|  |  |
| --- | --- |
|  | |
| import java.io.IOException; | |
|  | import java.util.TreeMap; |
|  |  |
|  | import org.apache.hadoop.conf.Configuration; |
|  | import org.apache.hadoop.fs.Path; |
|  | import org.apache.hadoop.io.LongWritable; |
|  | import org.apache.hadoop.io.NullWritable; |
|  | 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; |
|  |  |
|  |  |
|  | //use retail data D11,D12,D01 and D02 |
|  |  |
|  |  |
|  | public class Margin { |
|  |  |
|  | // Mapper Class |
|  |  |
|  | public static class TopProductMapperClass extends Mapper<LongWritable,Text,Text,Text> |
|  | { |
|  | public void map(LongWritable key, Text value, Context context) |
|  | { |
|  | try{ |
|  | String[] str = value.toString().split(";"); |
|  | String prodid = str[5]; |
|  | String sales = str[8]; |
|  | String cost = str[7]; |
|  | String qty = str[6]; |
|  | String myValue = qty + ',' + cost + ',' + sales; |
|  | context.write(new Text(prodid), new Text(myValue)); |
|  | } |
|  | catch(Exception e) |
|  | { |
|  | System.out.println(e.getMessage()); |
|  | } |
|  | } |
|  | } |
|  |  |
|  | //Reducer class |
|  |  |
|  | public static class TopProductReducerClass extends Reducer<Text,Text,NullWritable,Text> |
|  | { |
|  | private TreeMap<Double, Text> repToRecordMap = new TreeMap<Double, Text>(); |
|  |  |
|  | public void reduce(Text key, Iterable <Text> values, Context context) throws IOException, InterruptedException |
|  | { |
|  | long totalsales = 0; |
|  | long totalcost = 0; |
|  | int totalqty = 0; |
|  | long profit = 0; |
|  | double margin = 0.00; |
|  | for (Text val : values) |
|  | { |
|  | String[] token = val.toString().split(","); |
|  | totalsales = totalsales + Long.parseLong(token[2]); |
|  | totalcost = totalcost + Long.parseLong(token[1]); |
|  | totalqty = totalqty + Integer.parseInt(token[0]); |
|  | } |
|  | profit = totalsales - totalcost; |
|  |  |
|  | if (totalcost != 0) |
|  | margin = ((totalsales - totalcost)\*100)/totalcost; |
|  | else |
|  | margin = ((totalsales - totalcost)\*100); |
|  |  |
|  | String myValue = key.toString(); |
|  | String myProfit = String.format("%d", profit); |
|  | String myMargin = String.format("%f", margin); |
|  | String myQty = String.format("%d", totalqty); |
|  | myValue = myValue + ',' + myQty + ',' + myProfit + ',' + myMargin; |
|  |  |
|  | repToRecordMap.put(new Double(margin), new Text(myValue)); |
|  | } |
|  |  |
|  | protected void cleanup(Context context) throws IOException, |
|  | InterruptedException |
|  | { |
|  |  |
|  | for (Text t : repToRecordMap.descendingMap().values()) |
|  | { |
|  | context.write(NullWritable.get(), t); |
|  | } |
|  | } |
|  |  |
|  | } |
|  |  |
|  | //Main class |
|  |  |
|  | public static void main(String[] args) throws Exception { |
|  |  |
|  | Configuration conf = new Configuration(); |
|  | Job job = Job.getInstance(conf, "Margin"); |
|  | job.setJarByClass(Margin.class); |
|  | job.setMapperClass(TopProductMapperClass.class); |
|  | job.setReducerClass(TopProductReducerClass.class); |
|  | job.setMapOutputKeyClass(Text.class); |
|  | job.setMapOutputValueClass(Text.class); |
|  | job.setOutputKeyClass(NullWritable.class); |
|  | job.setOutputValueClass(Text.class); |
|  | FileInputFormat.addInputPath(job, new Path(args[0])); |
|  | FileOutputFormat.setOutputPath(job, new Path(args[1])); |
|  | System.exit(job.waitForCompletion(true) ? 0 : 1); |
|  | } |
|  | } |