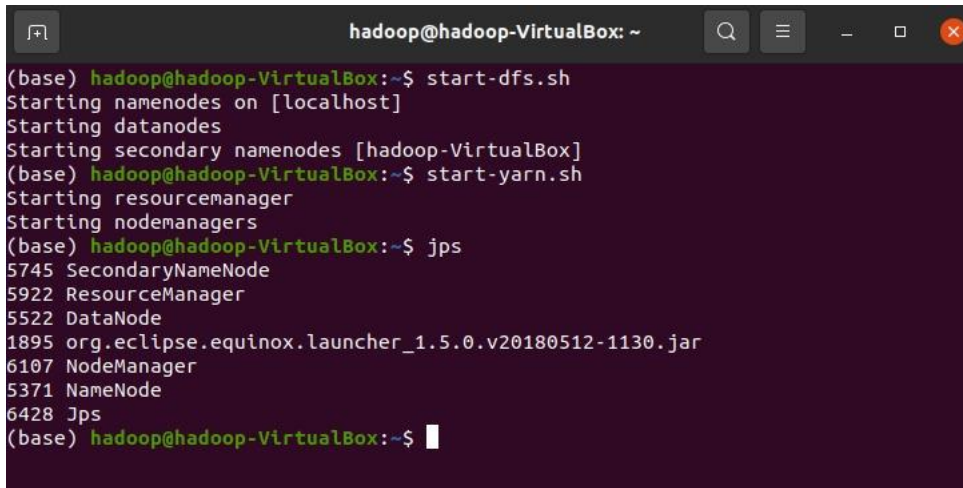


### Implementation Steps:

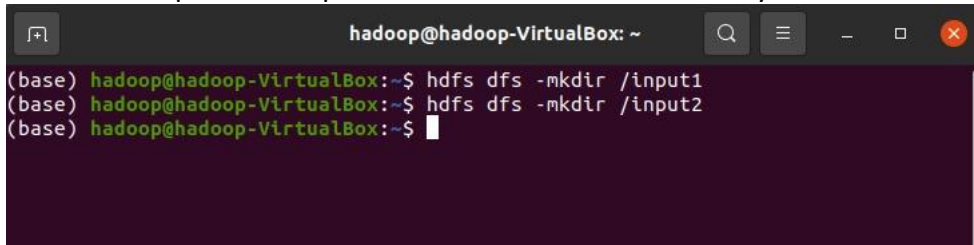
1. Create the CustomerMapper, TransactionMapper, JoinReducer, CustomerCountMapper, CustomerCountReducer and CustomerTransactionDriver classes and generate a .jar file. The complete implementation and code are explained in the section 2.3. A separate file 'MapReduce\_Code.docx' is submitted along with the coursework.
2. Start the HDFS file system as shown below and confirm if all the services are running or not.



```
hadoop@hadoop-VirtualBox: ~  
(base) hadoop@hadoop-VirtualBox:~$ start-dfs.sh  
Starting namenodes on [localhost]  
Starting datanodes  
Starting secondary namenodes [hadoop-VirtualBox]  
(base) hadoop@hadoop-VirtualBox:~$ start-yarn.sh  
Starting resourcemanager  
Starting nodemanagers  
(base) hadoop@hadoop-VirtualBox:~$ jps  
5745 SecondaryNameNode  
5922 ResourceManager  
5522 DataNode  
1895 org.eclipse.equinox.launcher_1.5.0.v20180512-1130.jar  
6107 NodeManager  
5371 NameNode  
6428 Jps  
(base) hadoop@hadoop-VirtualBox:~$
```

Fig-2

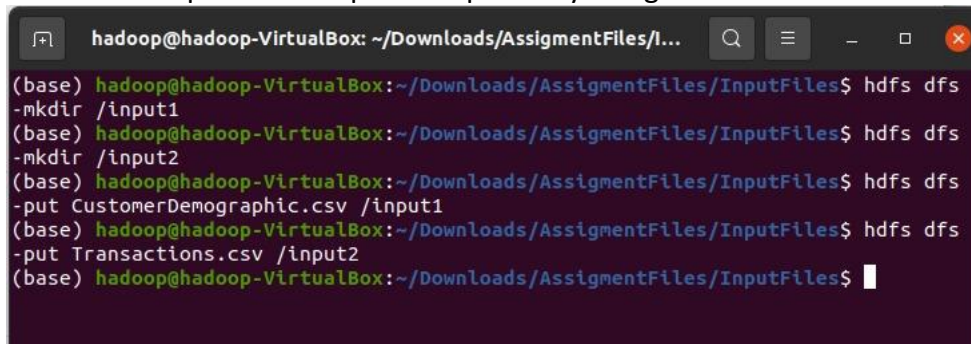
3. Create the input1 and input2 directories in the HDFS file system as shown below.



```
hadoop@hadoop-VirtualBox: ~  
(base) hadoop@hadoop-VirtualBox:~$ hdfs dfs -mkdir /input1  
(base) hadoop@hadoop-VirtualBox:~$ hdfs dfs -mkdir /input2  
(base) hadoop@hadoop-VirtualBox:~$
```

Fig-3

4. Copy and place the input files 'CustomerDemographic.csv' and 'Transactions.csv' in the folders 'input1' and 'input2' respectively using the commands as shown below.



```
hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles/InputFiles  
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles/InputFiles$ hdfs dfs  
-mkdir /input1  
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles/InputFiles$ hdfs dfs  
-mkdir /input2  
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles/InputFiles$ hdfs dfs  
-put CustomerDemographic.csv /input1  
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles/InputFiles$ hdfs dfs  
-put Transactions.csv /input2  
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles/InputFiles$
```

Fig-4

5. Run the jar file using the below command.  
hadoop jar <jar file name> <input1 path> <input2 path> <output path>

Example: `hadoop jar CustomerDemographicsAnalysis.jar /input1 /input2 /output`

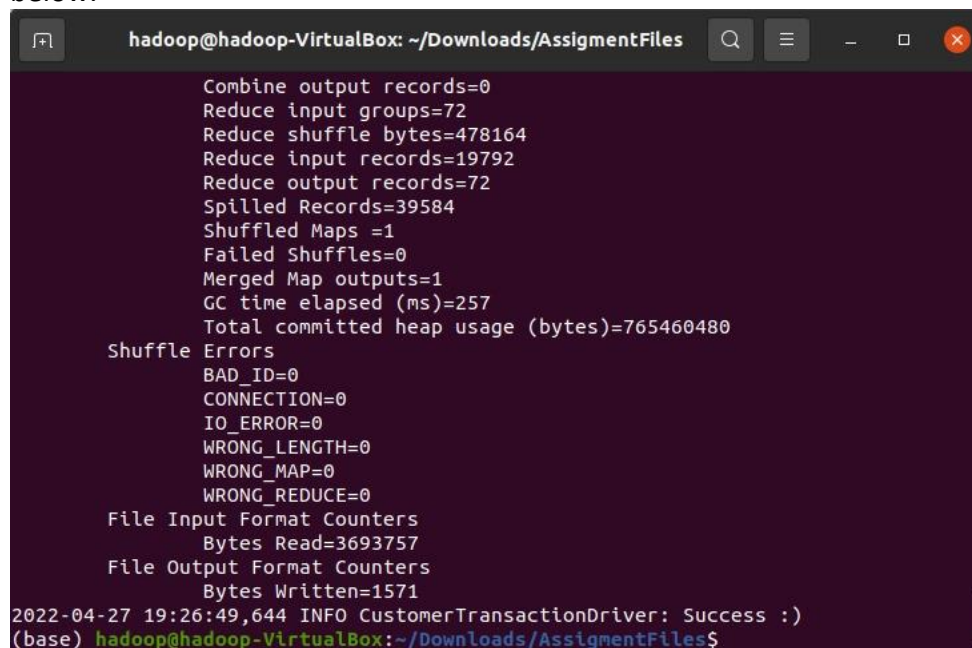
**Note:** There is no need to provide the driver class name, as the jar file already consists of those details. It is a good practice to avoid using driver class name, when there is only one driver class. So that, it would be easier for the second person to run the job as they don't need to worry about the driver class name.



```
hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles/InputFiles$ cd ..
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles$ ls
CustomerCount.jar      InputFiles  Project-Backup
CustomerDemographicsAnalysis.jar  Jars      Project-Source
CustomerTransactionDriver.java    output    wordcount.jar
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles$ hadoop jar CustomerDemographicsAnalysis.jar /input1 /input2 /output
```

Fig-5

- When the jobs are run, the end of the run will display the success message as shown below.

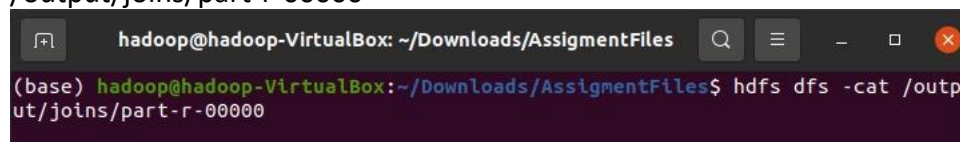


```
hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles
Combine output records=0
Reduce input groups=72
Reduce shuffle bytes=478164
Reduce input records=19792
Reduce output records=72
Spilled Records=39584
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=257
Total committed heap usage (bytes)=765460480
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=3693757
File Output Format Counters
Bytes Written=1571
2022-04-27 19:26:49,644 INFO CustomerTransactionDriver: Success :)
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles$
```

Fig-6

- The code is developed to put the output of the inner join MapReduce job (first MR job) in following location.

`/output/joins/part-r-00000`



```
hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles
(base) hadoop@hadoop-VirtualBox:~/Downloads/AssignmentFiles$ hdfs dfs -cat /output/joins/part-r-00000
```

```

hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles
, $507.58, 38750
3499, Shelton Tewkesberrie, Male, 29, 1979-06-17, 42.00 , NA, Manufacturing, Mass Custo
mer, N, Yes, 7, 18403, 67, 09/11/2017, TRUE, Approved, Norco Bicycles, Road, medium, medium
, 544.05, $376.84, 38859
3500, Josy Fleeman, Female, 71, 1967-07-21, 54.00 , NA, Entertainment, Affluent Custome
r, N, No, 17, 6309, 69, 09/01/2017, TRUE, Approved, Giant Bicycles, Road, medium, medium, 79
2.9, $594.68, 33879
3500, Josy Fleeman, Female, 71, 1967-07-21, 54.00 , NA, Entertainment, Affluent Custome
r, N, No, 17, 8562, 33, 08/08/2017, FALSE, Approved, Giant Bicycles, Standard, medium, smal
l, 1311.44, "$1,167.18", 33888
3500, Josy Fleeman, Female, 71, 1967-07-21, 54.00 , NA, Entertainment, Affluent Custome
r, N, No, 17, 2463, 2, 25/04/2017, FALSE, Approved, Solex, Standard, medium, medium, 71.49, $
53.62, 41245
3500, Josy Fleeman, Female, 71, 1967-07-21, 54.00 , NA, Entertainment, Affluent Custome
r, N, No, 17, 8416, 74, 16/02/2017, FALSE, Approved, WeareA2B, Standard, medium, medium, 122
8.07, $400.91, 36668
3500, Josy Fleeman, Female, 71, 1967-07-21, 54.00 , NA, Entertainment, Affluent Custome
r, N, No, 17, 14870, 22, 11/01/2017, FALSE, Approved, WeareA2B, Standard, medium, medium, 60
.34, $45.26, 34165
3500, Josy Fleeman, Female, 71, 1967-07-21, 54.00 , NA, Entertainment, Affluent Custome
r, N, No, 17, 19836, 40, 14/03/2017, TRUE, Approved, OHM Cycles, Standard, high, medium, 145
8.17, $874.90, 38750
(base) hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles$

```

Fig-7

8. The output of the final job is placed in the following location.  
/output/output/part-r-00000

```

hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles
(base) hadoop@hadoop-VirtualBox: ~/Downloads/AssignmentFiles$ hdfs dfs -cat /output
t/output/part-r-00000
01/2017, large, no, 175
01/2017, large, yes, 167
01/2017, medium, no, 561
01/2017, medium, yes, 525
01/2017, small, no, 117
01/2017, small, yes, 118
02/2017, large, no, 150
02/2017, large, yes, 147
02/2017, medium, no, 536
02/2017, medium, yes, 535
02/2017, small, no, 117
02/2017, small, yes, 127
03/2017, large, no, 168
03/2017, large, yes, 159
03/2017, medium, no, 533
03/2017, medium, yes, 536
03/2017, small, no, 115
03/2017, small, yes, 118
04/2017, large, no, 151
04/2017, large, yes, 163
04/2017, medium, no, 526
04/2017, medium, yes, 570

```

Fig-8

### 2.3. JAVA Classes Implementation

1. Create the project as shown below figure -9. Four additional jars were used in the development of this project.
  - a. hadoop-common-3.0.3.jar – available at  
</usr/local/hadoop/share/hadoop/common>
  - b. hadoop-mapred-client-core-3.0.3.jar – available at  
</usr/local/hadoop/share/hadoop/mapreduce>



- c. `hadoop-mapreduce-client-jobclient-3.0.3.jar` – available at  
`</usr/local/hadoop/share/hadoop/mapreduce>`
- d. `log4j-1.2.17.jar` – available at  
`</home/Hadoop/Downloads/AssignmentFiles/Jars>`

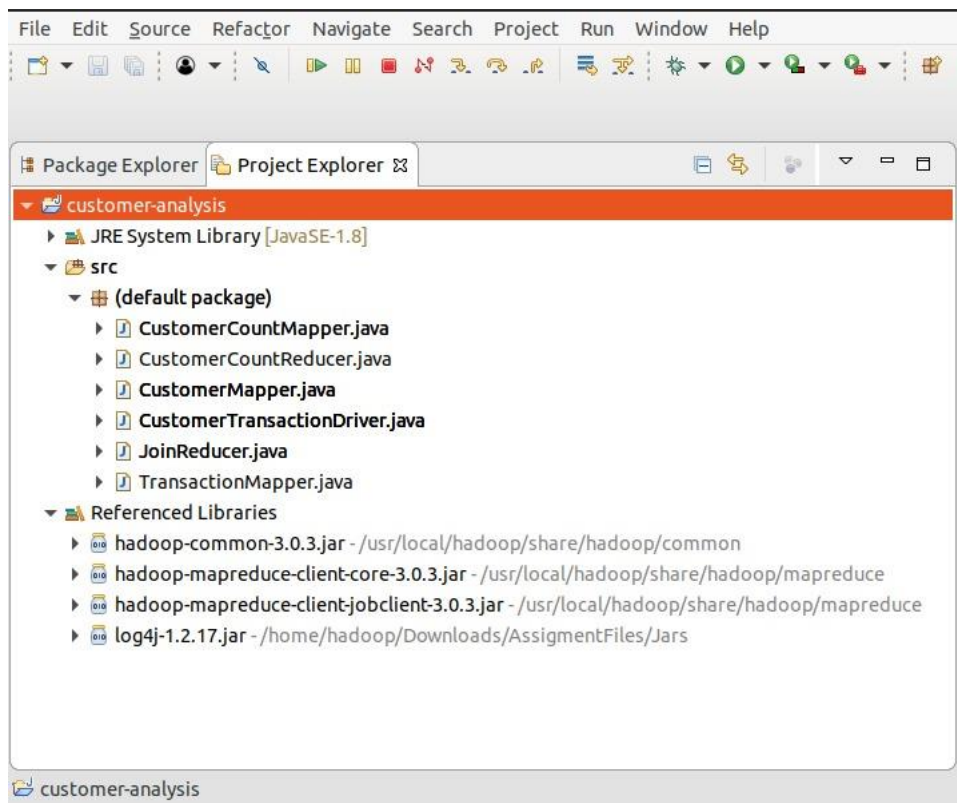


Fig.9

## 2. CustomerMapper

```

CustomerMapper.java
1 import org.apache.hadoop.io.IntWritable;
2 import org.apache.hadoop.io.LongWritable;
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Mapper;
5 import org.apache.log4j.LogManager;
6 import org.apache.log4j.Logger;
7
8 import java.io.IOException;
9 import java.util.Arrays;
10 import java.util.List;
11 import java.util.regex.Pattern;
12 import java.util.stream.Collectors;
13
14 /**
15  *
16  * @author Uma
17  * CustomerMapper class is used read the customer related data from file and
18  * generate key, value pairs
19  *
20  */
21 public class CustomerMapper extends Mapper<LongWritable, Text, IntWritable, Text> {
22
23     private static final Logger _log = LogManager.getLogger(CustomerMapper.class);
24
25     private final Pattern separator = Pattern.compile(",");
26
27     private final IntWritable custIdAsKey = new IntWritable();
28     private final Text custRecordAsValue = new Text();
29
30 }

```

```

29
30- /**
31  * Process input line and write city and count in output
32  * @param offset input line offset
33  * @param line input line
34  * @param context mapper context
35  */
36- @Override
37  protected void map(LongWritable offset, Text line, Context context) throws IOException, InterruptedException {
38      try {
39          // Split input line to get customer id
40          final List<String> columns = Arrays.stream(separator.split(line.toString(), 4)).collect(Collectors.toList());
41
42          try {
43              // Extract customer id from customer record and parse it
44              int customerId = Integer.parseInt(columns.remove(0));
45
46              // Add customer record indicator (will use in reducer)
47              columns.add(0, "C");
48
49              // Write customer id as key and customer record as value
50              custIdAsKey.set(customerId);
51              custRecordAsValue.set(String.join(",", columns));
52              context.write(custIdAsKey, custRecordAsValue);
53
54          } catch (NumberFormatException ignore) {}
55      }
56      catch (Exception e) {
57          log.error("Failed to process record: " + line.toString(), e);
58          throw e;
59      }
60  }
61 }

```

### 3. TransactionMapper

```

TransactionMapper.java
1 import org.apache.hadoop.io.IntWritable;
2 import org.apache.hadoop.io.LongWritable;
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Mapper;
5 import org.apache.log4j.LogManager;
6 import org.apache.log4j.Logger;
7
8 import java.io.IOException;
9 import java.util.Arrays;
10 import java.util.List;
11 import java.util.regex.Pattern;
12 import java.util.stream.Collectors;
13
14- /**
15  *
16  * @author Uma
17  * TransactionMapper class is used read the Transaction related data from file and
18  * generate key, value pairs
19  */
20 public class TransactionMapper extends Mapper<LongWritable, Text, IntWritable, Text> {
21
22     private static final Logger _log = LogManager.getLogger(TransactionMapper.class);
23
24     private final Pattern separator = Pattern.compile(",");
25
26     private final IntWritable custIdAsKey = new IntWritable();
27     private final Text transactionAValue = new Text();
28
29- /**
30  * Process input line and write city and count in output
31  *
32  * @param offset input line offset
33  * @param line input line

```

```

TransactionMapper.java
34  * @param context mapper context
35  */
36  @Override
37  protected void map(LongWritable offset, Text line, Context context) throws IOException, InterruptedException {
38
39      try {
40          // Split input line to get customer id
41          final List<String> columns = Arrays.stream(separator.split(line.toString(), 4)).collect(Collectors.toList());
42
43          try {
44              // Extract customer id from transaction record and convert to integer
45              int customerId = Integer.parseInt(columns.remove(2));
46
47              // Add transaction record indicator (will use in reducer)
48              columns.add(0, "T");
49
50              // Write customer id as key and transaction record as value
51              custIdAsKey.set(customerId);
52              // Converting transaction list values as comma separated string and
53              // setting to transaction value
54              transactionAValue.set(String.join(",", columns));
55              // Writing customer id key and transaction value to output
56              context.write(custIdAsKey, transactionAValue);
57
58          } catch (NumberFormatException ignore) {
59
60          }
61
62      } catch (Exception e) {
63          _log.error("Failed to process record: " + line.toString(), e);
64          throw e;
65      }
66  }

```

#### 4. JoinReducer

```

JoinReducer.java
1  import org.apache.hadoop.io.IntWritable;
2  import org.apache.hadoop.io.Text;
3  import org.apache.hadoop.mapreduce.Reducer;
4
5  import java.io.IOException;
6  import java.util.ArrayList;
7  import java.util.List;
8  import java.util.regex.Pattern;
9
10 /**
11  *
12  * @author Uma
13  * JoinReducer class is used to join the output of the CustomerMapper and TransactionMapper
14  *
15  */
16 public class JoinReducer extends Reducer<IntWritable, Text, IntWritable, Text> {
17
18     private final Pattern separator = Pattern.compile(",");
19     private final Text joinedRecordAsValue = new Text();
20
21
22     @Override
23     protected void reduce(IntWritable custId, Iterable<Text> records, Context context) throws IOException, InterruptedException {
24
25         String customerRecord = null;
26         List<String> customerTransactions = new ArrayList<>();
27
28         // Collect customer records and transactions separately based on flag 'C' & 'T'
29         // respectively at first position of each value
30         for (Text record: records) {
31             String[] typeRecord = separator.split(record.toString(), 2);
32             if ("C".equals(typeRecord[0])) {
33                 customerRecord = typeRecord[1];
34             } else if ("T".equals(typeRecord[0])) {
35                 customerTransactions.add(typeRecord[1]);
36             }
37         }
38
39         if (customerRecord != null) {
40             // Joining customer records with each transaction record
41             for (String transaction : customerTransactions) {
42                 joinedRecordAsValue.set(customerRecord + "," + transaction);
43                 // Writing joined data along with customer id to output
44                 context.write(custId, joinedRecordAsValue);
45             }
46         }
47     }
48 }
49

```

#### 5. CustomerCountMapper

```

CustomerCountMapper.java
1 import org.apache.hadoop.io.IntWritable;
2 import org.apache.hadoop.io.LongWritable;
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Mapper;
5 import org.apache.log4j.LogManager;
6 import org.apache.log4j.Logger;
7 import java.io.IOException;
8 import java.util.regex.Pattern;
9
10 /**
11  *
12  * @author Uma
13  * CustomerCountMapper class is used to obtain the joined output from first MR job
14  * then create key value pairs for each record
15  *
16  */
17 public class CustomerCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
18
19     private static final Logger _log = LogManager.getLogger(CustomerCountMapper.class);
20
21     private final Pattern separator = Pattern.compile(",");
22
23     private final Text KEY = new Text();
24     private final IntWritable VALUE = new IntWritable(1);
25
26     @Override
27     protected void map(LongWritable offset, Text joinedRecord, Context context) throws IOException, InterruptedException {
28
29         try {
30             // Split input line to get customer id
31             final String[] columns = separator.split(joinedRecord.toString().toLowerCase());
32             // Getting Date, Product Size and Car owned information using their positions in file
33             if (!columns[14].isEmpty() && !columns[20].isEmpty() && !columns[10].isEmpty()) {
34
35                 // Getting Date, Product Size and Car owned information using their positions in file
36                 if (!columns[14].isEmpty() && !columns[20].isEmpty() && !columns[10].isEmpty()) {
37                     // Extracting month and year from date
38                     String date = columns[14].split("/", 2)[1];
39                     String prodSize = columns[20];
40                     String ownsCar = columns[10];
41                     //Setting extracted date, product size and owns car as key
42                     KEY.set(date + "," + prodSize + "," + ownsCar);
43
44                     // Write date, product size and owns car as key and 1 as value
45                     context.write(KEY, VALUE);
46                 }
47             }
48         } catch (Exception e) {
49             _log.error("Failed to process record: " + joinedRecord.toString(), e);
50             try {
51                 throw e;
52             } catch (Exception e1) {
53                 e1.printStackTrace();
54             }
55         }
56     }
57 }

```

## 6. CustomerCountReducer



```

CustomerCountReducer.java
1 import org.apache.hadoop.io.IntWritable;
2 import org.apache.hadoop.io.Text;
3 import org.apache.hadoop.mapreduce.Reducer;
4 import java.io.IOException;
5
6 /**
7  *
8  * @author Uma
9  * CustomerCountReducer class is used to aggregate the output obtained from CustomerCount Mapper
10  *
11  */
12 public class CustomerCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
13
14     private final IntWritable COUNT = new IntWritable();
15
16     @Override
17     protected void reduce(Text dateProdTypeOwncar, Iterable<IntWritable> counts, Context context) throws IOException, InterruptedException {
18
19         // Count number of customer transactions
20         int sum = 0;
21         for (IntWritable c : counts) {
22             sum += c.get();
23         }
24         COUNT.set(sum);
25
26         // Writing date, product size and owns car along with count to output
27         context.write(dateProdTypeOwncar, COUNT);
28     }
29 }
30

```

## 7. CustomerTransactionDriver

```

CustomerTransactionDriver.java
1 import org.apache.hadoop.conf.Configuration;
2 import org.apache.hadoop.fs.Path;
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapred.JobConf;
6 import org.apache.hadoop.mapreduce.Job;
7 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
8 import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
9 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
10 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
12 import org.apache.log4j.LogManager;
13 import org.apache.log4j.Logger;
14
15 /**
16  *
17  * @author Uma
18  * CustomerTransactionDriver class is the main driving class used to run the two map reduce jobs
19  * sequentially by taking input files
20  *
21  */
22 public class CustomerTransactionDriver {
23
24     private static final Logger _log = LogManager.getLogger(CustomerTransactionDriver.class);
25
26     public static void main(String[] args) {
27
28         // Displaying error message if user did not pass 3 arguments which are 2 input
29         // paths and 1 output path
30         if (args.length < 3) {
31             _log.error(String.format("Usage: %s <customer-file-path> <transaction-file-path> <output-path>",
32                                     CustomerTransactionDriver.class.getSimpleName()));
33             System.exit(-1);
34         }
35     }
36 }
37

```



# CustomerTransactionDriver.java

```

36     try {
37         // Creating configuration instance and setting comma as mapreduce output separator
38         Configuration conf = new Configuration();
39         conf.set("mapreduce.output.textoutputformat.separator", ",");
40
41         // Creating job configuration instance
42         JobConf jobConf = new JobConf(conf);
43         // Creating job instance with job name
44         Job job = Job.getInstance(jobConf, "Customer Inner Join Transaction Job");
45         job.setJarByClass(CustomerTransactionDriver.class);
46
47         // Setting input paths and map corresponding mapper classes along with input format
48         // CustomerDemographic file will be processed by CustomerMapper
49         // Transactions file will be processed by TransactionMapper
50         MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, CustomerMapper.class);
51         MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, TransactionMapper.class);
52
53         // Appending / at the end of output path if it is not given by user
54         String outputPath = args[2];
55         if (!outputPath.endsWith("/") && !outputPath.endsWith("\\")) {
56             outputPath += '/';
57         }
58
59         // Setting output path
60         FileOutputFormat.setOutputPath(job, new Path(outputPath + "joins/"));
61
62         // Setting output type, format classes and reducer class
63         job.setMapOutputKeyClass(IntWritable.class);
64         job.setMapOutputValueClass(Text.class);
65         job.setOutputKeyClass(IntWritable.class);
66         job.setOutputValueClass(Text.class);
67         job.setOutputFormatClass(TextOutputFormat.class);
68         job.setReducerClass(JoinReducer.class);

```

# CustomerTransactionDriver.java

```

69
70     // Submit first job
71     if (job.waitForCompletion(true)) {
72
73         // Creating configuration instance and setting comma as mapreduce output separator
74         conf = new Configuration();
75         conf.set("mapreduce.output.textoutputformat.separator", ",");
76
77         // Prepare job instance
78         jobConf = new JobConf(conf);
79         // Creating job instance with job names
80         job = Job.getInstance(jobConf, "Count Customers By Month, ProductType, OwnsCar");
81         job.setJarByClass(CustomerTransactionDriver.class);
82
83         // Setting output from first MR job as input for second MR job
84         FileInputFormat.addInputPath(job, new Path(outputPath + "joins/"));
85         FileOutputFormat.setOutputPath(job, new Path(outputPath + "output/"));
86
87         // Setting output type and format classes
88         job.setMapOutputKeyClass(Text.class);
89         job.setMapOutputValueClass(IntWritable.class);
90         job.setOutputKeyClass(Text.class);
91         job.setOutputValueClass(IntWritable.class);
92         job.setOutputFormatClass(TextOutputFormat.class);
93
94         // Setting mapper and reduce implementation classes
95         job.setMapperClass(CustomerCountMapper.class);
96         job.setReducerClass(CustomerCountReducer.class);
97
98         // Start job and wait for complete
99         if (job.waitForCompletion(true)) {
100             // Showing Success message
101             _log.info("Success :");
102         } else {
103             // Showing fail message
104             _log.error("Fail :");
105         }
106     } else {
107         // Showing fail message
108         _log.error("Fail :");
109     }
110     // Exit the process once job is completed
111     System.exit(job.isSuccessful() ? 0 : 1);
112 } catch (Exception e) {
113     _log.error("Error", e);
114     System.exit(1);
115 }
116 }
117 }

```