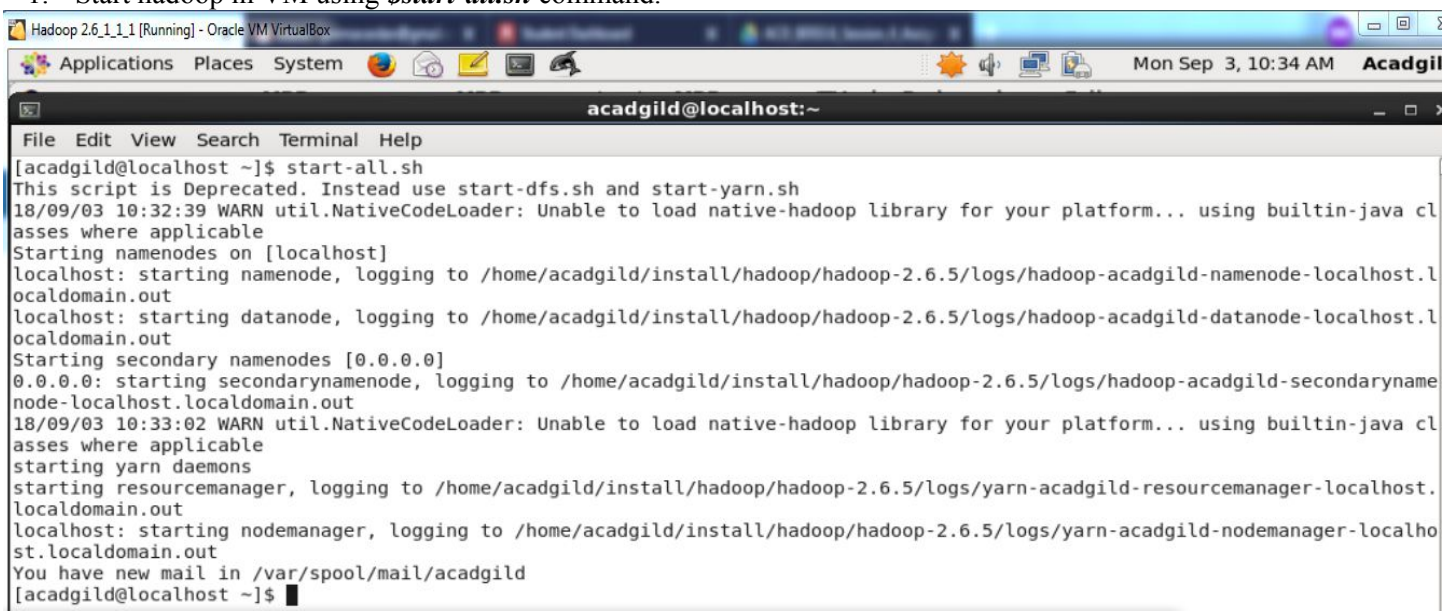
TV Sales

1. Start hadoop in VM using ***$start-all.sh*** command.



2. Check whether all the daemons of hadoop started using the command ***$jps***

We can observe that all the components **ResourceManager, NodeManager, NameNode, DataNode, SecondaryNameNode** are all started properly.

Now to calculate the TVSales using MapReduce, we need write three Java files, ***TVsales.java*** is a driver program which reads the input and calls the Mapper and Reducer to calculate TVsales.

TVsalesMapper.java is a Mapper source code for TV Sales.

TVsalesReducer.java is a Reducer source code for TV Sales.

1. Now the following source code is for ***TVsales.java*** is a driver program.

```
package myMRPrograms;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
```

```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TVsales {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: TVsales <input path> <output path>");
      System.exit(-1);
    }

          //Job Related Configurations
          Configuration conf = new Configuration();
          //Job job = new Job.getInstance(conf, "My TV Sales with combiner");
          Job job = Job.getInstance(conf, "My TV Sales with combiner");
          job.setJarByClass(TVsales.class);


    // Specify the number of reducer to 2
    job.setNumReduceTasks(1);

    //Provide paths to pick the input file for the job
    FileInputFormat.setInputPaths(job, new Path(args[0]));


    //Provide paths to pick the output file for the job, and delete it if already present
          Path outputPath = new Path(args[1]);
          FileOutputFormat.setOutputPath(job, outputPath);
          outputPath.getFileSystem(conf).delete(outputPath, true);


    //To set the mapper and reducer of this job
    job.setMapperClass(TVsalesMapper.class);
    job.setReducerClass(TVsalesReducer.class);

    //Set the combiner
    job.setCombinerClass(TVsalesReducer.class);

    //set the input and output format class
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    //set up the output key and value classes
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    //execute the job
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

2. The following source code is for ***TVsalesMapper.java***.

```
package myMRPrograms;
```

```
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.mapreduce.Mapper;
import java.util.*;

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

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


  @Override
  public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
          String line = value.toString();
          StringTokenizer tokenizer = new StringTokenizer(line);
          if(line.contains("NA") != true) {
                    word.set(line);
             context.write(word, one);
          }
  }
}
```

3. The following source code is for ***TVsalesReducer.java***.

```
package myMRPrograms;
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;


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

  @Override
  public void reduce(Text key, Iterable<IntWritable> values,
     Context context)
     throws IOException, InterruptedException {
     System.out.println("From The Reducer=>"+key) ;

     int sum = 0;
     for (IntWritable value : values) {
                    sum+=value.get();
      }
      context.write(key, new IntWritable(sum));
  }
}
```

Now we have the ***TVsales.java*** driver, ***TVsalesMapper.java*** and ***TVsalesReducer.java***.
Create a jar file from all these source files and name it as ***TVsales.jar*** to calculate TV Sales.

Task 1:
Write a Map Reduce program to filter out the invalid records. Map only job will fir for this context.

1. To stop using the reducer program, we need to make number of reducers to 0 in the ***TVsales.java*** source code.
   Ex: ***// Specify the number of reducer to 0***
   job.setNumReduceTasks(0);
2. This will makes the ***TVsales*** program to run only mapper, but not the reducer program.
3. Now make some changes in the ***TVsalesMapper.java*** source file to filter the records of TVsales with fields NA.
4. To do that change the ***TVsalesMapper.java*** source code as follows:

```
String line = value.toString();
if(line.contains("NA") != true) {
        word.set(line);
   context.write(word, one);
}
```

5. You can use the following source code with the above changes.

```
package myMRPrograms;
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.mapreduce.Mapper;
import java.util.*;

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

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


  @Override
  public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
     String line = value.toString();
     if(line.contains("NA") != true) {
             word.set(line);
        context.write(word, one);
     }
  }
}
```

6. Now run the ***TVsales.jar*** file with input file ***tele.txt*** which contains the sales records of different companies. This can be run by running the command: ***$hadoop jar TVsales.jar /tele.txt /output***
7. Here ***/output*** is the folder which contains the result of the TVsales records.

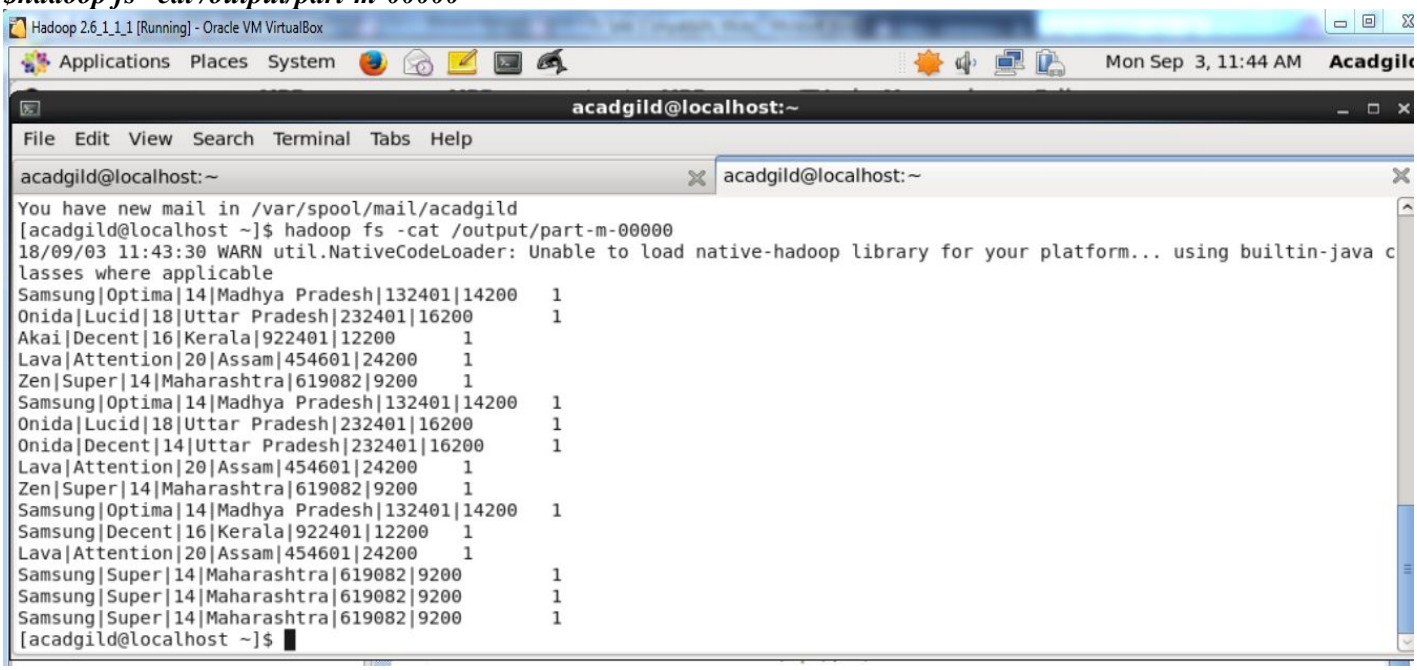
```
[acadgild@localhost ~]$ hadoop jar TVsales.jar /tele.txt /output
18/09/03 11:31:55 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
asses where applicable
18/09/03 11:31:57 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/09/03 11:31:59 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool i
terface and execute your application with ToolRunner to remedy this.
18/09/03 11:32:00 INFO input.FileInputFormat: Total input paths to process : 1
18/09/03 11:32:01 INFO mapreduce.JobSubmitter: number of splits:1
18/09/03 11:32:02 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1535950987975_0001
18/09/03 11:32:03 INFO impl.YarnClientImpl: Submitted application application_1535950987975_0001
18/09/03 11:32:03 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1535950987975_0001/
18/09/03 11:32:03 INFO mapreduce.Job: Running job: job_1535950987975_0001
18/09/03 11:32:20 INFO mapreduce.Job: Job job_1535950987975_0001 running in uber mode : false
18/09/03 11:32:20 INFO mapreduce.Job:  map 0% reduce 0%
18/09/03 11:32:35 INFO mapreduce.Job:  map 100% reduce 0%
18/09/03 11:32:36 INFO mapreduce.Job: Job job_1535950987975_0001 completed successfully
18/09/03 11:32:36 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=107687
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=828
                HDFS: Number of bytes written=678
                HDFS: Number of read operations=5
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=11088
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=11088
                Total vcore-milliseconds taken by all map tasks=11088
```

```
                Total megabyte-milliseconds taken by all map tasks=11354112
        Map-Reduce Framework
                Map input records=18
                Map output records=16
                Input split bytes=95
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=142
                CPU time spent (ms)=1070
                Physical memory (bytes) snapshot=163139584
                Virtual memory (bytes) snapshot=2075860992
                Total committed heap usage (bytes)=104857600
        File Input Format Counters
                Bytes Read=733
        File Output Format Counters
                Bytes Written=678
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

8. Now check the ***/output*** folder for status and output of the program. This can be done by running the command: ***$hadoop fs –ls /output.***

```
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -ls /output
18/09/03 11:39:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
Found 2 items
-rw-r--r--   1 acadgild supergroup          0 2018-09-03 11:32 /output/_SUCCESS
-rw-r--r--   1 acadgild supergroup        678 2018-09-03 11:32 /output/part-m-00000
[acadgild@localhost ~]$
```

9. Here you can observe that the ***_SUCCESS*** flag indicates that the process ran successfully. And ***part-m-00000*** represents the Mapper outputput, the ***–m-*** stands for mapper.
10. Now open the file ***/output/part-m-00000*** to see the actual output. This can be done with the command:
    $hadoop fs –cat /output/part-m-00000



11. We can observe here that the filtered records, which does not have NA fields in the records.

Task 2:
Write a Map Reduce program to calculate the total units sold for each Company.

1. Now to do this task, we need to enable the Reducer and makes number of reducer as 1. This can be done by changing the ***TVsales.java*** source code as follows:

Ex:
```
// Specify the number of reducer to 1
job.setNumReduceTasks(1);
```

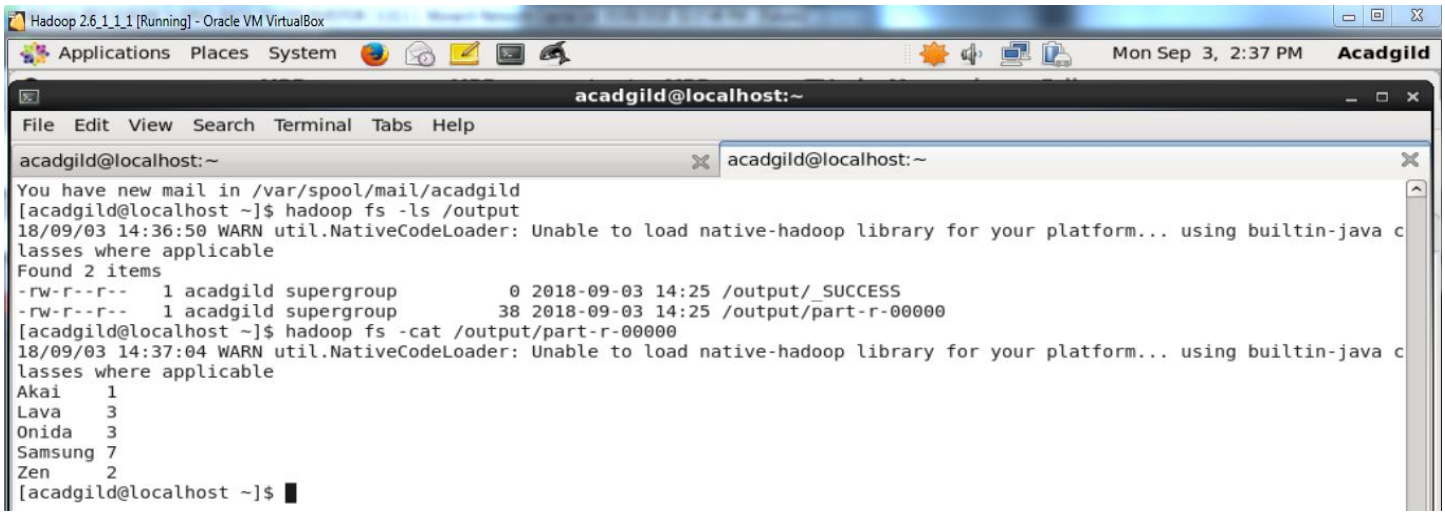
2. Now change the Mapper source code to make suitable for the task as follows:
```
StringTokenizer tokenizer = new StringTokenizer(line);
if(line.contains("NA") != true) {
        tokenizer.nextToken("|");
        word.set(tokenizer.nextToken("|");
       context.write(word, one);
```

3. We can use the following Mapper program with the above change:
```
package myMRPrograms;
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.mapreduce.Mapper;
import java.util.*;

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

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


  @Override
  public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        if(line.contains("NA") != true) {
                word.set(tokenizer.nextToken("|");
               context.write(word, one);
          }
}
```
4. Now create the jar file ***TVsales.jar*** with the three files ***TVsales.java, TVsalesMapper.java*** and ***TVsalesReducer.java***.

5. Now run the jar file with the input as TV sales and followed by an output directory as:
```
$hadoop jar TVsales.jar /tele.txt /output
```

```
[acadgild@localhost ~]$ hadoop jar TVsales.jar /tele.txt /output
18/09/03 14:13:04 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
18/09/03 14:13:05 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/09/03 14:13:07 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool i
nterface and execute your application with ToolRunner to remedy this.
18/09/03 14:13:07 INFO input.FileInputFormat: Total input paths to process : 1
18/09/03 14:13:07 INFO mapreduce.JobSubmitter: number of splits:1
18/09/03 14:13:07 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1535950987975_0002
18/09/03 14:13:08 INFO impl.YarnClientImpl: Submitted application application_1535950987975_0002
18/09/03 14:13:08 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1535950987975_0002/
18/09/03 14:13:08 INFO mapreduce.Job: Running job: job_1535950987975_0002
18/09/03 14:13:20 INFO mapreduce.Job: Job job_1535950987975_0002 running in uber mode : false
18/09/03 14:13:20 INFO mapreduce.Job:  map 0% reduce 0%
18/09/03 14:13:28 INFO mapreduce.Job:  map 100% reduce 0%
18/09/03 14:13:38 INFO mapreduce.Job:  map 100% reduce 100%
18/09/03 14:13:39 INFO mapreduce.Job: Job job_1535950987975_0002 completed successfully
18/09/03 14:13:39 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=72
                FILE: Number of bytes written=215791
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=828
                HDFS: Number of bytes written=46
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=6762
                Total time spent by all reduces in occupied slots (ms)=6657
                Total megabyte-milliseconds taken by all reduce tasks=6816768
        Map-Reduce Framework
                Map input records=18
                Map output records=16
                Map output bytes=178
                Map output materialized bytes=72
                Input split bytes=95
                Combine input records=16
                Combine output records=5
                Reduce input groups=5
                Reduce shuffle bytes=72
                Reduce input records=5
                Reduce output records=5
                Spilled Records=10
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=208
                CPU time spent (ms)=2570
                Physical memory (bytes) snapshot=413507584
                Virtual memory (bytes) snapshot=4160065536
                Total committed heap usage (bytes)=287309824
        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=733
        File Output Format Counters
                Bytes Written=46
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ clear
```

6. Now we see the output from the directory ***/output***.



7. We can observe the output as ***/output/_SUCCESS*** and the reducer output present in the file ***/output/part-r-00000***.
8. And also we can see the number of units which are sold per company. Here the output is presented as the records with 'NA' are filtered out.

1. Now we can do the same process with all the records in the TV sales file without filtering any records.
2. For this case we no need to change the ***TVsales.java file*** but we need to change the logic in the ***TVsalesMapper.java*** file.
3. We can use the following mapper program for this purpose.

```
package myMRPrograms;
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.mapreduce.Mapper;
import java.util.*;

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

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


  @Override
  public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
                String line = value.toString();
                StringTokenizer tokenizer = new StringTokenizer(line);
                word.set(tokenizer.nextToken("|");
                 context.write(word, one);
     }
}
```

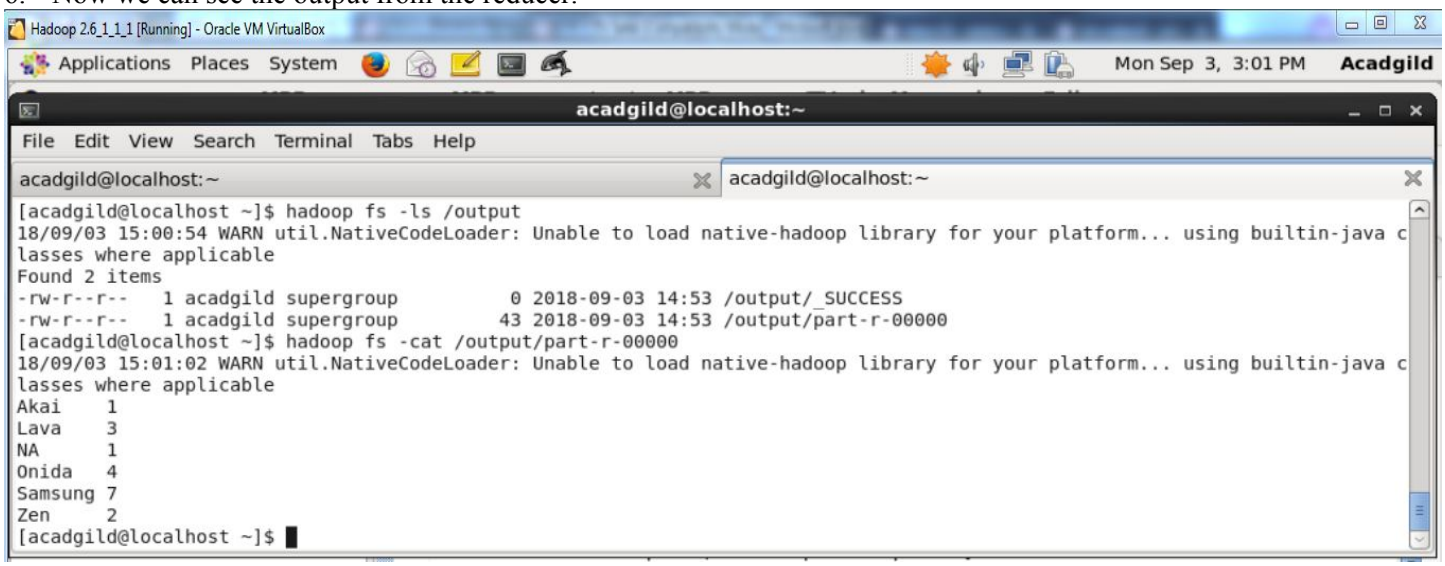
4. Create the jar ***TVsales.jar*** file again with the above changes from the files ***TVsales.java, TVsalesMapper.java*** and ***TVsalesReducer.java.***
5. Now run the jar file with the following command:

Ex: ***$hadoop jar TVsales.jar /tele.txt /output***

```
[acadgild@localhost ~]$ hadoop jar TVsales.jar /tele.txt /output
18/09/03 14:53:28 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
18/09/03 14:53:29 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/09/03 14:53:30 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool i
nterface and execute your application with ToolRunner to remedy this.
18/09/03 14:53:30 INFO input.FileInputFormat: Total input paths to process : 1
18/09/03 14:53:30 INFO mapreduce.JobSubmitter: number of splits:1
18/09/03 14:53:31 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1535950987975_0004
18/09/03 14:53:31 INFO impl.YarnClientImpl: Submitted application application_1535950987975_0004
18/09/03 14:53:31 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1535950987975_0004/
18/09/03 14:53:31 INFO mapreduce.Job: Running job: job_1535950987975_0004
18/09/03 14:53:42 INFO mapreduce.Job: Job job_1535950987975_0004 running in uber mode : false
18/09/03 14:53:42 INFO mapreduce.Job:  map 0% reduce 0%
18/09/03 14:53:49 INFO mapreduce.Job:  map 100% reduce 0%
18/09/03 14:53:57 INFO mapreduce.Job:  map 100% reduce 100%
18/09/03 14:53:58 INFO mapreduce.Job: Job job_1535950987975_0004 completed successfully
18/09/03 14:53:58 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=73
                FILE: Number of bytes written=215793
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=828
                HDFS: Number of bytes written=43
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=4973
                Total time spent by all reduces in occupied slots (ms)=5279
```

```
                Total megabyte-milliseconds taken by all reduce tasks=5405696
        Map-Reduce Framework
                Map input records=18
                Map output records=18
                Map output bytes=183
                Map output materialized bytes=73
                Input split bytes=95
                Combine input records=18
                Combine output records=6
                Reduce input groups=6
                Reduce shuffle bytes=73
                Reduce input records=6
                Reduce output records=6
                Spilled Records=12
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=232
                CPU time spent (ms)=2160
                Physical memory (bytes) snapshot=423428096
                Virtual memory (bytes) snapshot=4165091328
                Total committed heap usage (bytes)=293076992
        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=733
        File Output Format Counters
                Bytes Written=43
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$ hadoop fs -cat /output/part-r-00000
```

6. Now we can see the output from the reducer.



9. In the above output we can see that how many units sold per each company. And we can also see that There is a unit sold without a company name (NA).

Task 3:
Write a Map Reduce program to calculate the total units sold in each state for Onida company.

1. Now for this task we no need to change the **TVsales.java** program.
2. But for the Mapper program we need to do some changes.

```
StringTokenizer tokenizer = new StringTokenizer(line);
if((line.contains("Onida") == true) && (line.contains("NA") != true)) {
   for(int i = 0; i < 3; i++)
      tokenizer.nextToken("|");
   word.set(tokenizer.nextToken("|"));
   context.write(word, one);
}
```

3. We can use the following source code for Mapper program:

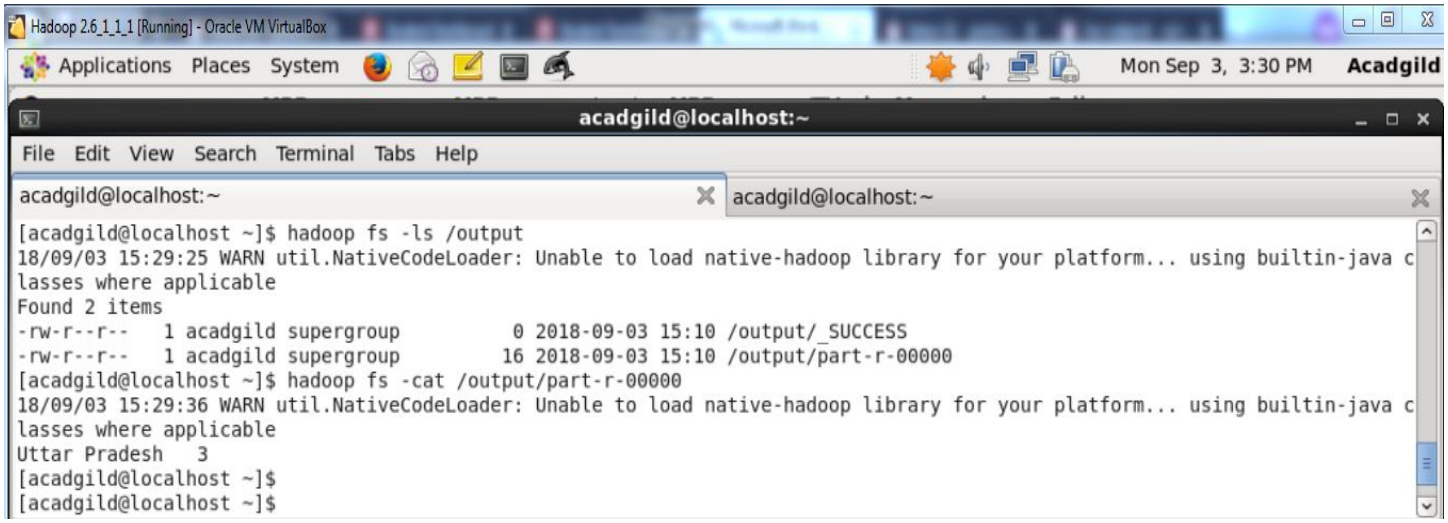
```
package myMRPrograms;
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.util.*;

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

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


  @Override
  public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
     String line = value.toString();
     StringTokenizer tokenizer = new StringTokenizer(line);
     if((line.contains("Onida") == true) && (line.contains("NA") != true)) {
              for(int i = 0; i < 3; i++)
                       tokenizer.nextToken("|");
              word.set(tokenizer.nextToken("|"));
              context.write(word, one);
              }
     }
}
```

4. Now create a new jar with name ***TVsales.jar*** from ***TVsales.java, TVsalesMapper.java*** and ***TVsalesReducer.java.***
5. Run the jar file with the command as follows:

```
$hadoop jar TVsales.jar /tele.txt /output
```

```
[acadgild@localhost ~]$ hadoop jar TVsales.jar /tele.txt /output
18/09/03 15:09:58 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
18/09/03 15:09:59 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/09/03 15:10:00 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool i
nterface and execute your application with ToolRunner to remedy this.
18/09/03 15:10:01 INFO input.FileInputFormat: Total input paths to process : 1
18/09/03 15:10:01 INFO mapreduce.JobSubmitter: number of splits:1
18/09/03 15:10:01 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1535950987975_0005
18/09/03 15:10:01 INFO impl.YarnClientImpl: Submitted application application_1535950987975_0005
18/09/03 15:10:01 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1535950987975_0005/
18/09/03 15:10:01 INFO mapreduce.Job: Running job: job_1535950987975_0005
18/09/03 15:10:11 INFO mapreduce.Job: Job job_1535950987975_0005 running in uber mode : false
18/09/03 15:10:11 INFO mapreduce.Job:  map 0% reduce 0%
18/09/03 15:10:17 INFO mapreduce.Job:  map 100% reduce 0%
18/09/03 15:10:25 INFO mapreduce.Job:  map 100% reduce 100%
18/09/03 15:10:25 INFO mapreduce.Job: Job job_1535950987975_0005 completed successfully
18/09/03 15:10:25 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=26
                FILE: Number of bytes written=215699
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=828
                HDFS: Number of bytes written=16
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=4256
        Map-Reduce Framework
                Map input records=18
                Map output records=3
                Map output bytes=54
                Map output materialized bytes=26
                Input split bytes=95
                Combine input records=3
                Combine output records=1
                Reduce input groups=1
                Reduce shuffle bytes=26
                Reduce input records=1
                Reduce output records=1
                Spilled Records=2
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=187
                CPU time spent (ms)=2240
                Physical memory (bytes) snapshot=424951808
                Virtual memory (bytes) snapshot=4165701632
                Total committed heap usage (bytes)=294649856
        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=733
        File Output Format Counters
                Bytes Written=16
You have new mail in /var/spool/mail/acadgild
[acadgild@localhost ~]$
```

6. Now we see the output of the Reducer program.



7. Now we can see that the number of units sold for Onida company in Uttar Pradesh as 3.
8. Note that the output provided here is a filtered output with fields NA.

1. Now we do the same process with removing the record fields NA.
2. Nothing to change in the ***TVsales.java program and TVsalesReducer.java*** programs.
3. But only change required in the ***TVsalesMapper.java*** program. The changes are as follows:

```
StringTokenizer tokenizer = new StringTokenizer(line);
if(line.contains("Onida") == true) {
   for(int i = 0; i < 3; i++)
      tokenizer.nextToken("|");
   word.set(tokenizer.nextToken("|"));
   context.write(word, one);
}
```

9. We can use the following source code for Mapper program:

```
package myMRPrograms;
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.mapreduce.Mapper;
import java.util.*;

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

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


  @Override
  public void map(LongWritable key, Text value, Context context)
     throws IOException, InterruptedException {
          String line = value.toString();
     StringTokenizer tokenizer = new StringTokenizer(line);
     if(line.contains("Onida") == true) {
              for(int i = 0; i < 3; i++)
                              tokenizer.nextToken("|");
          word.set(tokenizer.nextToken("|"));
          context.write(word, one);
```

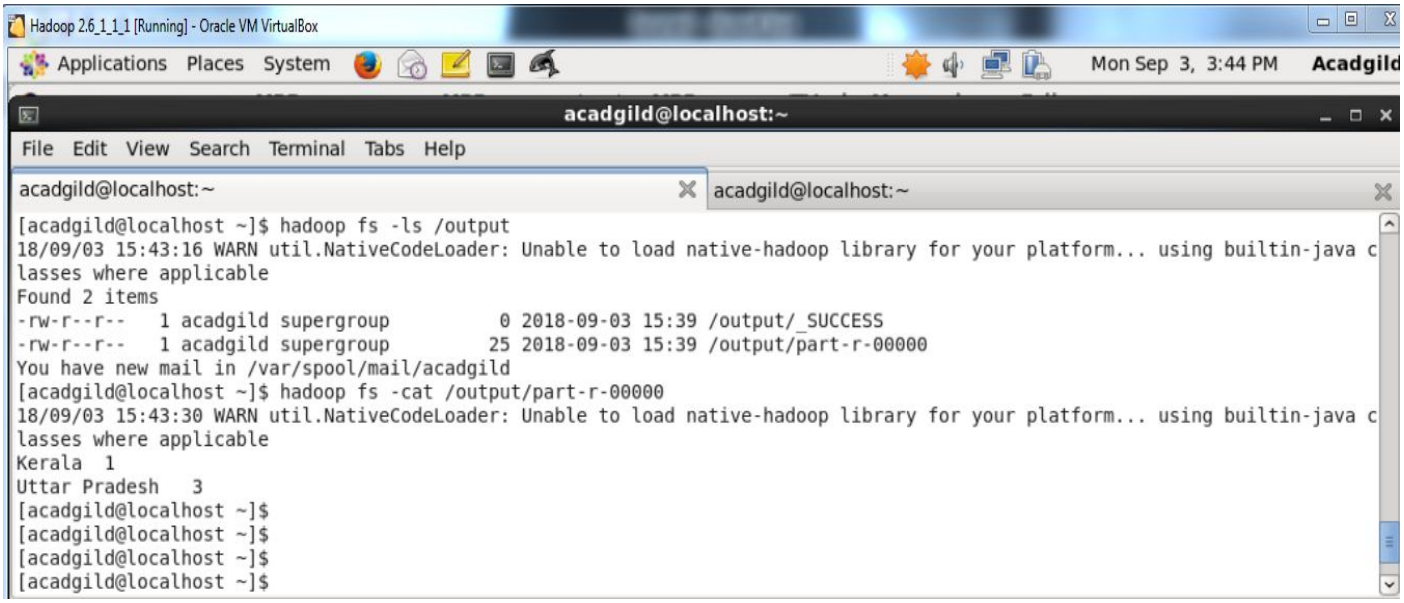
```
        }
    }
}
```

4. Now create a jar file with ***TVsales.jar*** with the three files ***TVsales.java, TVsalesMapper.java*** and ***TVsalesReducer.java.***
5. Run the jar file with the input file as ***tele.txt*** and output directory as ***/output.***
   Ex: ***$hadoop jar TVsales.jar /tele.txt /output.***

```
Hadoop 2.6_1_1_1 [Running] - Oracle VM VirtualBox
Applications  Places  System                                   Mon Sep 3, 3:40 PM  Acadgild
acadgild@localhost:~
File  Edit  View  Search  Terminal  Tabs  Help
acadgild@localhost:~   acadgild@localhost:~
[acadgild@localhost ~]$ hadoop jar TVsales.jar /tele.txt /output
18/09/03 15:38:57 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
lasses where applicable
18/09/03 15:38:58 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
18/09/03 15:38:59 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool i
nterface and execute your application with ToolRunner to remedy this.
18/09/03 15:38:59 INFO input.FileInputFormat: Total input paths to process : 1
18/09/03 15:38:59 INFO mapreduce.JobSubmitter: number of splits:1
18/09/03 15:38:59 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1535950987975_0006
18/09/03 15:39:00 INFO impl.YarnClientImpl: Submitted application application_1535950987975_0006
18/09/03 15:39:00 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1535950987975_0006/
18/09/03 15:39:00 INFO mapreduce.Job: Running job: job_1535950987975_0006
18/09/03 15:39:09 INFO mapreduce.Job: Job job_1535950987975_0006 running in uber mode : false
18/09/03 15:39:09 INFO mapreduce.Job:  map 0% reduce 0%
18/09/03 15:39:16 INFO mapreduce.Job:  map 100% reduce 0%
18/09/03 15:39:24 INFO mapreduce.Job:  map 100% reduce 100%
18/09/03 15:39:24 INFO mapreduce.Job: Job job_1535950987975_0006 completed successfully
18/09/03 15:39:25 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=39
                FILE: Number of bytes written=215725
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=828
                HDFS: Number of bytes written=25
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=4695
```

```
                Total megabyte-milliseconds taken by all reduce tasks=4572160
        Map-Reduce Framework
                Map input records=18
                Map output records=4
                Map output bytes=65
                Map output materialized bytes=39
                Input split bytes=95
                Combine input records=4
                Combine output records=2
                Reduce input groups=2
                Reduce shuffle bytes=39
                Reduce input records=2
                Reduce output records=2
                Spilled Records=4
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=178
                CPU time spent (ms)=2240
                Physical memory (bytes) snapshot=421728256
                Virtual memory (bytes) snapshot=4165349376
                Total committed heap usage (bytes)=292552704
        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=733
        File Output Format Counters
                Bytes Written=25
[acadgild@localhost ~]$
```

6. Now we see the output of the reducer process. This can be done with the command as follows:
   $hadoop fs –ls /output
   $hadoop fs –cat /output/part-r-00000



7. Now we can observe that there are totally 4 units for the company Onida which were sold in **Kerala** and **Uttar Pradesh.**
8. Note here the output with no filter for fields in the record with NA.