**Problem:**

10. Enhance the task 8 (refer session 6, assignment 1) to calculate the top 3 state-wise sales for each company.

You may use multiple reducers for this activity.

**Task2.java**

job.setJarByClass(Task2.**class**);

job.setMapOutputKeyClass(Text.**class**);

job.setMapOutputValueClass(StateSales.**class**);

job.setOutputKeyClass(Text.**class**);

job.setOutputValueClass(StateSales.**class**);

job.setMapperClass(Task2Mapper.**class**);

job.setReducerClass(Task2Reducer.**class**);

job.setInputFormatClass(TextInputFormat.**class**);

job.setOutputFormatClass(TextOutputFormat.**class**);

**Task1Mapper.java**

this mapper will put company as key and value would be custom object which contains state and count of units sold.

**public** **void** map(LongWritable key, Text value, Context context)

**throws** IOException, InterruptedException {

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

outKey.set(lineArray[0]);

context.write(outKey, **new** StateSales(lineArray[1], Integer.*parseInt*(lineArray[2])));

}

**Task1Reducer.java**

This reducer will sort the values based on state and only write top 3 units sold in state per company.

**public** **class** Task2Reducer **extends** Reducer<Text, StateSales, Text, StateSales>

{

ArrayList<StateSales> stateSalesData = **new** ArrayList<StateSales>();

**public** **void** reduce(Text key, Iterable<StateSales> values,Context context) **throws** IOException, InterruptedException

{

stateSalesData.clear();

**for** (StateSales value : values) {

//value is a single object containing different instances in different passes.

//So, create new StateSales objects every time we need to store it to the ArrayList

stateSalesData.add(**new** StateSales(value.getState(), value.getSales()));

}

Collections.*sort*(stateSalesData);

**int** counter = 1;

**for** (StateSales statesales : stateSalesData) {

**if** (counter > 3) {

**break**;

}

context.write(key, statesales);

counter++;

}

}

}

**StateSales.java**

Stores state and sales count

@Override

**public** String toString() {

**return** state + "\t" + sales;

}

@Override

**public** **int** compareTo(StateSales ss) {

**return** (-1) \* (sales - ss.getSales());

}

@Override

**public** **boolean** equals(Object o)

{

**if**(o **instanceof** StateSales)

{

StateSales ss = (StateSales) o;

**return** state.equalsIgnoreCase(ss.getState()) && (sales == ss.getSales());

}

**return** **false**;

}

**Output:**

hadoop jar S6\_A3.jar mapreduce.demo.task2.Task2 /niki\_output/A6\_3\_1/part-r-00000 /niki\_output/A6\_3\_2

