

Aim: Write MapReduce code to find countrywise and payment method wise total sales.

Country-wise

import java.io.IOException;

import java.util.Iterator;

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.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

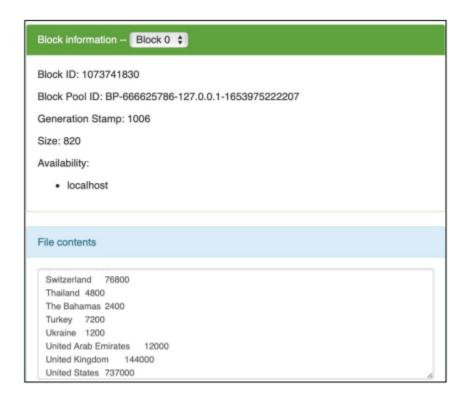
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

 $import\ org. a pache. hado op. mapreduce. lib. output. File Output Format;$

import org.apache.hadoop.io.LongWritable;

```
public class SalesCountryDriver {
public static class SalesMapper
extends Mapper<LongWritable, Text, Text, IntWritable> {
private final static IntWritable one = new IntWritable(1);
private final Text word = new Text();
// number of time mapper runs = no. of rows in csv file , done by recorder
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
if (\text{key.get}() == 0) {
return;
} else {
String valueString = value.toString();
String[] SingleCountryData = valueString.split(",");
context.write(new Text(SingleCountryData[7]), new
IntWritable(Integer.parseInt(SingleCountryData[2])));
}
}
}
public static class SalesCountryReducer extends Reducer<Text,
IntWritable, Text, IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text t_key, Iterable<IntWritable> values,
Context context) throws IOException, InterruptedException {
```

```
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(t key, result);
}
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "Sale per Country");
job.setJarByClass(SalesCountryDriver.class);
job.setMapperClass(SalesMapper.class);
job.setCombinerClass(SalesCountryReducer.class);
job.setReducerClass(SalesCountryReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[1]));
FileOutputFormat.setOutputPath(job, new Path(args[2]));
System.exit(job.waitForCompletion(true)?0:1);
}
```



Payment-wise

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.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.io.LongWritable;

```
public class SalesPaymentDriver {
public static class SalesMapper
extends Mapper<LongWritable, Text, Text, IntWritable> {
private final static IntWritable one = new IntWritable(1);
private final Text word = new Text();
// number of time mapper runs = no. of rows in csv file , done
by recorder
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
if (\text{key.get}() == 0) {
return;
} else {
String valueString = value.toString();
String[] SingleCountryData = valueString.split(",");
context.write(new Text(SingleCountryData[3]), new
IntWritable(Integer.parseInt(SingleCountryData[2])));
}
}
}
public static class SalesPaymentReducer extends Reducer<Text,
IntWritable, Text, IntWritable> {
private IntWritable result = new IntWritable();
public void reduce(Text t_key, Iterable<IntWritable> values,
```

```
Context context) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values) {
sum += val.get();
}
result.set(sum);
context.write(t_key, result);
}
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "Sale per Payment");
job.setJarByClass(SalesPaymentDriver.class);
job.setMapperClass(SalesMapper.class);
job.setCombinerClass(SalesPaymentReducer.class);
job.setReducerClass(SalesPaymentReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[1]));
FileOutputFormat.setOutputPath(job, new Path(args[2]));
System.exit(job.waitForCompletion(true)?0:1);
}
```

Block information -- Block 0 \$

Block ID: 1073741831

Block Pool ID: BP-666625786-127.0.0.1-1653975222207

Generation Stamp: 1007

Size: 56

Availability:

localhost

File contents

Amex 188900 Diners 133800 Mastercard 458450

Visa 836350