Problem:

8. Write a Map Reduce program that takes the output of Task 5 (refer session 5, assignment 1) as input, and produce output which is sorted on the total units sold.

You may use a single reducer for the sorting.

Use Sequence File formats as an output for Task 5 and as input for this task.

**First Mapreducet job code**:

**Task1.java**

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

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

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

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

job.setMapperClass(Task1Mapper.**class**);

job.setReducerClass(Task1Reducer.**class**);

job.setCombinerClass(Task1Reducer.**class**);

job.setPartitionerClass(Task4Partitioner.**class**);

job.setNumReduceTasks(4);

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

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

**Task1\_Mapper.java**

**public** **class** Task1Mapper **extends** Mapper<LongWritable, Text, Text, IntWritable> {

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

**throws** IOException, InterruptedException {

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

**if**(!("NA".equalsIgnoreCase(lineArray[0])) &&!("NA".equalsIgnoreCase(lineArray[1])))

{

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

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

}

}

}

**Task1\_Reducer.java**

**public** **class** Task1Reducer **extends** Reducer<Text, IntWritable, Text, IntWritable>

{

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

{

**int** unitSold=0;

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

unitSold=unitSold+value.get();

}

context.write(key, **new** IntWritable(unitSold));

}

}

**Task4Partitioner.java**

**public** **class** Task4Partitioner **extends** Partitioner<Text, IntWritable> {

@Override

**public** **int** getPartition(Text key, IntWritable value, **int** numPartitions) {

String word = key.toString();

**char** letter = word.toLowerCase().charAt(0);

**int** partitionNumber = 0;

**switch**(letter) {

**case** 'a':

**case** 'b':

**case** 'c':

**case** 'd':

**case** 'e':

**case** 'f': partitionNumber = 0; **break**;

**case** 'g':

**case** 'h':

**case** 'i':

**case** 'j':

**case** 'k':

**case** 'l': partitionNumber = 1; **break**;

**case** 'm':

**case** 'n':

**case** 'o':

**case** 'p':

**case** 'q':

**case** 'r': partitionNumber = 2; **break**;

**default**: partitionNumber = 3; **break**;

}

**return** partitionNumber;

}

}

**Mapreduce job 2**

**Task2.java**

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

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

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

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

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

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

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

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

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

**TaskMapper.java**

**public** **class** TaskMapper **extends** Mapper<Text, IntWritable, IntWritable,Text> {

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

**throws** IOException, InterruptedException {

context.write(value, key);

}

}

**TaskReducer.java**

**public** **class** TaskReducer **extends** Reducer< IntWritable,Text, Text, IntWritable>

{

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

{

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

context.write(value, key);

}

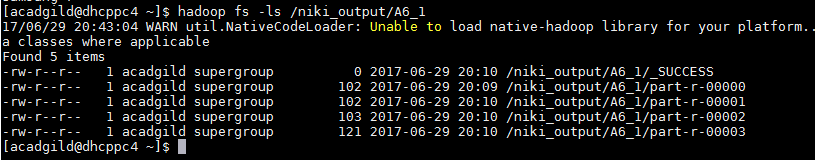
}

}

**Output:**

1. hadoop jar S6\_A1.jar mapreduce.demo.task1.Task1 /niki/television.txt /niki\_output/A6\_1

This will generate 4 files because we have used partiioner and 4 reducer. It will count the unit sold according to company



1. hadoop jar S6\_A1.jar mapreduce.demo.task2.Task2 /niki\_output/A6\_1 /niki\_output/A6\_1\_1

It will sort the unitsold in ascending order

