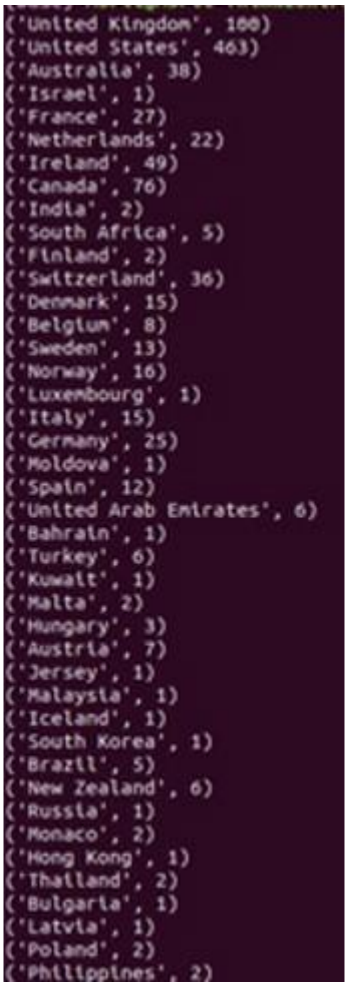


```
('M', 424363.33999999997)
('F', 291800.0)
```

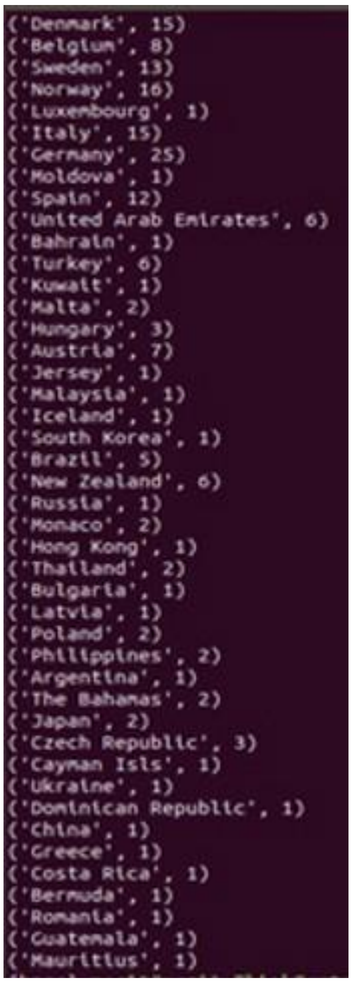
10. **Write a map-reduce program to analyze the given sales records over a period and generate data about the country's total sales, and the total number of the products**

```
import sys
if(len(sys.argv)!=3):
        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: (x.split(',')[7],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

Output:



```
('United Kingdom', 100)
('United States', 463)
('Australia', 38)
('Israel', 1)
('France', 27)
('Netherlands', 22)
('Ireland', 49)
('Canada', 76)
('India', 2)
('South Africa', 5)
('Finland', 2)
('Switzerland', 36)
('Denmark', 15)
('Belgium', 8)
('Sweden', 13)
('Norway', 16)
('Luxembourg', 1)
('Italy', 15)
('Germany', 25)
('Moldova', 1)
('Spain', 12)
('United Arab Emirates', 6)
('Bahrain', 1)
('Turkey', 6)
('Kuwait', 1)
('Malta', 2)
('Hungary', 3)
('Austria', 7)
('Jersey', 1)
('Malaysia', 1)
('Iceland', 1)
('South Korea', 1)
('Brazil', 5)
('New Zealand', 6)
('Russia', 1)
('Monaco', 2)
('Hong Kong', 1)
('Thailand', 2)
('Bulgaria', 1)
('Latvia', 1)
('Poland', 2)
('Philippines', 2)
```

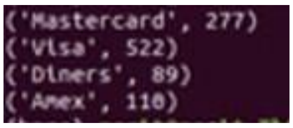


```
('Denmark', 15)
('Belgium', 8)
('Sweden', 13)
('Norway', 16)
('Luxembourg', 1)
('Italy', 15)
('Germany', 25)
('Moldova', 1)
('Spain', 12)
('United Arab Emirates', 6)
('Bahrain', 1)
('Turkey', 6)
('Kuwait', 1)
('Malta', 2)
('Hungary', 3)
('Austria', 7)
('Jersey', 1)
('Malaysia', 1)
('Iceland', 1)
('South Korea', 1)
('Brazil', 5)
('New Zealand', 6)
('Russia', 1)
('Monaco', 2)
('Hong Kong', 1)
('Thailand', 2)
('Bulgaria', 1)
('Latvia', 1)
('Poland', 2)
('Philippines', 2)
('Argentina', 1)
('The Bahamas', 2)
('Japan', 2)
('Czech Republic', 3)
('Cayman Isls', 1)
('Ukraine', 1)
('Dominican Republic', 1)
('China', 1)
('Greece', 1)
('Costa Rica', 1)
('Bermuda', 1)
('Romania', 1)
('Guatemala', 1)
('Mauritius', 1)
```

11. **Write a map-reduce program to analyze the given sales records over a period of time and generate data about the country’s total sales and the frequency of the payment mode.**

```
import sys
if(len(sys.argv)!=3):
        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: (x.split(',')[3],1))
data=temp.countByKey()
dd=sc.parallelize(data.items())
dd.saveAsTextFile(sys.argv[2])
```

Output:



```
('Mastercard', 277)
('Visa', 522)
('Diners', 89)
('Amex', 110)
```

PIG PROGRAMS

<u>Bash file:</u>

```
export JAVA_HOME=$(readlink -f $(which javac) | awk 'BEGIN {FS="/bin"} {print $1}')
if ! command -v pig &> /dev/null
then
export PATH=$(echo $PATH):$(pwd)/bin
fi
```

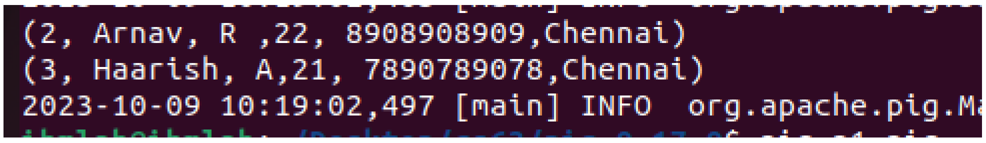
Command to run it: **source bash.sh**

Programs:

1. Create a program to **filter** out student data based on City Chennai

   student_details.txt:
   ```
   001,Rajiv,Reddy,21,9848022337,Hyderabad
   002,Arnav,R,22,8908908909,Chennai
   003,Haarish,A,21,7890789078,Chennai
   004,Preethi,Agarwal,21,9848022330,Pune
   005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
   006,Archana,Mishra,23,9848022335,Chennai
   007,Komal,Nayak,24,9848022334,trivendram
   008,Bharathi,Nambiayar,24,9848022333,Chennai
   ```

   ```
   student_details = LOAD '/home/msrit/Downloads/pig/test/Filter/student_details.txt'
   USING
   PigStorage(',') as (id:int, firstname:chararray, lastname:chararray, age:int,
   phone:chararray,
   city:chararray);
   filter_data = FILTER student_details BY city == 'Chennai';
   Dump filter_data;
   ```

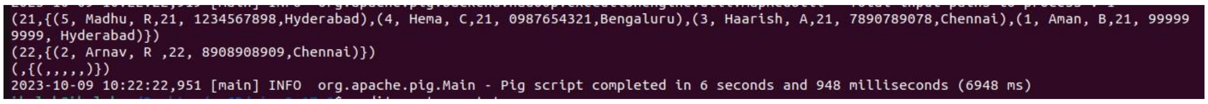
   

   ```
   (2, Arnav, R ,22, 8908908909,Chennai)
   (3, Haarish, A,21, 7890789078,Chennai)
   2023-10-09 10:19:02,497 [main] INFO  org.apache.pig.M
   ```

2. Create a program to **group** student data based on age

   ```
   student = LOAD 'student_details.txt' USING PigStorage(',') as (id:int,
   firstname:chararray,
   lastname:chararray, age:int, phone:chararray, city:chararray);
   ```

group_data = GROUP student by age;
Dump group_data;



```
(21,{(5, Madhu, R,21, 1234567898,Hyderabad),(4, Hema, C,21, 0987654321,Bengaluru),(3, Haarish, A,21, 7890789078,Chennai),(1, Aman, B,21, 99999
9999, Hyderabad)})
(22,{(2, Arnav, R ,22, 8908908909,Chennai)})
(,{(,,,,,)})
2023-10-09 10:22:22,951 [main] INFO  org.apache.pig.Main - Pig script completed in 6 seconds and 948 milliseconds (6948 ms)
```

3. Create a program to **Join** two separate data files of custome.txt and order.txt based id and order by id.

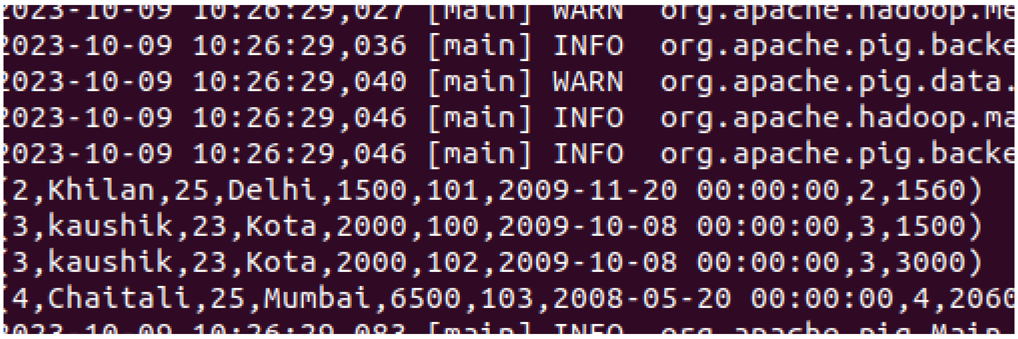
Customer.txt

1,Ramesh,32,Ahmedabad,2000.00
2,Khilan,25,Delhi,1500.00
3,kaushik,23,Kota,2000.00
4,Chaitali,25,Mumbai,6500.00
5,Hardik,27,Bhopal,8500.00
6,Komal,22,MP,4500.00
7,Muffy,24,Indore,10000.00

Order.txt

102,2009-10-08 00:00:00,3,3000
100,2009-10-08 00:00:00,3,1500
101,2009-11-20 00:00:00,2,1560
103,2008-05-20 00:00:00,4,2060

customers = LOAD 'customer.txt' USING PigStorage(',') as (id:int, name:chararray, age:int,
address:chararray, salary:int);
orders = LOAD 'order.txt' USING PigStorage(',') as (oid:int, date:chararray, customer_id:int,
amount:int);
join_result = JOIN customers BY id, orders BY customer_id;
Dump join_result;



```
2023-10-09 10:26:29,036 [main] INFO  org.apache.pig.backe
2023-10-09 10:26:29,040 [main] WARN  org.apache.pig.data.
2023-10-09 10:26:29,046 [main] INFO  org.apache.hadoop.ma
2023-10-09 10:26:29,046 [main] INFO  org.apache.pig.backe
(2,Khilan,25,Delhi,1500,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060
```

4. Create a program to obtain **union** of customer1 and customer2 dataset

Customer1.txt

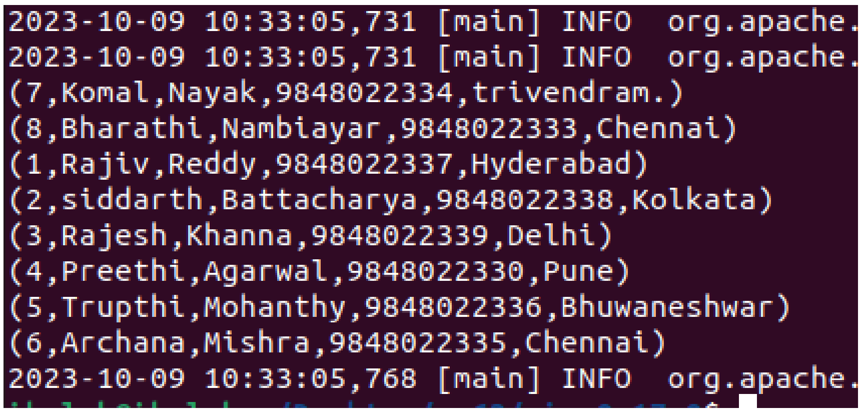
001,Rajiv,Reddy,9848022337,Hyderabad

002,siddarth,Battacharya,9848022338,Kolkata
003,Rajesh,Khanna,9848022339,Delhi
004,Preethi,Agarwal,9848022330,Pune
005,Trupthi,Mohanthy,9848022336,Bhuwaneshwar
006,Archana,Mishra,9848022335,Chennai.

Customer2.txt

7,Komal,Nayak,9848022334,trivendram.
8,Bharathi,Nambiayar,9848022333,Chennai.

```
cust1 = LOAD 'customer1.txt' USING PigStorage(',') as (id:int, firstname:chararray,
lastname:chararray, phone:chararray, city:chararray);
cust2 = LOAD 'customer2.txt' USING PigStorage(',') as (id:int, firstname:chararray,
lastname:chararray, phone:chararray, city:chararray);
cust = UNION cust1, cust2;
Dump cust;
```



```
2023-10-09 10:33:05,731 [main] INFO  org.apache.
2023-10-09 10:33:05,731 [main] INFO  org.apache.
(7,Komal,Nayak,9848022334,trivendram.)
(8,Bharathi,Nambiayar,9848022333,Chennai)
(1,Rajiv,Reddy,9848022337,Hyderabad)
(2,siddarth,Battacharya,9848022338,Kolkata)
(3,Rajesh,Khanna,9848022339,Delhi)
(4,Preethi,Agarwal,9848022330,Pune)
(5,Trupthi,Mohanthy,9848022336,Bhuwaneshwar)
(6,Archana,Mishra,9848022335,Chennai)
2023-10-09 10:33:05,768 [main] INFO  org.apache.
```