Assignment-4

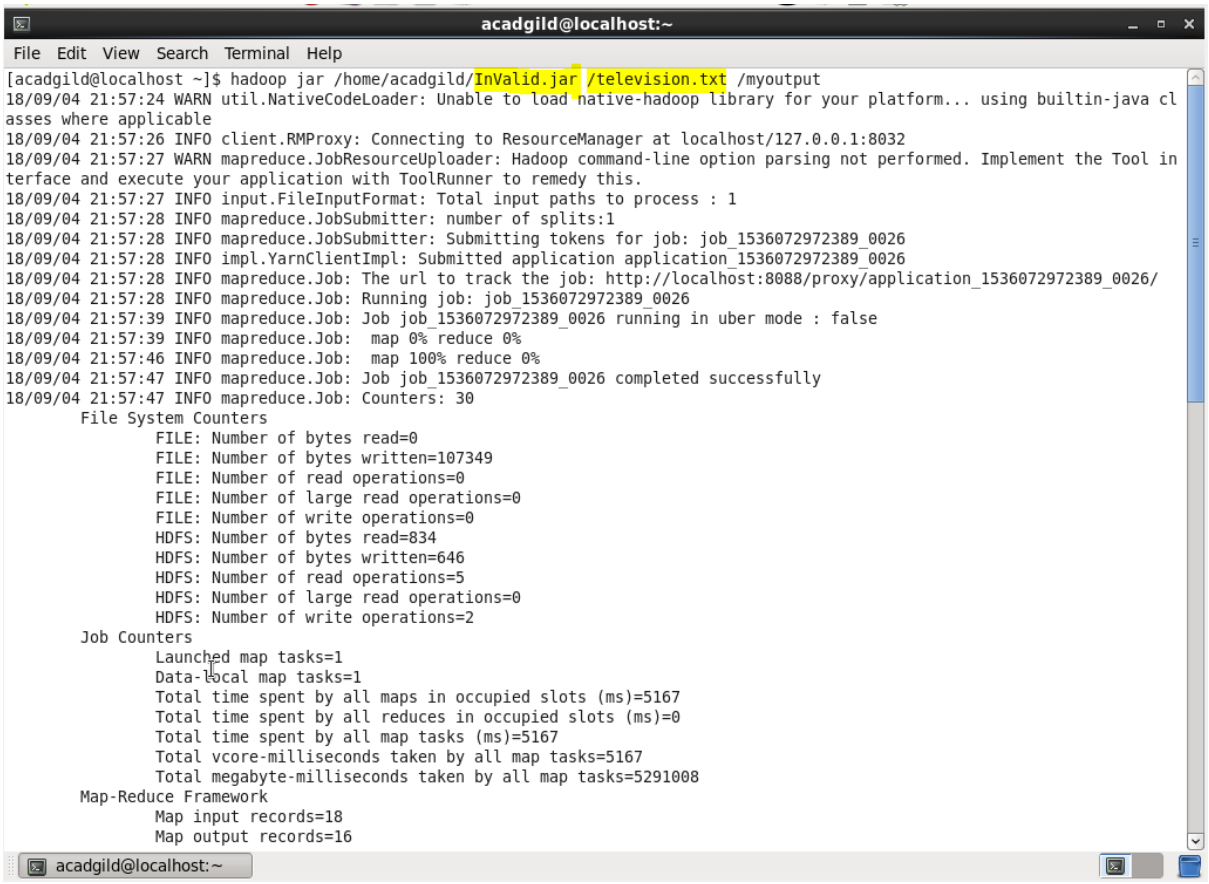
1). **Write a Map Reduce program to filter out the invalid records. Map only job will fit for this context.**

Mapreduce code written for the above task is as below:

```
package task1;

// imported all the built-in packages required for the task

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

      public class InValid
      {
         //Mapper class
         public static class Map extends Mapper <LongWritable ,/*Input
key Type */
         Text,                /*Input value Type*/
         Text,                /*Output key Type*/
         Text>        /*Output value Type*/
         {
            //Map function
            public void map(LongWritable key, Text value,
            Context context) throws IOException, InterruptedException
            {

            // Given data is splitted into an array based on the
delimiter present in the data.

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


           // If the company and product name not equals to "NA"
satisfies it will be sent to the output otherwise it won't.

              if(!parts[0].equals("NA") && !parts[1].equals("NA")) {

                 context.write(value, null);
               }
            }
         }
```

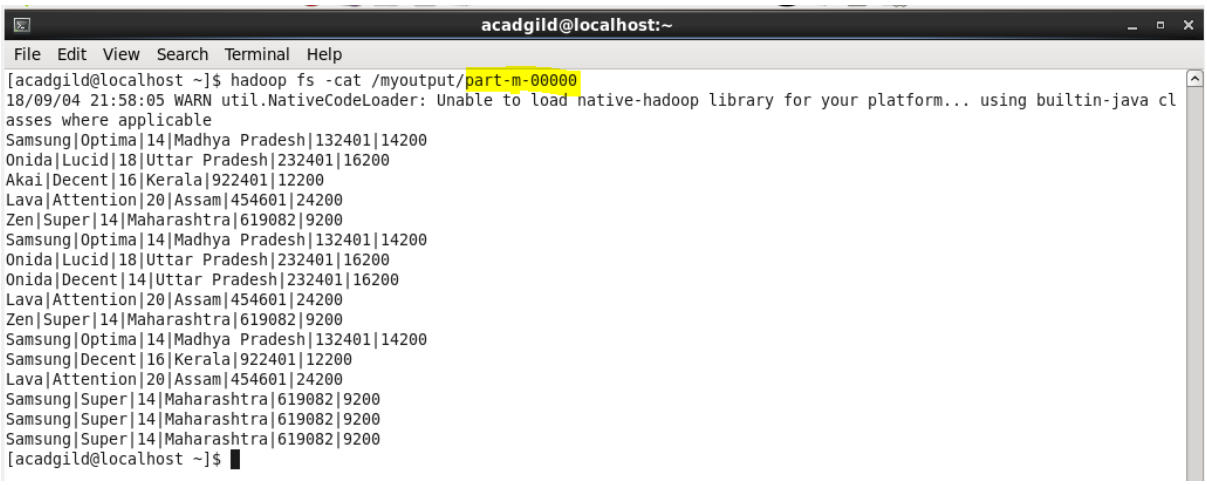
```
        //Main function
        public static void main(String args[])throws Exception
        {
             Configuration conf = new Configuration();
           Job job = Job.getInstance(conf, "InValid");

           job.setJarByClass(InValid.class);
           job.setOutputKeyClass(Text.class);
           job.setOutputValueClass(IntWritable.class);
           job.setMapperClass(Map.class);
           job.setInputFormatClass(TextInputFormat.class);
           job.setOutputFormatClass(TextOutputFormat.class);

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

          //Reducer tasks are set to "0" as reducer is not required
for this task.
           job.setNumReduceTasks(0);

           Path outputPath = new Path(args[1]);
           outputPath.getFileSystem(conf).delete(outputPath, true);

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

Executed the code **InValid.jar** with input file **television.txt**



Got the required output as shown in the below screenshot without "**NA**" records.

As there is no reduce code, the output file name will be with **'m'** instead of **'r'** as highlighted in the screenshot.



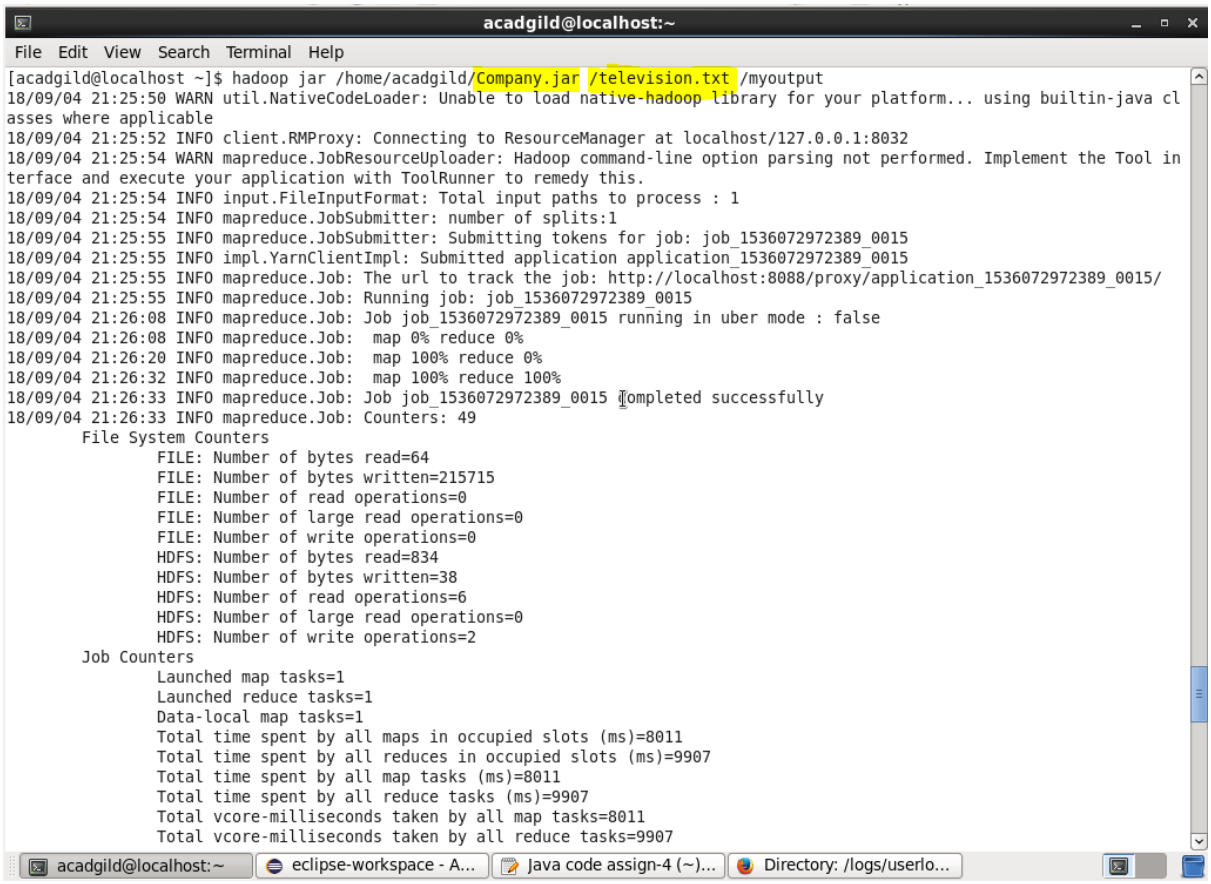
2). Write a Map Reduce program to calculate the total units sold for each Company.

Mapreduce code written for the above task is as below:

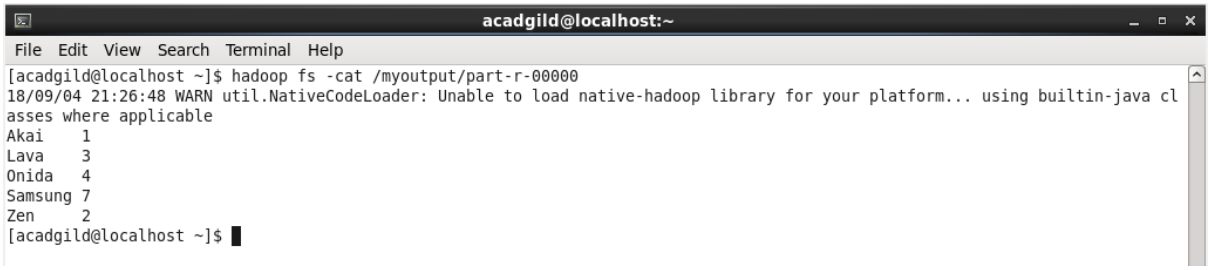
```
package task1;

// imported all the built-in packages required for the task

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

public class Company
{
      //Mapper class
      public static class Map extends Mapper <LongWritable ,/*Input key
Type */
      Text,                /*Input value Type*/
      Text,                /*Output key Type*/
      IntWritable>        /*Output value Type*/
      {
            //Map function
            public void map(LongWritable key, Text value,
                        Context context) throws IOException,
InterruptedException
            {
            // Given data is splitted into an array based on the
delimiter present in the data.
                  String[] parts = value.toString().split("[|]");

                  if(!parts[0].equals("NA")) {

            //If the company name is not equals "NA" satisfies then it
will perform the following steps.

                  value.set(parts[0]);
            //First position in the array is set "value".

                  context.write(new Text(value), new IntWritable(1));

            //The value is passed as key to the output of Mapper and as
the value is set with string, it is converted into Text. The value to
the key in the output is passed as "1".

                  }
            }
      }
```

```
//Reducer class (Received output from the mapper as Text and
intWritable)
      public static class Reduce extends Reducer< Text, IntWritable,
Text, IntWritable >
      {

            //Reduce function
            public void reduce( Text key, Iterable<IntWritable>
values,Context context) throws IOException, InterruptedException
            {
                  int sum=0;

            //the below step will check for the each value.

                  for(IntWritable x:values) {

            //each value will be added to the variable "sum"

                        sum +=x.get();

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

            //Now the output will be with key (company name) and value
(count of units)

            }
      }

      //Main function
      public static void main(String args[])throws Exception
      {
            Configuration conf = new Configuration();
            Job job = Job.getInstance(conf, "Company");

            job.setJarByClass(Company.class);
            job.setOutputKeyClass(Text.class);
            job.setOutputValueClass(IntWritable.class);
            job.setMapperClass(Map.class);
            job.setCombinerClass(Reduce.class);
            job.setReducerClass(Reduce.class);
            job.setInputFormatClass(TextInputFormat.class);
            job.setOutputFormatClass(TextOutputFormat.class);

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

            Path outputPath = new Path(args[1]);
            outputPath.getFileSystem(conf).delete(outputPath, true);

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

Executed the code **Company.jar** with input file **television.txt**



Got the required output as shown in the below screenshot.



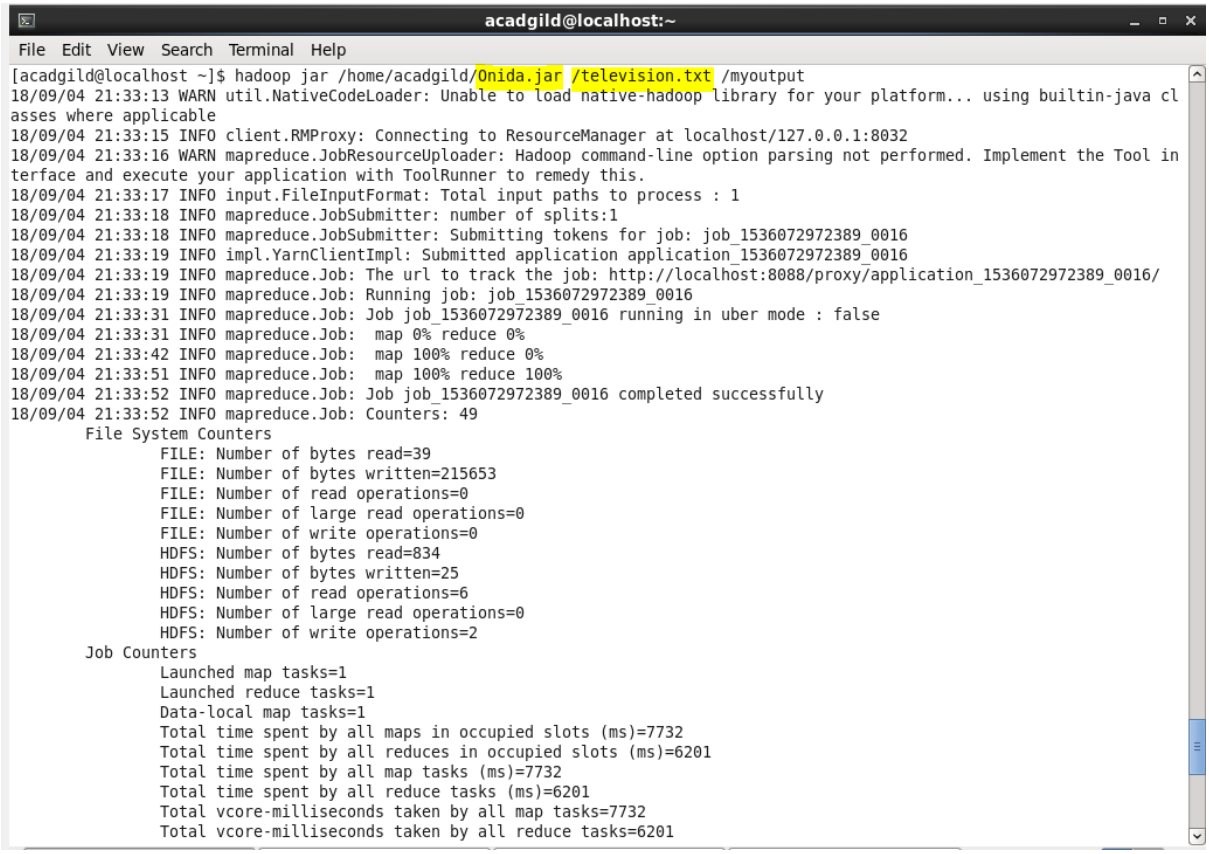
3). Write a Map Reduce program to calculate the total units sold in each state for Onida company.

Mapreduce code written for the above task is as below:

```
package task1;

// imported all the built-in packages required for the task

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.Path;

      public class Onida
      {
         //Mapper class
         public static class Map extends Mapper <LongWritable ,/*Input
key Type */
         Text,                /*Input value Type*/
         Text,                /*Output key Type*/
         IntWritable>        /*Output value Type*/
         {
            //Map function
            public void map(LongWritable key, Text value,
            Context context) throws IOException, InterruptedException
            {

            //Given data is splitted into an array based on the
delimiter present in the data.

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

            //If the company name equals to "Onida" satisfies it will
execute the statements in if condition.

               if(parts[0].equals("Onida")) {

            //The state name is set to the variable "value".

               value.set(parts[3]);

               context.write(new Text(value), new IntWritable(1));

           //The value is passed as key to the output of Mapper and as
the value is set with string it is converted into Text. The value to
the key in the output is passed as "1".
            }
```

```
        }
    }


        //Reducer class (Received output from the mapper as Text and
intWritable
        public static class Reduce extends Reducer< Text, IntWritable,
Text, IntWritable >
        {

           //Reduce function
           public void reduce( Text key, Iterable<IntWritable>
values,Context context) throws IOException, InterruptedException
              {
                 int sum=0;

           //the below step will check for the each value.

                 for(IntWritable x:values) {

           //each value will be added to the variable "sum"

                      sum +=x.get();

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

           //Now the output will be with key (company name) and value
(count of units)
              }
        }


        //Main function
        public static void main(String args[])throws Exception
        {
             Configuration conf = new Configuration();
           Job job = Job.getInstance(conf, "Onida");

           job.setJarByClass(Onida.class);
           job.setOutputKeyClass(Text.class);
           job.setOutputValueClass(IntWritable.class);
           job.setMapperClass(Map.class);
           job.setCombinerClass(Reduce.class);
           job.setReducerClass(Reduce.class);
           job.setInputFormatClass(TextInputFormat.class);
           job.setOutputFormatClass(TextOutputFormat.class);

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

           Path outputPath = new Path(args[1]);
           outputPath.getFileSystem(conf).delete(outputPath, true);

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

Executed the code **Onida.jar** with input file **television.txt**



Got the required output as shown in the below screenshot.