Dataset

```
Samsung|Optima|14|Madhya Pradesh|132401|14200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Akai|Decent|16|Kerala|922401|12200
Lava|Attention|20|Assam|454601|24200
Zen|Super|14|Maharashtra|619082|9200
Samsung|Optima|14|Madhya Pradesh|132401|14200
Onida|Lucid|18|Uttar Pradesh|232401|16200
Onida|Decent|14|Uttar Pradesh|232401|16200
Onida|NA|16|Kerala|922401|12200
Lava|Attention|20|Assam|454601|24200
Zen|Super|14|Maharashtra|619082|9200
Samsung|Optima|14|Madhya Pradesh|132401|14200
NA|Lucid|18|Uttar Pradesh|232401|16200
Samsung|Decent|16|Kerala|922401|12200
Lava|Attention|20|Assam|454601|24200
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
Samsung|Super|14|Maharashtra|619082|9200
```

Task 1:

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

```
public static class WordCountMapper
     extends Mapper<Object, Text, Text, IntWritable> {

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

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

    // Parse the input string into a nice map
    Map<String, String> parsed = MRDPUtils.transformXmlToMap(value.toString());

    // Grab the "Text" field, since that is what we are counting over
    String txt = parsed.get("Text");

    // .get will return null if the key is not there
    if (txt == “NA”) {
      // skip this record
       return;
    }
}
```

Task 2:

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

SalesMapper.java

```
package Product;

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[] ProductData = valueString.split("|");

                output.collect(new Text(SingleCountryData[7]), one);

        }

}
```

SalesReducer.java

```
package Product;

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 SalesReducer 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 prod_total_sales = 0;
```

```
                while (values.hasNext()) {
                        // replace type of value with the actual type of our value
                        IntWritable value = (IntWritable) values.next();
                        prod_total_sales += value.get();
                }
                output.collect(key, new IntWritable(prod_total_sales));
        }
}
```

SalesCountryDriver.java

```
package Product;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class Product {
        public static void main(String[] args) {
                JobClient my_client = new JobClient();
                // Create a configuration object for the job
                JobConf job_conf = new JobConf(Product.class);
                // Set a name of the Job
                job_conf.setJobName("Sales per product");
                // 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();
		}
	}
}
```

Task 3:

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

// A if validation is used to check and add only the total sales of Onida across each state.

SalesMapper.java

```
package Product;

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[] ProductData = valueString.split("|");

                output.collect(new Text(SingleCountryData[7]), one);

        }

}
```

SalesReducer.java

```
package Product;

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 SalesReducer 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 prod_total_sales = 0;
                while (values.hasNext()) {
                        // replace type of value with the actual type of our value
                        IntWritable value = (IntWritable) values.next();
                        If key == “Onida”
                        {
                        prod_total_sales += value.get();
                        }
                }
                output.collect(key, new IntWritable(prod_total_sales));
        }
}
```

SalesCountryDriver.java

```
package Product;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class Product {
        public static void main(String[] args) {
                JobClient my_client = new JobClient();
                // Create a configuration object for the job
                JobConf job_conf = new JobConf(Product.class);
                // Set a name of the Job
                job_conf.setJobName("Sales for Onida");
                // 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();

		}

	}

}
```