*Big Data and Hadoop Development*

**1.Introduction**

In this assignment you need to perform the tasks given.

**2.Objective**

This assignment will help you to consolidate the concepts learnt in the session 4.

**3.Prerequisites:**

None

**4.Associated Data Files**

https://drive.google.com/file/d/0Bxr27gVaXO5sVjQ5QW0wQ3RCTUU/view?usp=sharing

**5.Problem Statement**

We have a dataset of sales of different TV sets across different locations.

Records look like:

Samsung|Optima|14|Madhya Pradesh|132401|14200

The fields are arranged like:

Company Name|ProductName|Size in inches|State|PinCode|Price

There are some invalid records which contain 'NA' in either Company Name or Product Nam

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

**Driver class:**

import java.io.IOException;

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;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapred.JobConf;

public class OnidaUnitExample {

public static void main(String[] args) throws Exception {

if (args.length != 2) {

System.err.println("Usage: WordCount <input path> <output path>");

System.exit(-1);

}

Configuration conf = new Configuration();

Job job = new Job(conf, "calculate the total units sold for each Company");

job.setJarByClass(OnidaUnitExample.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setMapperClass(OnidaUnitMapper.class);

job.setReducerClass(OnidaUnitReducer.class);

job.setCombinerClass(OnidaUnitReducer.class);

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

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

System.exit(job.waitForCompletion(true)? 0: 1);

}

}

**Mapper class**

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;

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

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

{

String[] lineArray = value.toString().split("\\|");

Text company = new Text(lineArray[0]);

String common = company.toString();

if(!common.equals("NA"))

{

context.write(company, new IntWritable(1));

}

}

}

**Reducer class**

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

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

@Override

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

{

int totalNumber =0;

for(IntWritable value : values)

{

totalNumber = totalNumber + value.get();

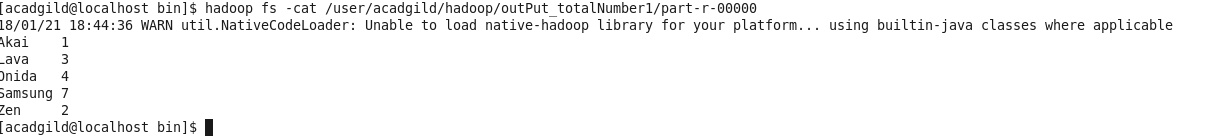
}

context.write(key, new IntWritable(totalNumber));

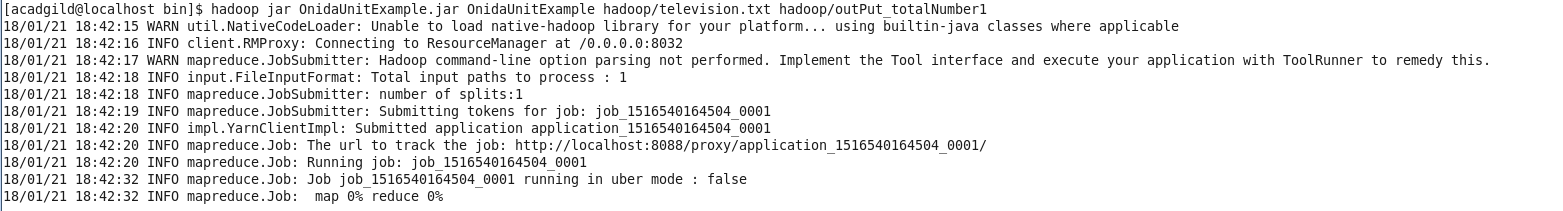
}

}

**Output:**



**Execution step:**



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

company.

**Driver class**

import java.io.IOException;

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;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapred.JobConf;

public class OnidaUnitExample {

public static void main(String[] args) throws Exception {

if (args.length != 2) {

System.err.println("Usage: WordCount <input path> <output path>");

System.exit(-1);

}

Configuration conf = new Configuration();

Job job = new Job(conf, "calculate the total Onida sold for each state");

job.setJarByClass(OnidaUnitExample.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setMapperClass(OnidaUnitMapper.class);

job.setReducerClass(OnidaUnitReducer.class);

job.setCombinerClass(OnidaUnitReducer.class);

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

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

System.exit(job.waitForCompletion(true)? 0: 1);

}

}

**Mapper class**

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;

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

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

{

String[] lineArray = value.toString().split("\\|");

Text company = new Text(lineArray[0]);

Text state = new Text(lineArray[3]);

String common = company.toString();

String commonState = state.toString();

if(common.equals("Onida"))

{

context.write(company, new IntWritable(1));

}

}

}

**Reducer class**

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

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

@Override

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

{

int totalNumber =0;

for(IntWritable value : values)

{

totalNumber = totalNumber + value.get();

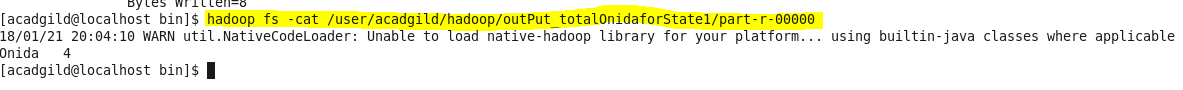
}

context.write(key, new IntWritable(totalNumber));

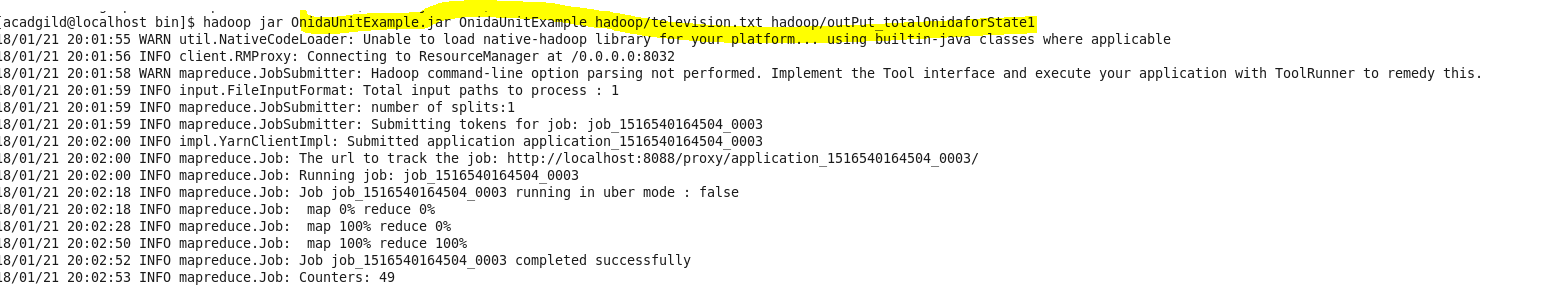
}

}

**output**

****

**Steps of execution**

****