1) SalesMapper.java

```
package SalesCountry;


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.mapred.*;


public class SalesMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

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


	public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {


		String valueString = value.toString();

		String[] SingleCountryData = valueString.split("-");

		output.collect(new Text(SingleCountryData[0]), one);

	}

}
```

2) SalesCountryReducer.java

```
package SalesCountry;


import java.io.IOException;

import java.util.*;


import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.*;


public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {


	public void reduce(Text t_key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
		Text key = t_key;
		int frequencyForCountry = 0;
		while (values.hasNext()) {
			// replace type of value with the actual type of our value
			IntWritable value = (IntWritable) values.next();
			frequencyForCountry += value.get();


		}
		output.collect(key, new IntWritable(frequencyForCountry));
	}
}
```

3) SalesCountryDriver.java

```
package SalesCountry;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class SalesCountryDriver {
	public static void main(String[] args) {
		JobClient my_client = new JobClient();
		// Create a configuration object for the job
		JobConf job_conf = new JobConf(SalesCountryDriver.class);

		// Set a name of the Job
		job_conf.setJobName("SalePerCountry");

		// Specify data type of output key and value
		job_conf.setOutputKeyClass(Text.class);
		job_conf.setOutputValueClass(IntWritable.class);

		// Specify names of Mapper and Reducer Class
		job_conf.setMapperClass(SalesCountry.SalesMapper.class);
		job_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

		// Specify formats of the data type of Input and output
		job_conf.setInputFormat(TextInputFormat.class);
		job_conf.setOutputFormat(TextOutputFormat.class);

		// Set input and output directories using command line arguments,
		//arg[0] = name of input directory on HDFS, and arg[1] = name of output
directory to be created to store the output file.
```

```
            FileInputFormat.setInputPaths(job_conf,  new Path(args[0]));
            FileOutputFormat.setOutputPath(job_conf,  new Path(args[1]));

            my_client.setConf(job_conf);
            try {
                    // Run the job
                    JobClient.runJob(job_conf);
            } catch (Exception e) {
                    e.printStackTrace();
            }
    }
}
```

OUTPUT:

```
student@mll12:~$ hdfs dfs -put ~/input25 /
student@mll12:~$ hadoop jar myjar.jar /input25 /output25
student@mll12:~$ hdfs dfs -cat /output25/part-00000
10.1.1.236      7
10.1.181.142    14
10.1.232.31     5
10.10.55.142    14
10.102.101.66           1
10.103.184.104          1
10.103.190.81           53
10.103.63.29    1
10.104.73.51    1
10.105.160.183          1
10.108.91.151           1
10.109.21.76    1
10.11.131.40    1
10.111.71.20    8
10.112.227.184          6
10.114.74.30    1
10.115.118.78   1
10.117.224.230          1
10.117.76.22    12
10.118.19.97    1
10.118.250.30   7
10.119.117.132          23
10.119.33.245   1
10.119.74.120   1
10.12.113.198   2
10.12.219.30    1
10.120.165.113          1
10.120.207.127          4
10.123.124.47           1
10.123.35.235           1
10.124.148.99           1
10.124.155.234          1
10.126.161.13           7
10.127.162.239          1
10.128.11.75    10
10.13.42.232    1
10.130.195.163          8
10.130.70.80    1
10.131.163.73           1
10.131.209.116          5
10.132.19.125           2
10.133.222.184          12
10.134.110.196          13
10.134.242.87           1
10.136.84.60    5
```

10.14.2.86 8
10.14.4.151 2
10.140.139.116 1
10.140.141.1 9
10.140.67.116 1
10.141.221.57 5
10.142.203.173 7
10.143.126.177 32
10.144.147.8 1
10.15.208.56 1
10.15.23.44 13
10.150.212.239 14
10.150.227.16 1
10.150.24.40 13
10.152.195.138 8
10.153.23.63 2
10.153.239.5 25
10.155.95.124 9
10.156.152.9 1
10.157.176.158 1
10.164.130.155 1
10.164.49.105 8
10.164.95.122 10
10.165.106.173 14
10.167.1.145 19
10.169.158.88 1
10.170.178.53 1
10.171.104.4 1
10.172.169.53 18
10.174.246.84 3
10.175.149.65 1
10.175.204.125 15
10.177.216.164 6
10.179.107.170 2
10.181.38.207 13
10.181.87.221 1
10.185.152.140 1
10.186.56.126 16
10.186.56.183 1
10.187.129.140 6
10.187.177.220 1
10.187.212.83 1
10.187.28.68 1
10.19.226.186 2
10.190.174.142 10