Driver class

```
package sparkAssignmentMRpayment;


import java.io.IOException;

import org.apache.hadoop.conf.*;
import org.apache.hadoop.util.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.*;

public class driverPayment extends Configured implements Tool {
	public static void main(String[] args) throws Exception {
		int returnStatus = ToolRunner.run(new Configuration(), new driverPayment(), args);
		System.exit(returnStatus);
	}


	public int run(String[] args) throws IOException{

		Job job = new Job(getConf());

		job.setJobName("Payment Cooccurence");

		job.setJarByClass(driverPayment.class);

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

		//job.setMapOutputKeyClass(TextPair.class);
		//job.setMapOutputValueClass(IntWritable.class);

		job.setMapperClass(driverPaymentMapper.class);

		job.setReducerClass(driverPaymentReducer.class);
		job.setNumReduceTasks(1);

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


		try {
			return job.waitForCompletion(true) ? 0 : 1;
		} catch (ClassNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
            return 0;



        }

}
```

Mapper Class

```
package sparkAssignmentMRpayment;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
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.Mapper;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

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

        @Override

        public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {


            {

                String records [] = value.toString().trim().split(",");

                if (records.length == 17)

                {
                    String VendorID = records[0];
                    String tpep_pickup_datetime = records[1];
                    String tpep_dropoff_datetime = records[2];
                    String passenger_count= records[3];
                    String trip_distance= records[4];
                    String RatecodeID= records[5];
```

```
                                String store_and_fwd_flag= records[6];
                                String PULocationID= records[7];
                                String DOLocationID= records[8];
                                String payment_type= records[9];
                                String fare_amount= records[10];
                                String extra= records[11];
                                String mta_tax=records[12];
                                String tip_amount= records[13];
                                String tolls_amount= records[14];
                                String improvement_surcharge= records[15];
                                String total_amount= records[16];

                                //int VendorIDInt = Integer.parseInt(VendorID);

                                for(String st1 :  records)
                                {
                                        if (payment_type.equals("payment_type")){
                                                continue;}
                                        else
                                        {context.write(new Text(payment_type), new
IntWritable(1)); }

                                }
                        }
                }
        }
}
```

Reducer Class

```
package sparkAssignmentMRpayment;


import java.io.*;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
import java.util.Map.Entry;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.Reducer.Context;
import org.apache.hadoop.mapred.lib.MultipleOutputs;

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

        static int totalCount;
```

```
	@Override
	public void setup(Context context) throws IOException,
	InterruptedException
	{
		totalCount = 0;
	}

	TreeMap<Text,IntWritable>result2 = new TreeMap<Text, IntWritable>();

	public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException {

		int count = 0;
		for (@SuppressWarnings("unused")
		IntWritable value : values) {
			count++;
		}
		result2.put(new Text(key),new IntWritable(count));

	}

	protected void cleanup(Context context)
			throws IOException, InterruptedException {

		Set<Entry<Text, IntWritable>> set = result2.entrySet();
		List<Entry<Text, IntWritable>> list = new
ArrayList<Entry<Text,IntWritable>>(set);
		Collections.sort( list, new Comparator<Map.Entry<Text,
IntWritable>>()
		{
			public int compare( Map.Entry<Text, IntWritable> o1,
Map.Entry<Text,IntWritable> o2 )
			{
				return (o1.getValue()).compareTo( o2.getValue() );
			}
		});

		for(Map.Entry<Text,IntWritable> entry:list){
			context.write(entry.getKey(),entry.getValue());
			totalCount++;

		}

	}
}
```