BIG DATA ENGINEERING FOR HADOOP & SPARK TRAINING

ASSIGNMENT 4

Task 1:

Write a Map Reduce program to filter out the invalid records. Map only job will fit for this context.

Sol : As per the given task the Input file or Dataset will be as follows

The fields are arranged like:

Company Name|Product Name|Size in inches|State|Pin Code|Price

Television.txt

```
Samsung|Optima|14|Madhya Pradesh|132401|14200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Akai|Decent|16|Kerala|922401|12200
Lava|Attention|20|Assam|454601|24200
Zen|Super|14|Maharashtra|619082|9200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Onida|Decent|14|Uttar Pradesh|232401|16200
Onida|NA|16|Kerala|922401|12200
Lava|Attention|20|Assam|454601|24200
Zen|Super|14|Maharashtra|619082|9200
Samsung|Optima|14|Madhya Pradesh|132401|14200
NA|Lucid|18|Uttar Pradesh|232401|16200
```

Samsung|Decent|16|Kerala|922401|12200

Lava|Attention|20|Assam|454601|24200

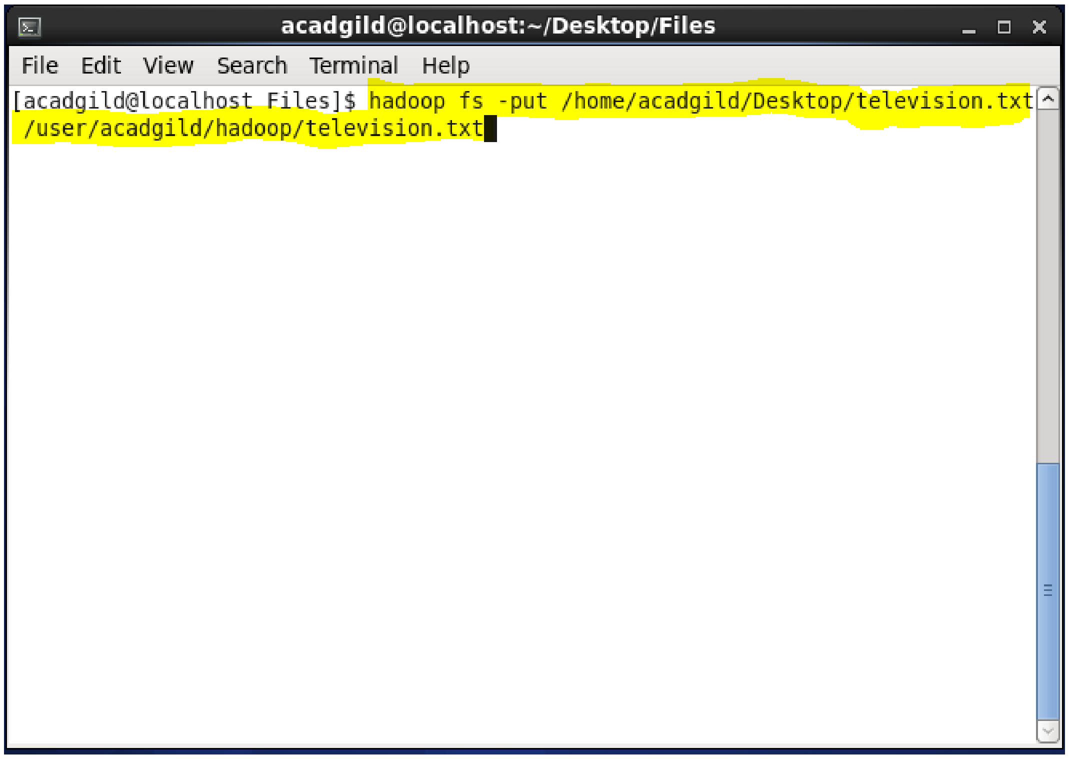
Samsung|Super|14|Maharashtra|619082|9200

Samsung|Super|14|Maharashtra|619082|9200

Samsung|Super|14|Maharashtra|619082|9200

To copy the file to hadoop file system we use

hadoop fs –put <local_source_dir>/filename <hadoop_destination_dir>/filename


acadgild@localhost:~/Desktop/Files
File Edit View Search Terminal Help

```
[acadgild@localhost Files]$ hadoop fs -put /home/acadgild/Desktop/television.txt
 /user/acadgild/hadoop/television.txt
```

Now we need to write a Map only Job for the given task as there is no operation to be performed on the output of Mapper class just we have to remove the “NA” records.

Driver Class Program

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

public class MapOnlyDriver
{
 public static void main( String[] args ) throws IOException,
ClassNotFoundException, InterruptedException
{
Configuration conf = new Configuration();
Job job = new ~~Job~~(conf, "Mapper_Only_Job");

job.setJarByClass(MapOnlyDriver.class);
job.setMapperClass(MapOnlyMapper.class);

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

job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

// Sets reducer tasks to 0 as we don't require reducer for this task
job.setNumReduceTasks(0);

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

boolean result = job.waitForCompletion(true);

System.exit(result ? 0 : 1);
}
}
```

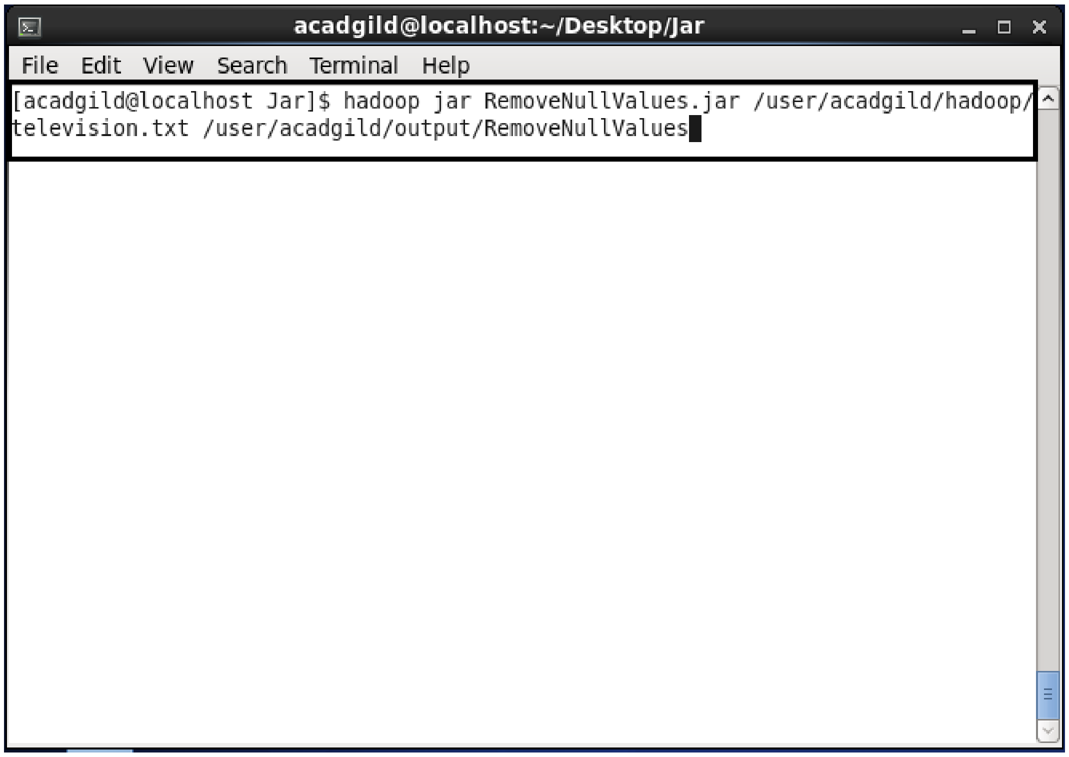
Mapper Class Program

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MapOnlyMapper extends Mapper<LongWritable, Text, Text, Text>
{

private Text word = new Text();

public void map(LongWritable key, Text value, Context context)
             throws IOException, InterruptedException {
        String line = value.toString();
        int count=0;

        //It divides the line into tokens with the delimiter "|"
        StringTokenizer tokenizer = new StringTokenizer(line,"|");

        while (tokenizer.hasMoreTokens())
        {
            word.set(tokenizer.nextToken());
            if(word.toString().equalsIgnoreCase("NA"))
            {

               count=count+1;
            }

        }
        if(count==0)
        {
        Text t = new Text(line);
        context.write(t,null);
        }
}
}
```

As we require only Driver and mapper jobs so we can create the Jar file for this project and execute the Jar on the Dataset.

Executing the Jar on the Dataset we use the command

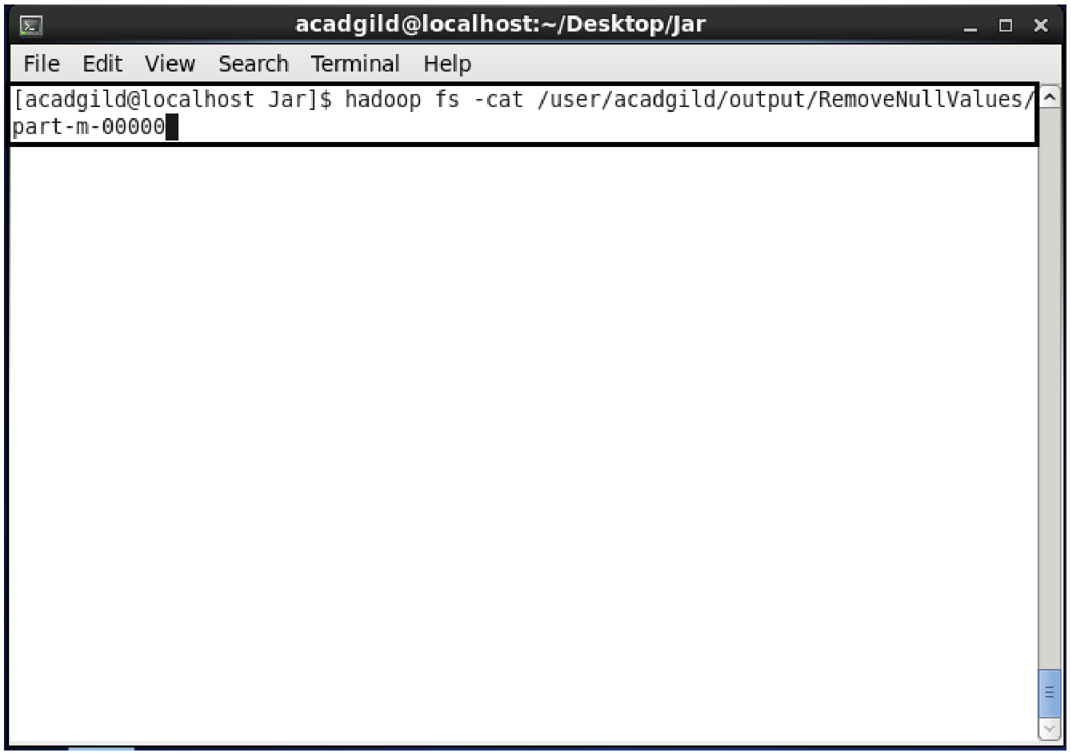
“hadoop jar RemoveNullValues.jar /user/acadgild/hadoop/television.txt /user/acadgild/output/RemoveNullValues”


acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

```
[acadgild@localhost Jar]$ hadoop jar RemoveNullValues.jar /user/acadgild/hadoop/
television.txt /user/acadgild/output/RemoveNullValues
```

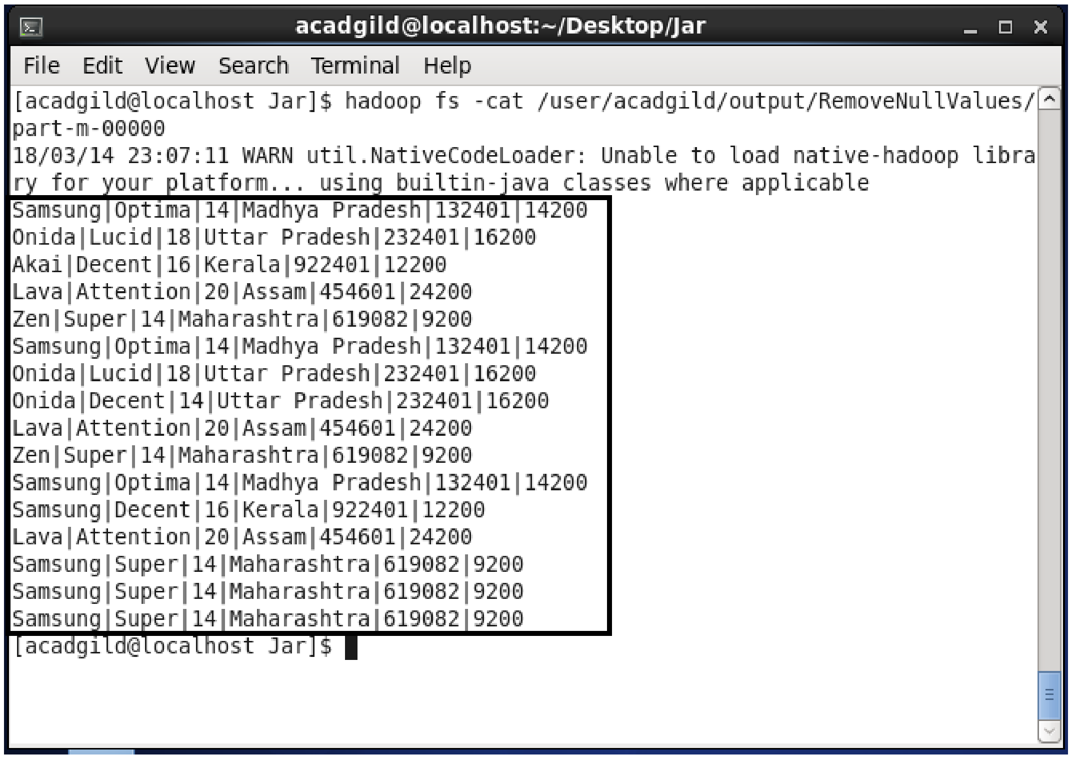
After the successful execution of the program we need to check the output which will be records that contain “NA” values will be removed.

“hadoop fs –cat /user/acadgild/output/RemoveNullValues/part-m-00000”



acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

```
[acadgild@localhost Jar]$ hadoop fs -cat /user/acadgild/output/RemoveNullValues/
part-m-00000
```



acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

```
[acadgild@localhost Jar]$ hadoop fs -cat /user/acadgild/output/RemoveNullValues/
part-m-00000
18/03/14 23:07:11 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Samsung|Optima|14|Madhya Pradesh|132401|14200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Akai|Decent|16|Kerala|922401|12200
Lava|Attention|20|Assam|454601|24200
Zen|Super|14|Maharashtra|619082|9200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Onida|Decent|14|Uttar Pradesh|232401|16200
Lava|Attention|20|Assam|454601|24200
Zen|Super|14|Maharashtra|619082|9200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Samsung|Decent|16|Kerala|922401|12200
Lava|Attention|20|Assam|454601|24200
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
[acadgild@localhost Jar]$
```

Task 2:

Write a Map Reduce program to calculate the total units sold for each Company.

Sol : Here we have to write Mapper as well as Reducer class programs because after segregation we have to aggregate to get the final output so that we get the total units sold for each company.

Driver Class Program

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

public class TotalSoldUnitsDriver {

public static void main(String[] args) throws Exception {
if (args.length != 2) {
System.err.println("Usage: TotalSoldUnits <input path> <output path>");
System.exit(-1);
}

//Job Related Configurations
Configuration conf = new Configuration();
Job job = new ~~Job~~(conf, "My TotalSoldUnits with combiner");
job.setJarByClass(TotalSoldUnitsDriver.class);


// Specify the number of reducer to 2
 //job.setNumReduceTasks(2);

 //Provide paths to pick the input file for the job
FileInputFormat.setInputPaths(job, new Path(args[0]));


 //Provide paths to pick the output file for the job, and delete it if already
present
Path outputPath = new Path(args[1]);
FileOutputFormat.setOutputPath(job, outputPath);
outputPath.getFileSystem(conf).delete(outputPath, true);
//To set the mapper and reducer of this job
job.setMapperClass(TotalSoldUnitsMapper.class);
job.setReducerClass(TotalSoldUnitsReducer.class);

//Set the combiner
job.setCombinerClass(TotalSoldUnitsReducer.class);

//set the input and output format class
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

//set up the output key and value classes
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

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

Mapper Class Program for Seggregation

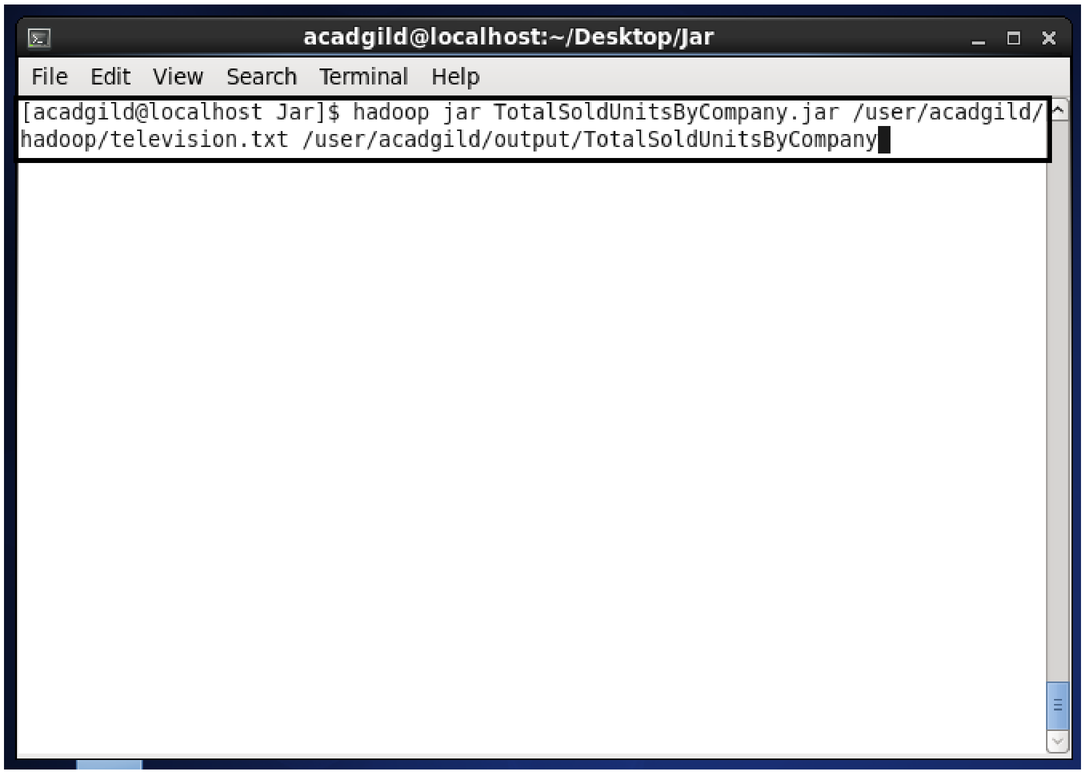
```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.util.*;


public class TotalSoldUnitsMapper

  extends Mapper<LongWritable, Text, Text, IntWritable> {


  private final static IntWritable one = new IntWritable(1);

  @Override

  public void map(LongWritable key, Text value, Context context)

      throws IOException, InterruptedException {

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

               Text t1 = new Text(line[0]);

               context.write(t1, one);

  }

}
```

Reducer Class Program for Aggregation

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class TotalSoldUnitsReducer

  extends Reducer<Text, IntWritable, Text, IntWritable> {

  @Override

  public void reduce(Text key, Iterable<IntWritable> values,

      Context context)

      throws IOException, InterruptedException {

      System.out.println("From The Reducer=>"+key) ;

      int sum = 0;

      for (IntWritable value : values) {

                        sum+=value.get();

       }

       context.write(key, new IntWritable(sum));

  }

}
```

Executing the Jar on the Dataset we use the command

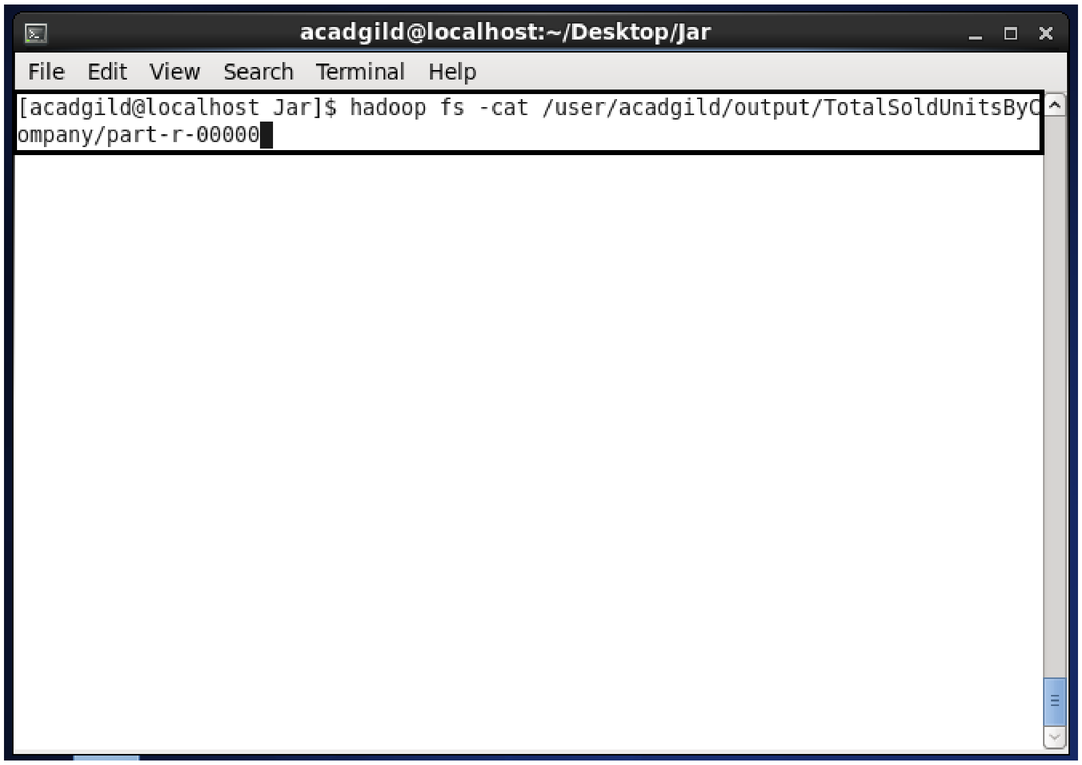
“hadoop jar TotalSoldUnitsByCompany.jar /user/acadgild/hadoop/television.txt /user/acadgild/output/ TotalSoldUnitsByCompany”


acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

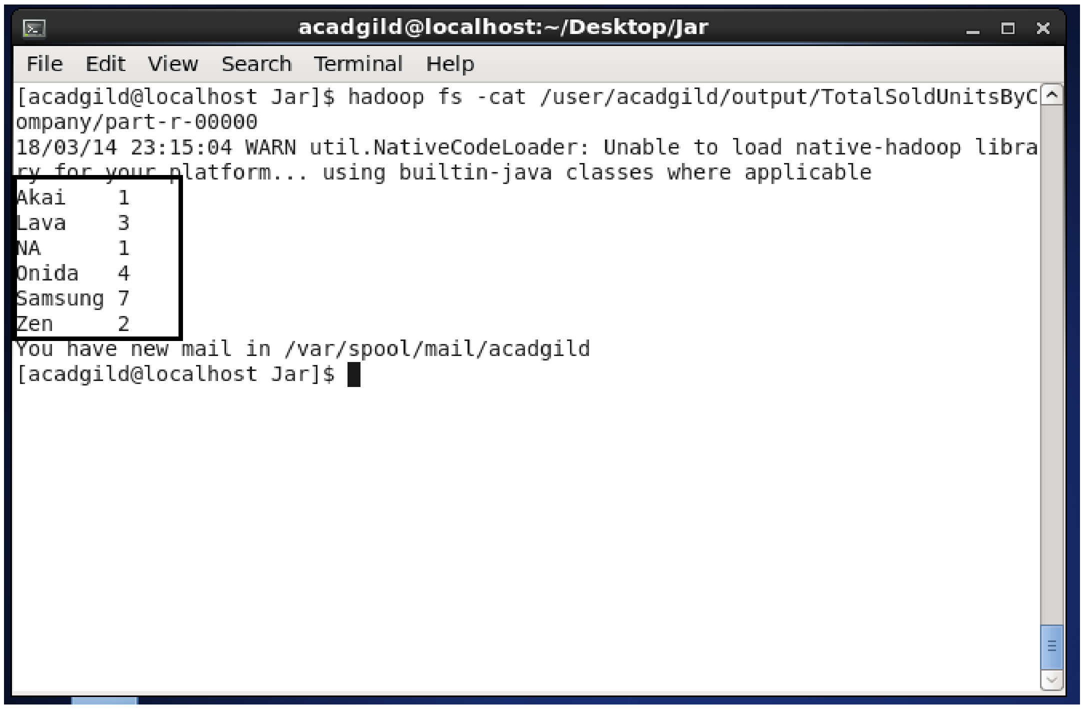
```
[acadgild@localhost Jar]$ hadoop jar TotalSoldUnitsByCompany.jar /user/acadgild/
hadoop/television.txt /user/acadgild/output/TotalSoldUnitsByCompany
```

After Successful execution of the Map Reduce Job on the Dataset we should see the output with total units sold for each company.

“hadoop fs –cat /user/acadgild/output/TotalSoldUnitsBy Comapny/part-r-00000”


acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

```
[acadgild@localhost Jar]$ hadoop fs -cat /user/acadgild/output/TotalSoldUnitsByC
ompany/part-r-00000
```


acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

```
[acadgild@localhost Jar]$ hadoop fs -cat /user/acadgild/output/TotalSoldUnitsByC
ompany/part-r-00000
18/03/14 23:15:04 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Akai    1
Lava    3
NA      1
Onida   4
Samsung 7
Zen     2
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Jar]$
```

Task 3:

Write a Map Reduce program to calculate the total units sold in each state for Onida company.

Sol: Here we have to calculate Onida company sales in each state.

Here also we require Mapper as well as Reducer class programs because after segregation we have to aggregate to get the final output of Onida sales in each state.

Driver Class Program

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

public class OnidaStateWiseDriver {

 public static void main(String[] args) throws Exception {
 if (args.length != 2) {
System.err.println("Usage: OnidaStateWise <input path> <output path>");
System.exit(-1);
}

//Job Related Configurations
Configuration conf = new Configuration();
Job job = new ~~Job~~(conf, "My OnidaStateWise with combiner");
job.setJarByClass(OnidaStateWiseDriver.class);


// Specify the number of reducer to 2
//job.setNumReduceTasks(2);

//Provide paths to pick the input file for the job
FileInputFormat.setInputPaths(job, new Path(args[0]));


//Provide paths to pick the output file for the job, and delete it if already
present
Path outputPath = new Path(args[1]);
FileOutputFormat.setOutputPath(job, outputPath);
outputPath.getFileSystem(conf).delete(outputPath, true);
//To set the mapper and reducer of this job
job.setMapperClass(OnidaStateWiseMapper.class);
job.setReducerClass(OnidaStateWiseReducer.class);

//Set the combiner
job.setCombinerClass(OnidaStateWiseReducer.class);

//set the input and output format class
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

//set up the output key and value classes
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

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

<u>Mapper Class Program</u>

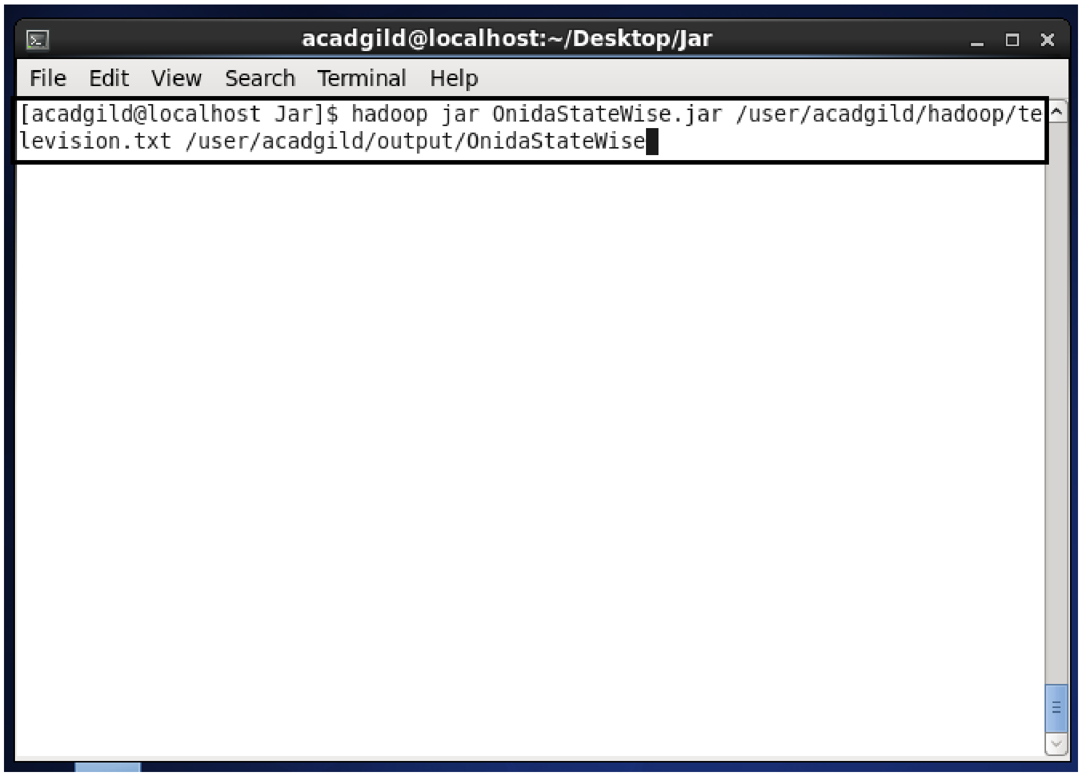
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.util.*;

public class OnidaStateWiseMapper
extends Mapper<LongWritable, Text, Text, IntWritable> {

private final static IntWritable one = new IntWritable(1);
private final static IntWritable zero = new IntWritable(0);
@Override
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
  String line [] = value.toString().split("\\|");

  if(line[0].equalsIgnoreCase("Onida"))
  {
        Text t1 = new Text(line[3]);
        context.write(t1, one);
  }
  else
  {
        Text t1 = new Text(line[3]);
        context.write(t1, zero);
  }

  }
}
```

Reducer Class Program

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


public class OnidaStateWiseReducer
extends Reducer<Text, IntWritable, Text, IntWritable> {

@Override
public void reduce(Text key, Iterable<IntWritable> values,
Context context)
throws IOException, InterruptedException {
System.out.println("From The Reducer=>"+key) ;

int sum = 0;
for (IntWritable value : values) {
       sum+=value.get();
 }
 context.write(key, new IntWritable(sum));
}
}
```

Executing the Jar on the Dataset we use the command

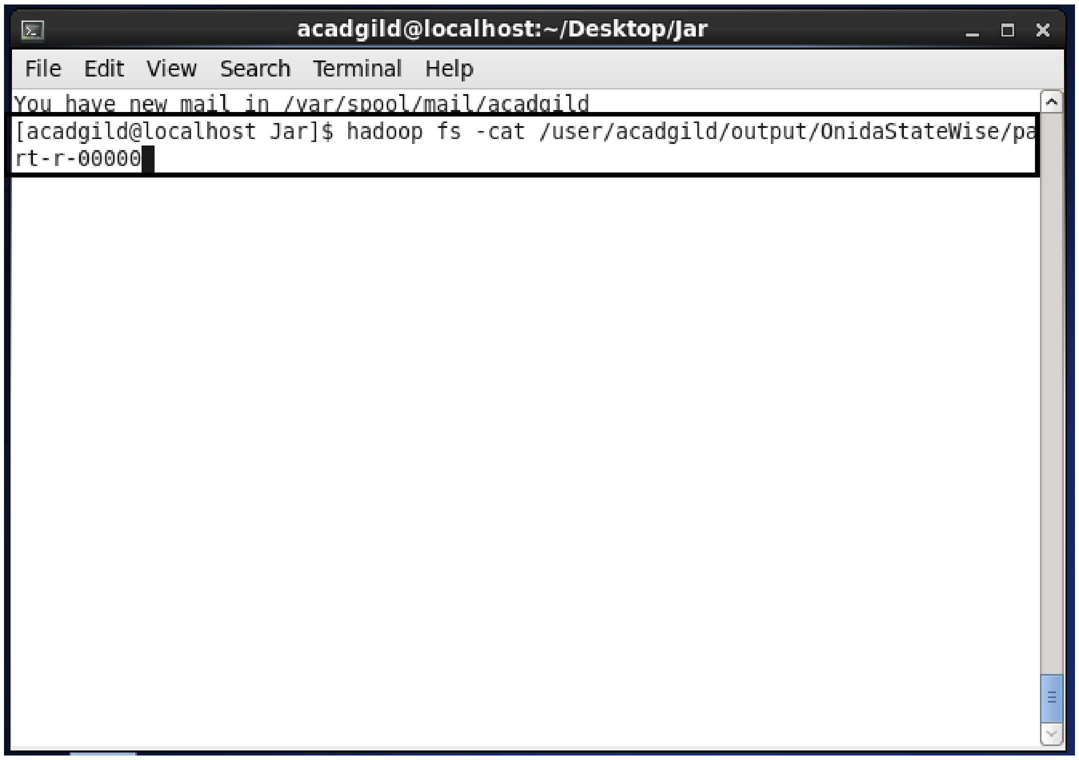
“hadoop jar OnidaStateWise.jar /user/acadgild/hadoop/television.txt /user/acadgild/output/ OnidaStateWise”


acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

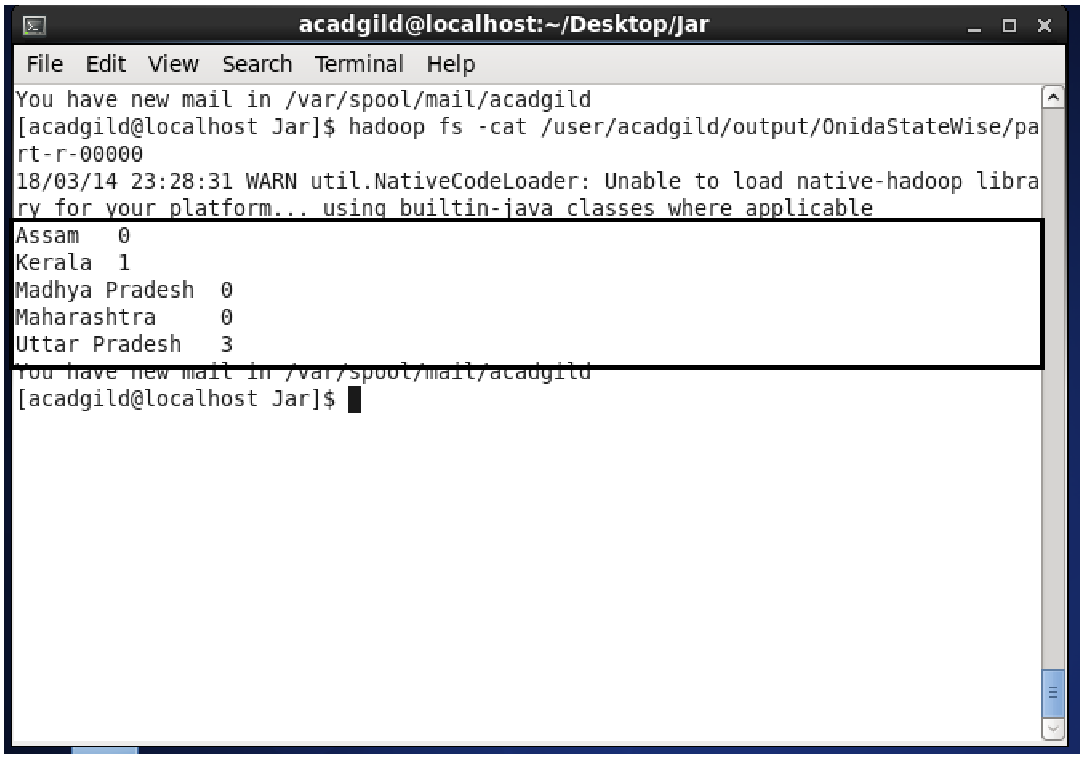
```
[acadgild@localhost Jar]$ hadoop jar OnidaStateWise.jar /user/acadgild/hadoop/te
levision.txt /user/acadgild/output/OnidaStateWise
```

After Successful execution of the Map Reduce Job on the Dataset we should see the output with total units sold for Onida company in each State.

“hadoop fs –cat /user/acadgild/output/OnidaStateWise/part-r-00000”


acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Jar]$ hadoop fs -cat /user/acadgild/output/OnidaStateWise/pa
rt-r-00000
```


acadgild@localhost:~/Desktop/Jar
File Edit View Search Terminal Help

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Jar]$ hadoop fs -cat /user/acadgild/output/OnidaStateWise/pa
rt-r-00000
18/03/14 23:28:31 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Assam   0
Kerala  1
Madhya Pradesh  0
Maharashtra     0
Uttar Pradesh   3
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost Jar]$
```