**Department of Computer Science & Engineering**

**CSL76: BIGDATA LABORATORY TERM: Sep2022- Jan2022**

**HADOOP REPORT**

***By***

**Manoj Kumar M 1MS19CS068**

**Submitted to**

**Pradeep Kumar. D Assistant Professor**

**M S RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[**www.msrit.edu**](http://www.msrit.edu/)

2022

**Hadoop Installation**

1. Download Hadoop file from the following link from any browser
2. https://dlcdn.apache.org/hadoop/common/hadoop-3.3.2/hadoop-3.3.2.tar.gz
3. Create a new folder inside Desktop, name the folder with USN as file name “1MS19CS086”
4. Move downloaded tar.gz file to Desktop/1MS19CS086
5. Right-click on the tar.gz file and extract it there.
6. Open terminal and navigate to Desktop/1MS19CS086/hadoop-3.3.2/
7. Create Bash.sh file using - vi Bash.sh and add the following code.
8. Paste the following block of code to Bash.sh file :
9. Execute bash.sh file using the command - source Bash.sh
10. Verify JAVA\_HOME variable is set to Java Path and CLASSPATH variable has the
11. USN Hadoop Folder by executing – echo $JAVA\_HOME

echo $CLASSPATH

1. Verify Hadoop's installation by executing hadoop command.

**Hadoop Execution**

1. Create a Test folder and move to that folder
2. Make the driver.java , mapper.java and reducer.java files
3. Compile all java files (driver.java mapper.java reducer.java)
   * javac -d . \*.java
4. Set driver class in manifest
5. echo Main-Class: Test.driver > Manifest.txt
6. Create an executable jar file
7. jar cfm Test.jar Manifest.txt Test/\*.class
8. Run the jar file
9. hadoop jar wordcount.jar data.txt output
   * Where data.txt is the input file
10. To see the Output
11. cat output/\*

# 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:**

0067011990999991950051507004+68750+023550FM-

12+038299999V0203301N00671220001CN9999999N9+00001+99999999999

0043011990999991950051512004+68750+023550FM-

12+038299999V0203201N00671220001CN9999999N9+00221+99999999999

0043011990999991950051518004+68750+023550FM-

12+038299999V0203201N00261220001CN9999999N9-00111+99999999999

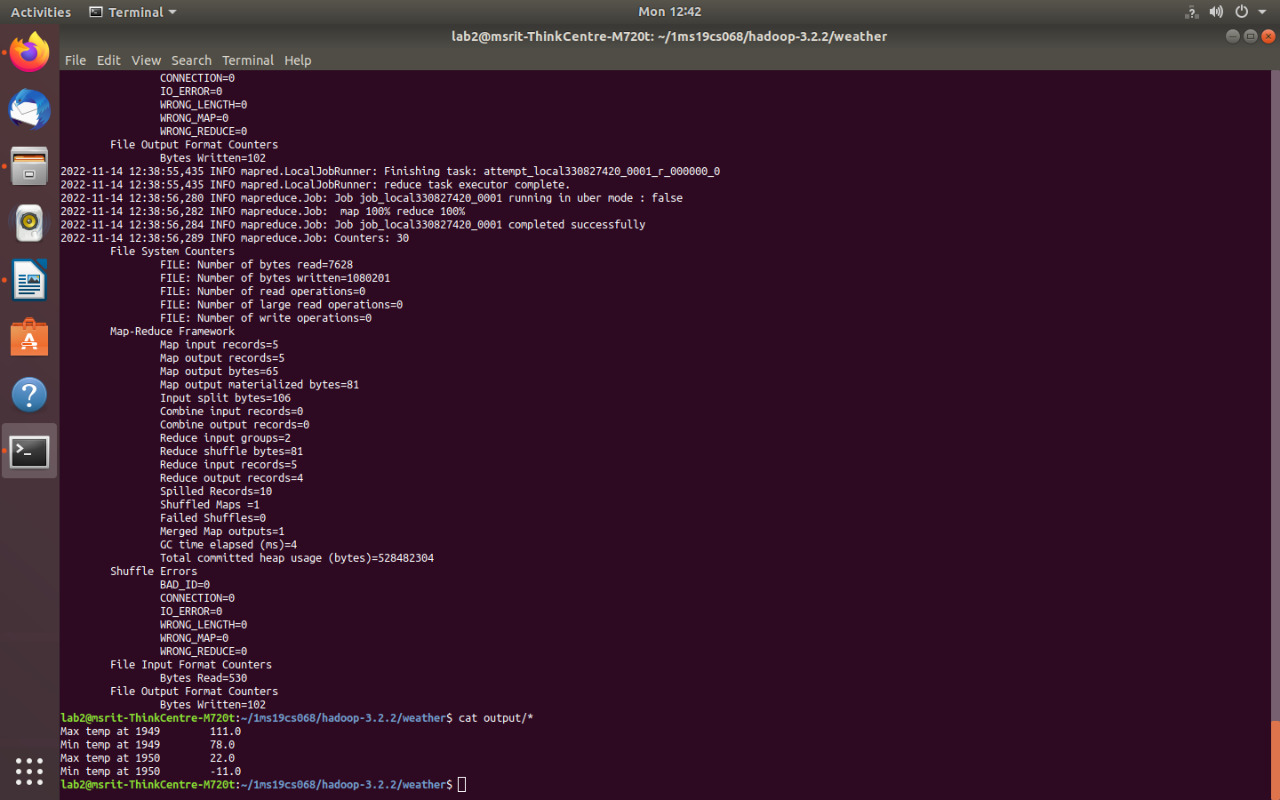
0043012650999991949032412004+62300+010750FM-

12+048599999V0202701N00461220001CN0500001N9+01111+99999999999

0043012650999991949032418004+62300+010750FM-

12+048599999V0202701N00461220001CN0500001N9+00781+99999999999

**Output:**



1. **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) 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);

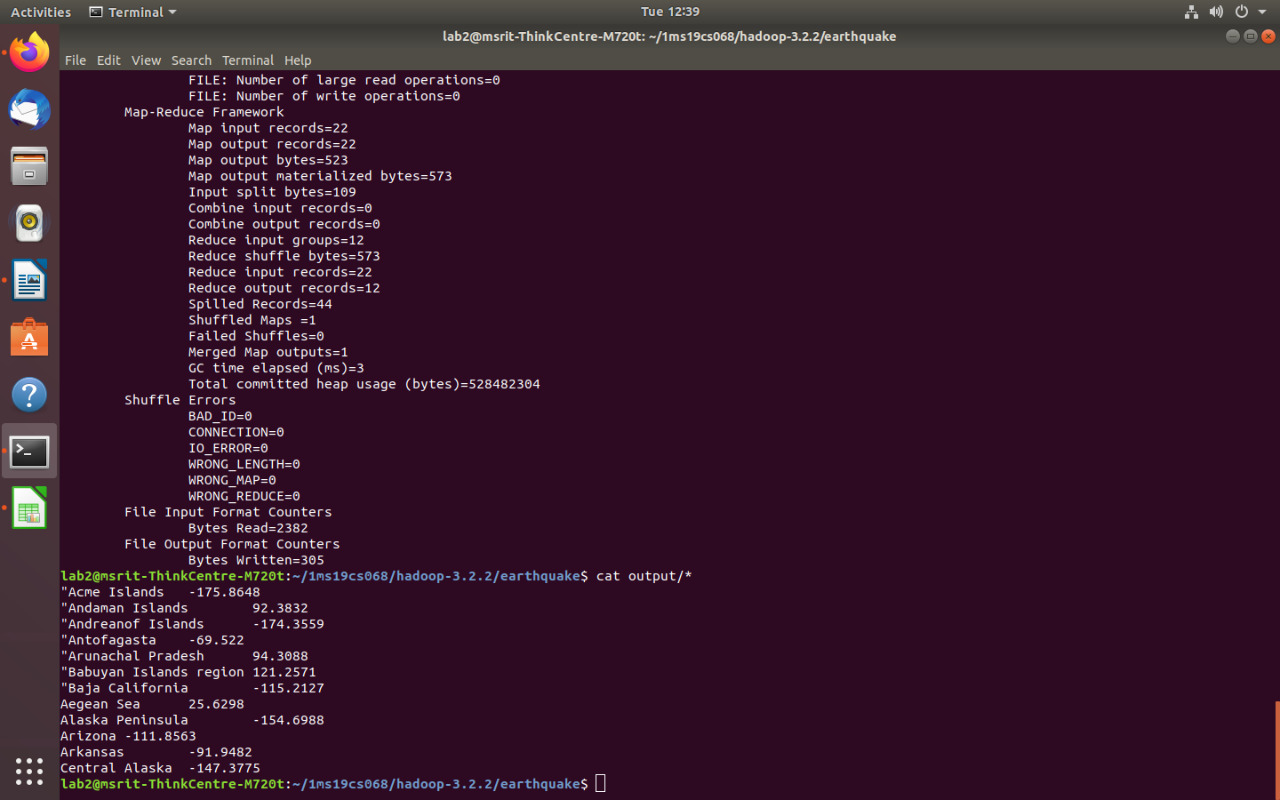
}

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

}

}

**Output:**



1. **Write a MapReduce program 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) {

}

else{

}

}

}

}

output.collect(new Text("even"),new IntWritable(number));

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));

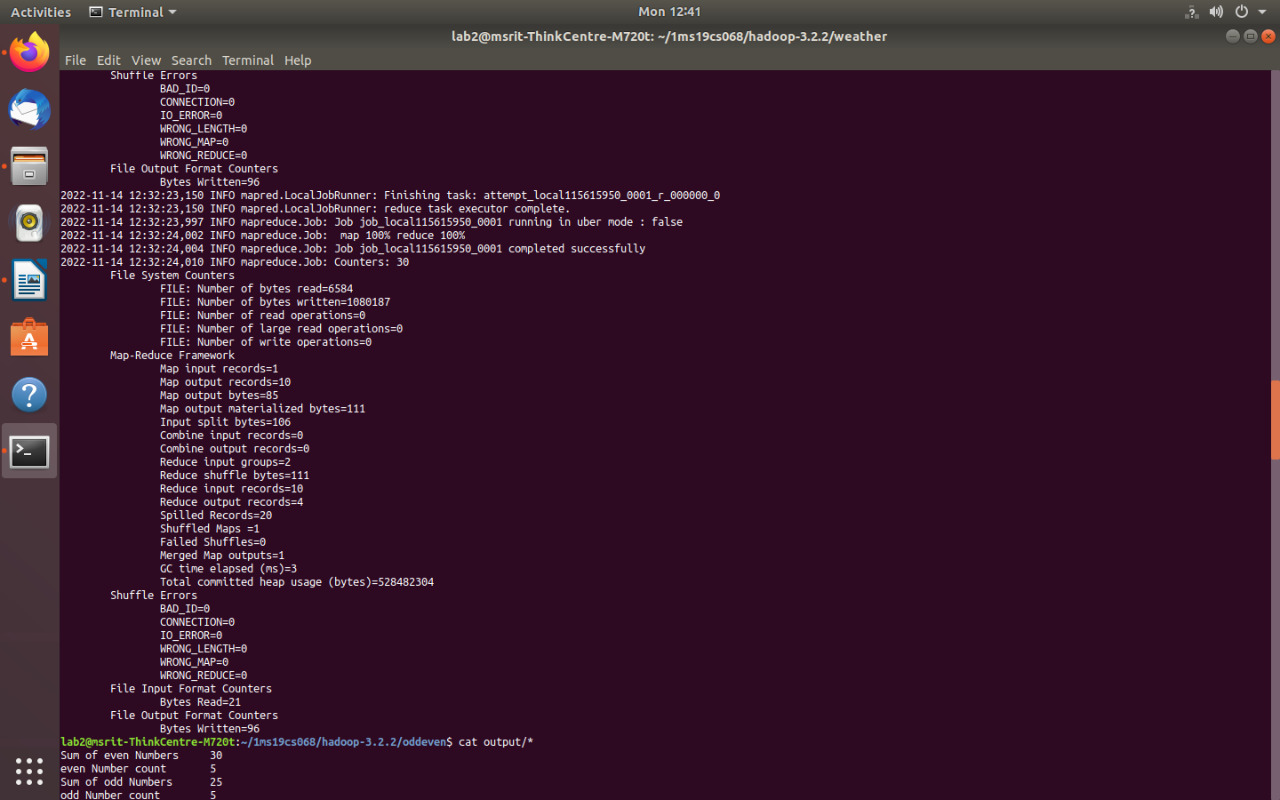
}

}

**Input:**

1 2 3 4 5 6 7 8 9 10

**Output:**



# 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();

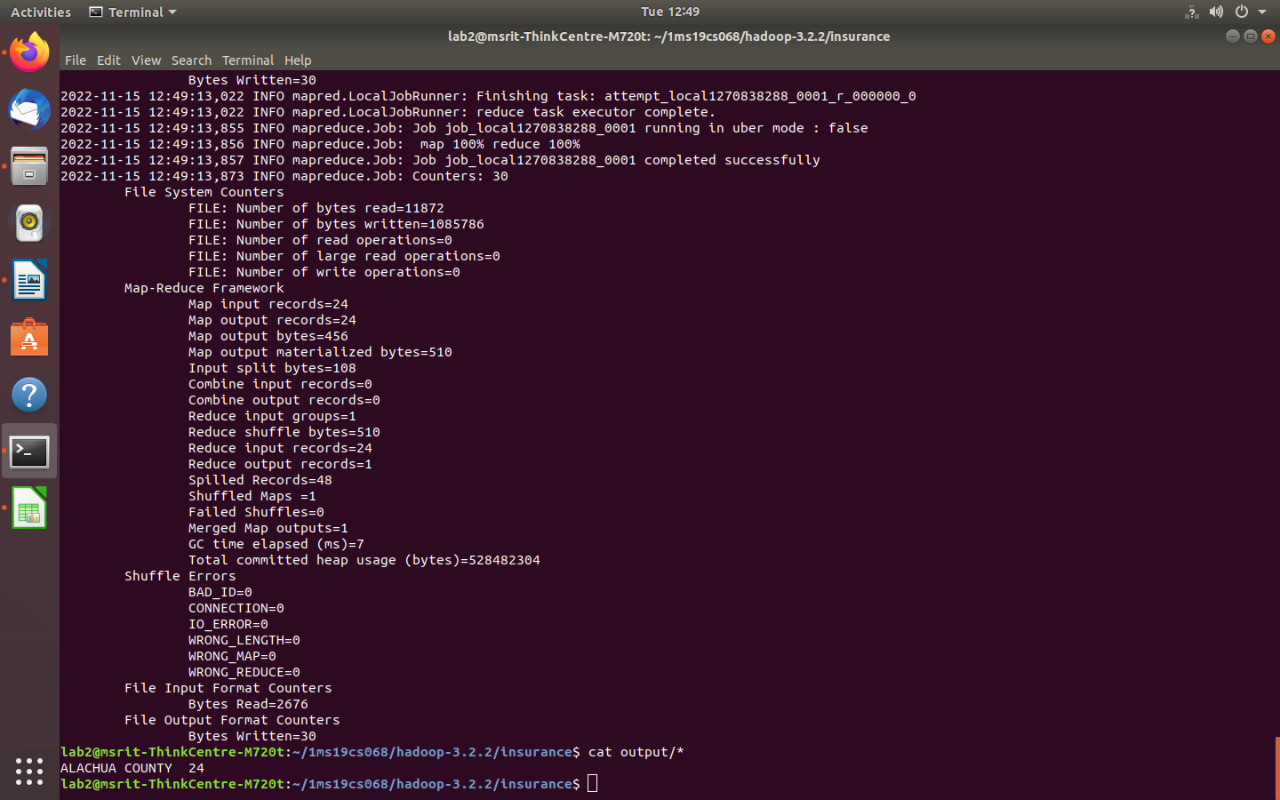
}

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

}

}

**Output:**



# Write a MapReduce program 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));

}}

**redducer.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;

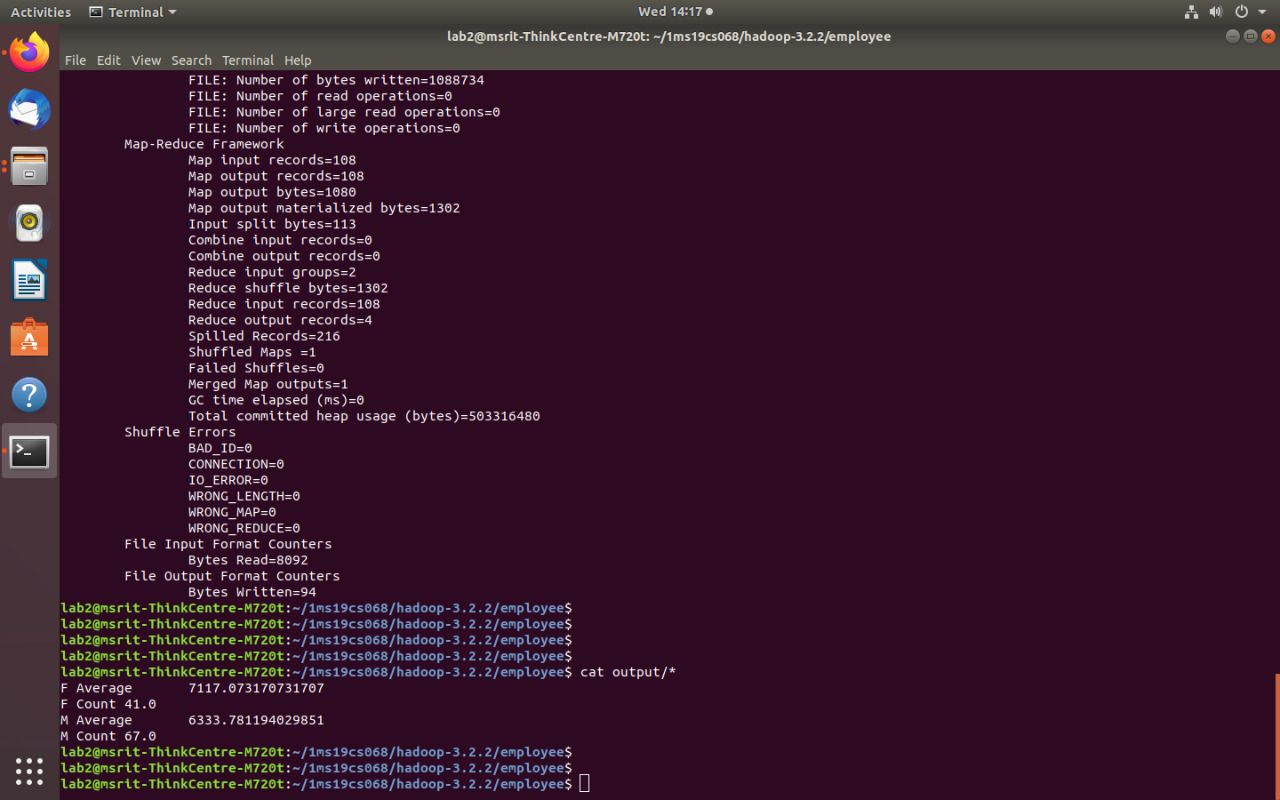
}

output.collect(new Text(key+" Average"), new DoubleWritable(sum/count)); output.collect(new Text(key+" Count"), new DoubleWritable(count));

}

}

**Output:**



# 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 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

{

}}

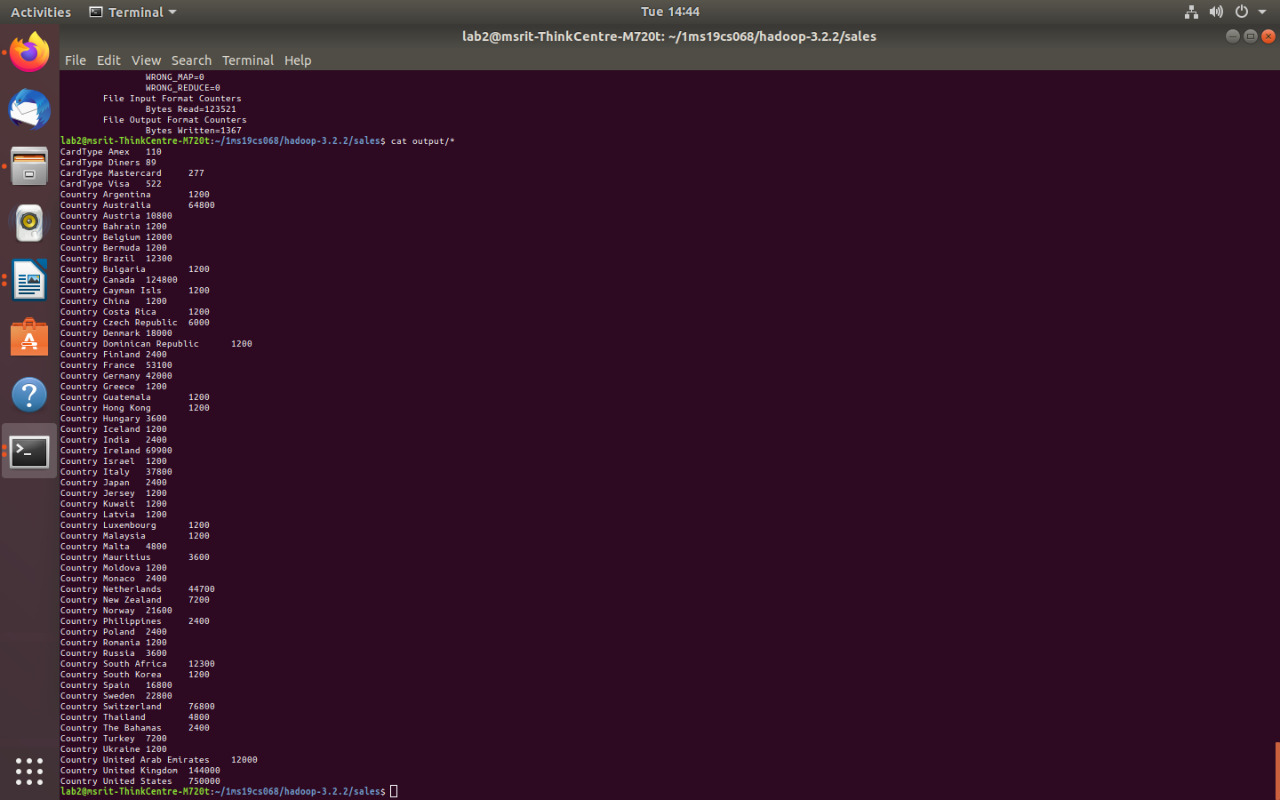
int sum=0; while(value.hasNext())

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

}

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

**Output:**



1. **Matrix multiplication using MapReduce**

**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"))

{

}

else

{

}

}

a.put(Integer.parseInt(v[1]),Float.parseFloat(v[2]));

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:**

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:**



1. **Word Count using Map Reduce 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:**

HDFS is a storage unit of Hadoop MapReduce is a processing tool of Hadoop **Output:**

