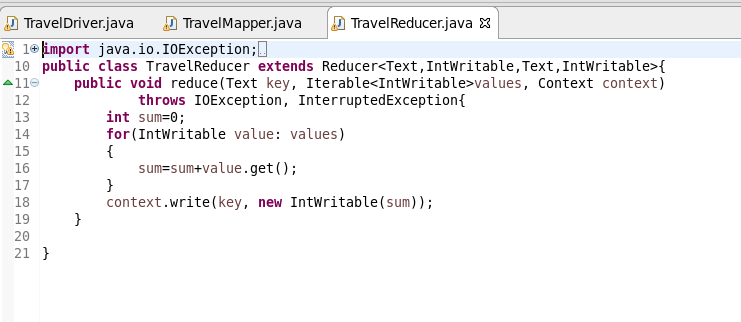
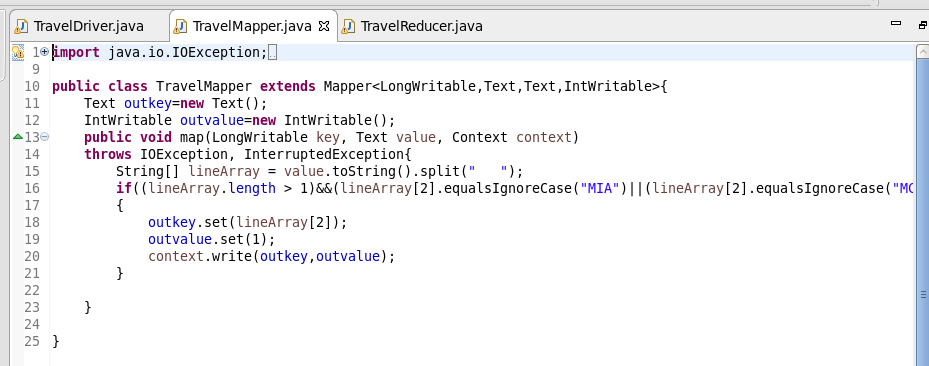
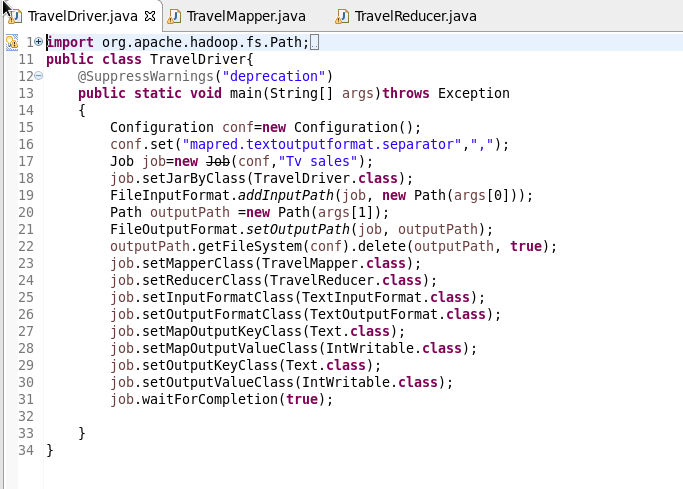
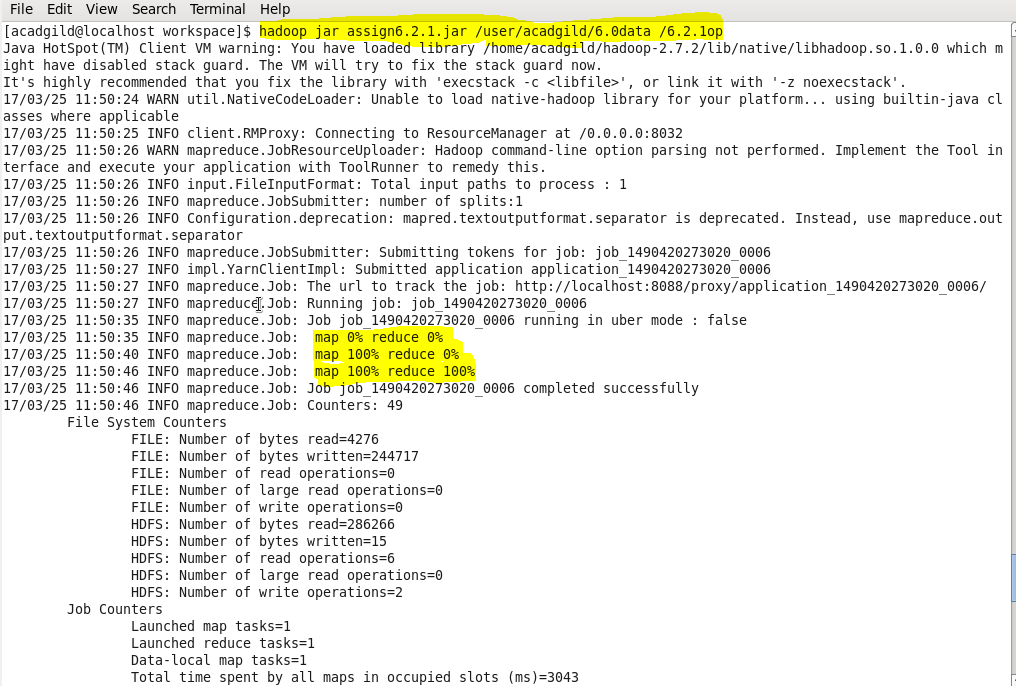
*Ques:- Use key value output format to save the output of Session 6 – Assignment 1(Travel Data analysis) and save the output as comma (,) separated instead of tab(\t) separated.*

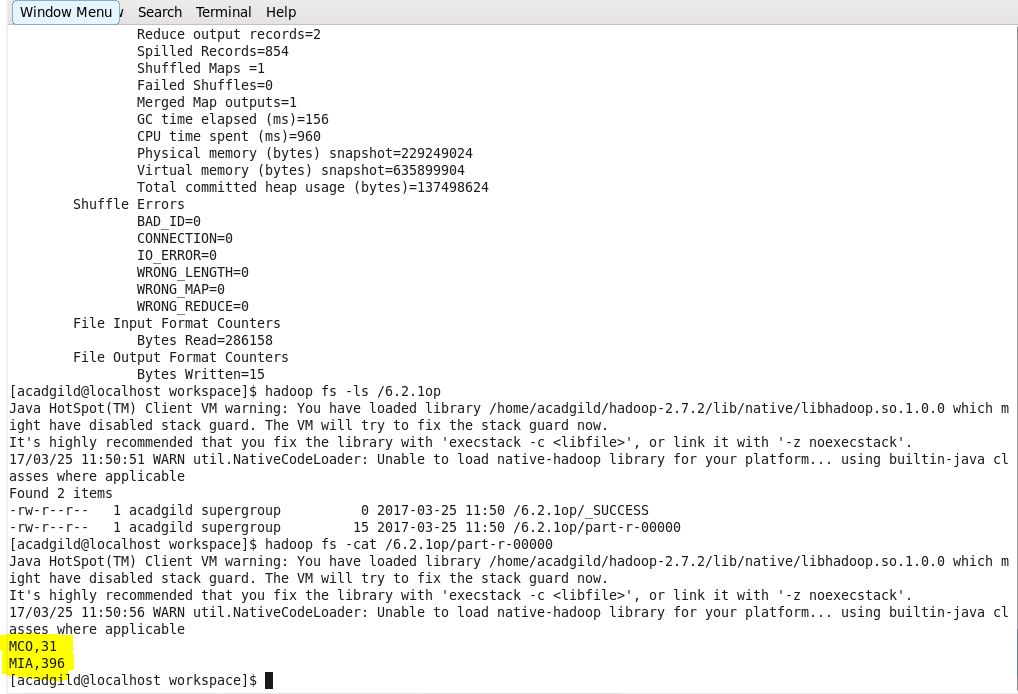
1. Find out how many people has chosen their destination as MIA and MCO

import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.conf.\*;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
public class TravelDriver{  
    @SuppressWarnings("deprecation")  
    public static void main(String[] args)throws Exception  
    {  
        Configuration conf=new Configuration();  
        conf.set("mapred.textoutputformat.separator",",");  
        Job job=new Job(conf,"Tv sales");  
        job.setJarByClass(TravelDriver.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        Path outputPath =new Path(args[1]);  
        FileOutputFormat.setOutputPath(job, outputPath);  
        outputPath.getFileSystem(conf).delete(outputPath, true);  
        job.setMapperClass(TravelMapper.class);  
        job.setReducerClass(TravelReducer.class);  
        job.setInputFormatClass(TextInputFormat.class);  
        job.setOutputFormatClass(TextOutputFormat.class);  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(IntWritable.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
        job.waitForCompletion(true);  
  
    }  
}import java.io.IOException;  
  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
  
public class TravelMapper extends Mapper<LongWritable,Text,Text,IntWritable>{  
    Text outkey=new Text();  
    IntWritable outvalue=new IntWritable();  
    public void map(LongWritable key, Text value, Context context)  
    throws IOException, InterruptedException{  
        String[] lineArray = value.toString().split("    ");  
        if((lineArray.length >  
1)&&(lineArray[2].equalsIgnoreCase("MIA")||(lineArray[2].equalsIgnoreCase("MCO"))))  
        {  
            outkey.set(lineArray[2]);  
            outvalue.set(1);  
            context.write(outkey,outvalue);  
        }  
  
    }  
  
}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.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
import org.apache.hadoop.mapreduce.Mapper.Context;  
public class TravelReducer extends Reducer<Text,IntWritable,Text,IntWritable>{  
    public void reduce(Text key, Iterable<IntWritable>values, Context context)  
            throws IOException, InterruptedException{  
        int sum=0;  
        for(IntWritable value: values)  
        {  
            sum=sum+value.get();  
        }  
        context.write(key, new IntWritable(sum));  
    }  
  
}



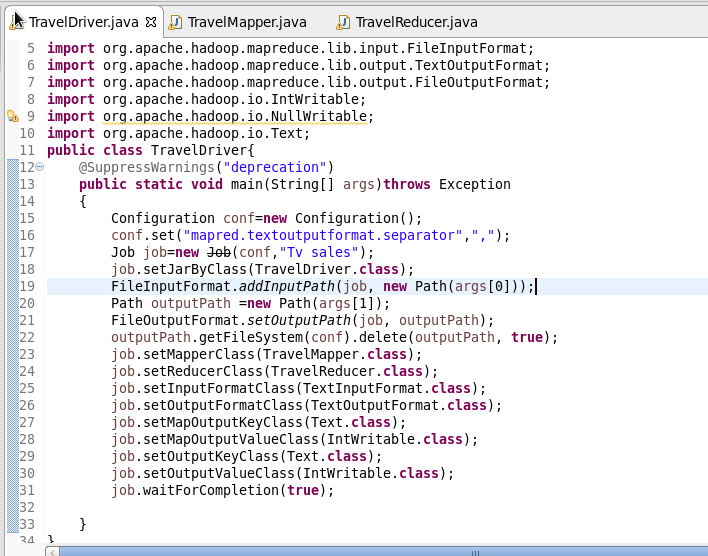
**Output:**

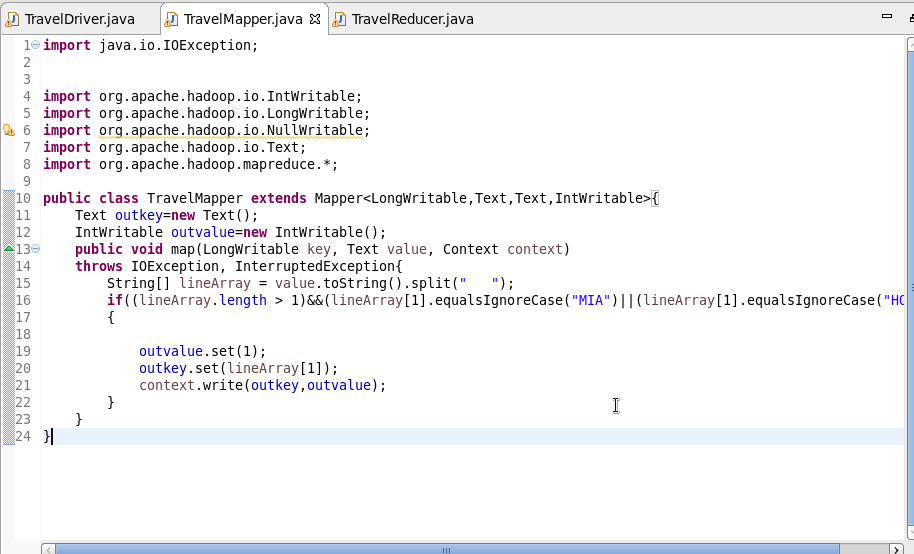
****

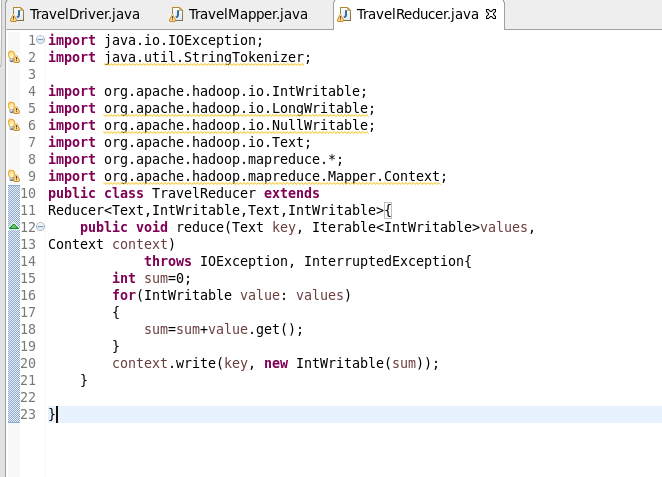
****

1. **Find out the number of people undertaken the trips from the places MIA and HOU**

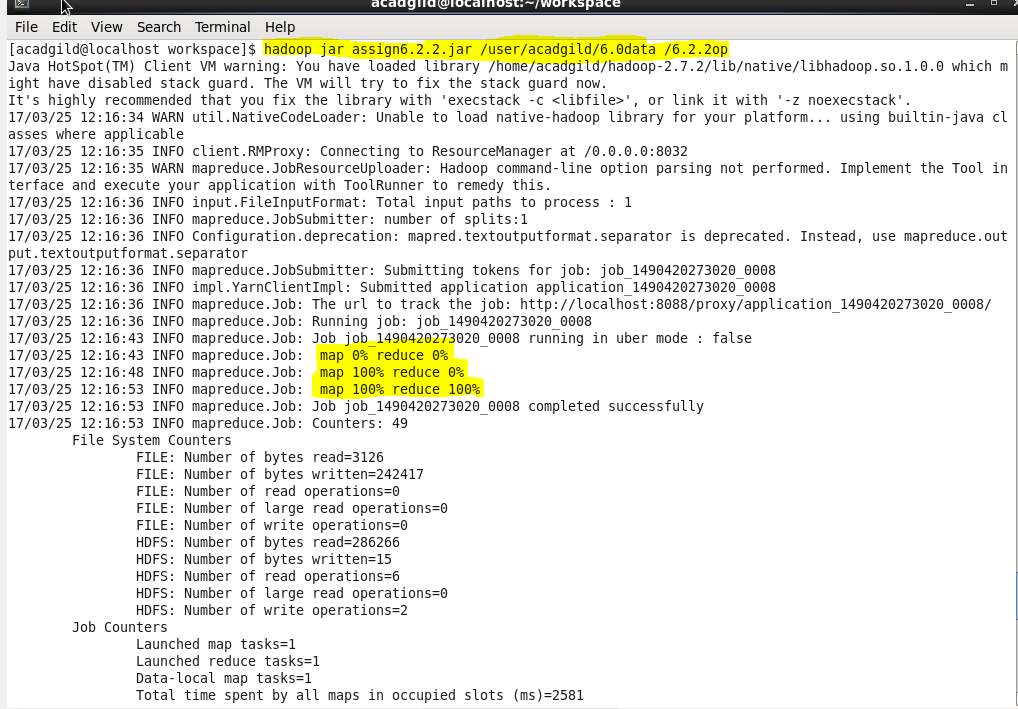
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.conf.\*;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
public class TravelDriver{  
    @SuppressWarnings("deprecation")  
    public static void main(String[] args)throws Exception  
    {  
        Configuration conf=new Configuration();  
        conf.set("mapred.textoutputformat.separator",",");  
        Job job=new Job(conf,"Tv sales");  
        job.setJarByClass(TravelDriver.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        Path outputPath =new Path(args[1]);  
        FileOutputFormat.setOutputPath(job, outputPath);  
        outputPath.getFileSystem(conf).delete(outputPath, true);  
        job.setMapperClass(TravelMapper.class);  
        job.setReducerClass(TravelReducer.class);  
        job.setInputFormatClass(TextInputFormat.class);  
        job.setOutputFormatClass(TextOutputFormat.class);  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(IntWritable.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
        job.waitForCompletion(true);  
  
    }  
}import java.io.IOException;  
  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
  
public class TravelMapper extends Mapper<LongWritable,Text,Text,IntWritable>{  
    Text outkey=new Text();  
    IntWritable outvalue=new IntWritable();  
    public void map(LongWritable key, Text value, Context context)  
    throws IOException, InterruptedException{  
        String[] lineArray = value.toString().split("    ");  
        if((lineArray.length >  
1)&&(lineArray[1].equalsIgnoreCase("MIA")||(lineArray[1].equalsIgnoreCase("HOU"))))  
        {  
  
            outvalue.set(1);  
            outkey.set(lineArray[1]);  
            context.write(outkey,outvalue);  
        }  
    }  
}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.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
import org.apache.hadoop.mapreduce.Mapper.Context;  
public class TravelReducer extends  
Reducer<Text,IntWritable,Text,IntWritable>{  
    public void reduce(Text key, Iterable<IntWritable>values,  
Context context)  
            throws IOException, InterruptedException{  
        int sum=0;  
        for(IntWritable value: values)  
        {  
            sum=sum+value.get();  
        }  
        context.write(key, new IntWritable(sum));  
    }  
  
}

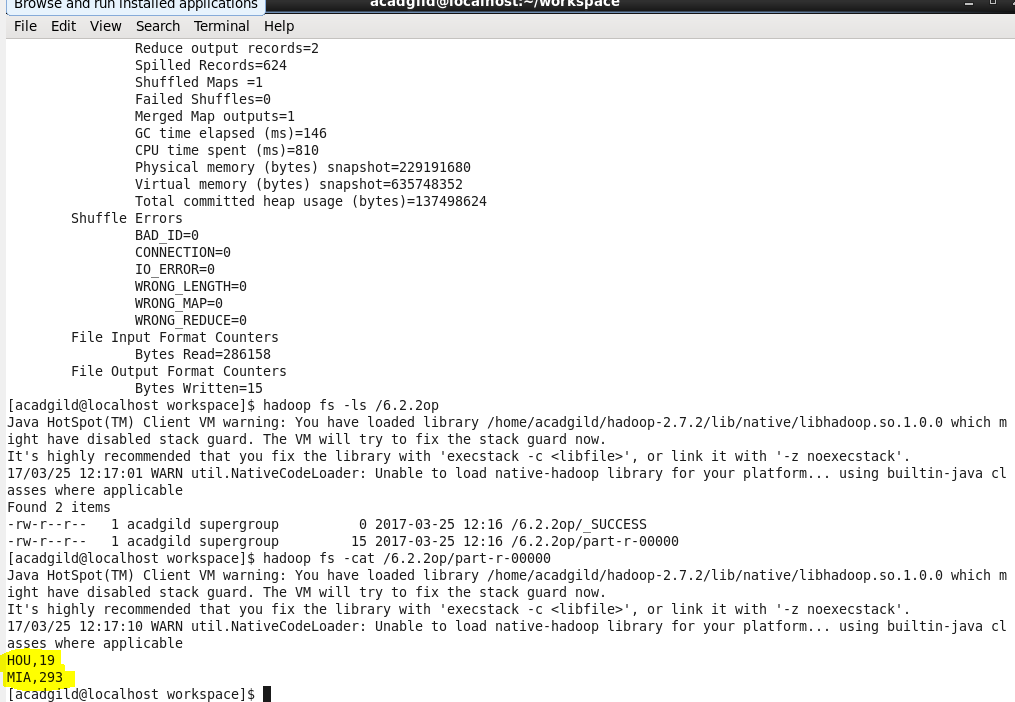
****

****

****

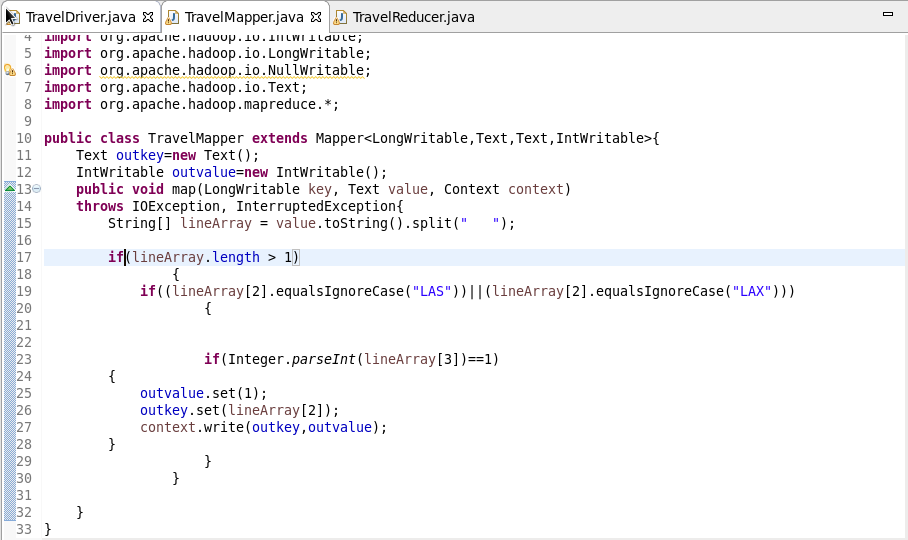
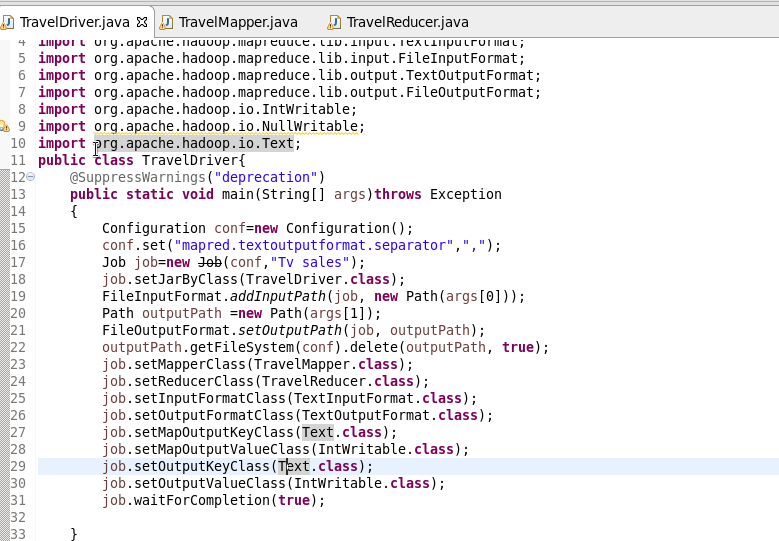
**Output:**

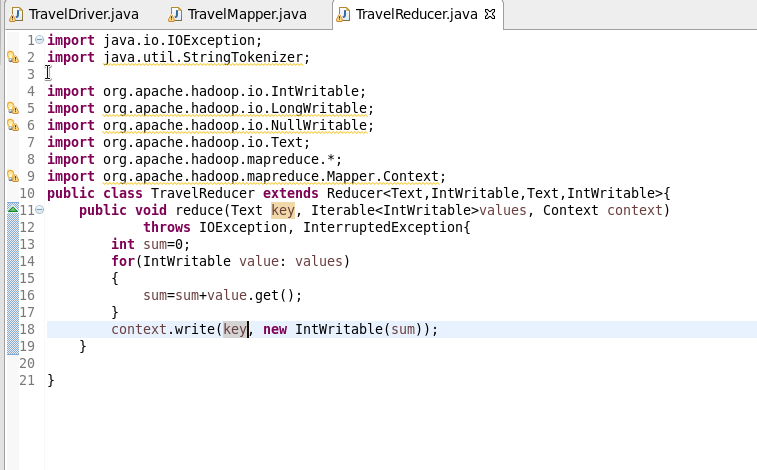
****

****

1. Find out how many people has chosen airline mode of travel for the places LAS and LAX

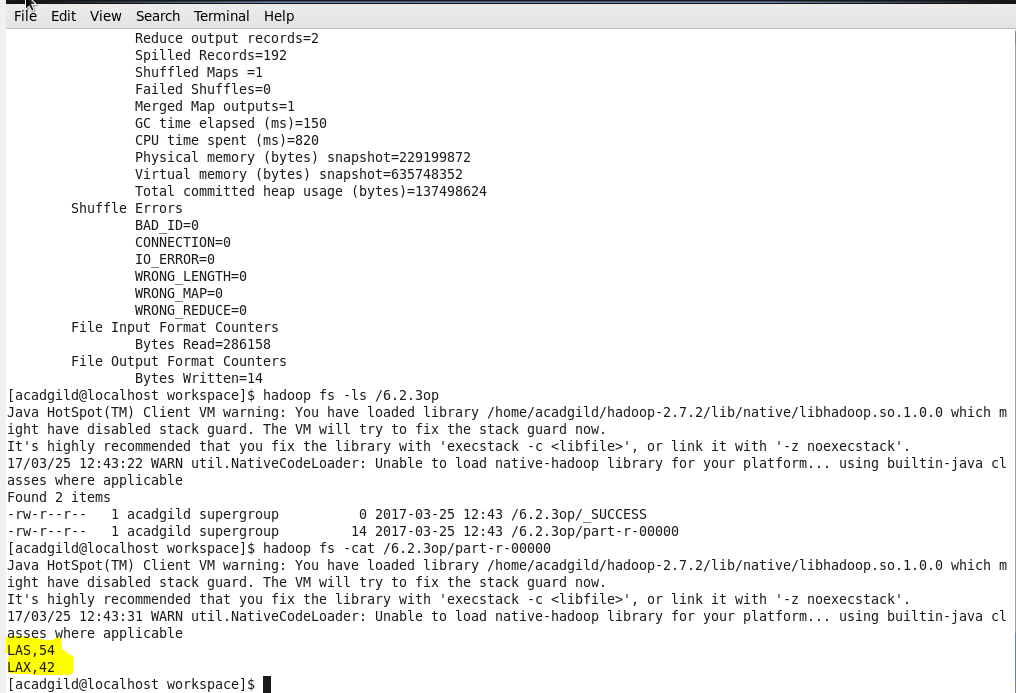
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.conf.\*;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
public class TravelDriver{  
    @SuppressWarnings("deprecation")  
    public static void main(String[] args)throws Exception  
    {  
        Configuration conf=new Configuration();  
        conf.set("mapred.textoutputformat.separator",",");  
        Job job=new Job(conf,"Tv sales");  
        job.setJarByClass(TravelDriver.class);  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        Path outputPath =new Path(args[1]);  
        FileOutputFormat.setOutputPath(job, outputPath);  
        outputPath.getFileSystem(conf).delete(outputPath, true);  
        job.setMapperClass(TravelMapper.class);  
        job.setReducerClass(TravelReducer.class);  
        job.setInputFormatClass(TextInputFormat.class);  
        job.setOutputFormatClass(TextOutputFormat.class);  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(IntWritable.class);  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
        job.waitForCompletion(true);  
  
    }  
}import java.io.IOException;  
  
  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
  
public class TravelMapper extends Mapper<LongWritable,Text,Text,IntWritable>{  
    Text outkey=new Text();  
    IntWritable outvalue=new IntWritable();  
    public void map(LongWritable key, Text value, Context context)  
    throws IOException, InterruptedException{  
        String[] lineArray = value.toString().split("    ");  
  
        if(lineArray.length > 1)  
                {  
            if((lineArray[2].equalsIgnoreCase("LAS"))||(lineArray[2].equalsIgnoreCase("LAX")))  
                    {  
  
  
                    if(Integer.parseInt(lineArray[3])==1)  
        {  
            outvalue.set(1);  
            outkey.set(lineArray[2]);  
            context.write(outkey,outvalue);  
        }  
                    }  
                }  
  
    }  
}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.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.\*;  
import org.apache.hadoop.mapreduce.Mapper.Context;  
public class TravelReducer extends Reducer<Text,IntWritable,Text,IntWritable>{  
    public void reduce(Text key, Iterable<IntWritable>values, Context context)  
            throws IOException, InterruptedException{  
        int sum=0;  
        for(IntWritable value: values)  
        {  
            sum=sum+value.get();  
        }      context.write(key, new IntWritable(sum));  
    }}}

****

****

**Output:**

****

****