**Problem 1** . Program to print only mapper output.

//import java.io.\*;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class NYSEInput {

public static class InputMapClass extends Mapper<LongWritable,Text,LongWritable,Text>

{

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

{

try{

context.write(key,value);

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

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

Configuration conf = new Configuration();

//conf.set("name", "value")

Job job = Job.getInstance(conf, "Volume Count");

job.setJarByClass(NYSEInput.class);

job.setMapperClass(InputMapClass.class);

job.setNumReduceTasks(0);

job.setOutputKeyClass(LongWritable.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);

}

}

**Problem 2** . Program to print stock volume data.

import java.io.\*;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

// find total volume of stock

public class StockVolume

{

public static class MapClass extends Mapper<LongWritable,Text,Text,LongWritable>

{

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

{

try

{

String[] str = value.toString().split(",");

long vol = Long.parseLong(str[7]);

context.write(new Text(str[1]),new LongWritable(vol));

}

catch(Exception e)

{ System.out.println(e.getMessage()); }

}

}

public static class ReduceClass extends Reducer<Text,LongWritable,Text,LongWritable>

{

private LongWritable result = new LongWritable();

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

{

long sum = 0;

for (LongWritable val : values)

{ sum += val.get(); }

result.set(sum);

context.write(key, result);

}

}

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

{

Configuration conf = new Configuration();

//conf.set("name", "value")

Job job = Job.getInstance(conf, "Volume Count");

job.setJarByClass(StockVolume.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

//job.setNumReduceTasks(0);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

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

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

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

}

}

**Problem 3** . Program to find total duration of caller who has made long distance call > 60 min.

import java.io.\*;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

//find total duration of caller who has made long total distance call > 60 min

public class STDCalls

{

public static class MapClass extends Mapper<LongWritable,Text,Text,IntWritable>

{

Text phoneNumber = new Text();

//LongWritable phone1 = new LongWritable();

IntWritable durationInMinutes = new IntWritable();

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

{

try

{

String[] parts = value.toString().split(",");

if (parts[4].equals("1")) {

phoneNumber.set(parts[0]);

//long callerid = Long.parseLong(parts[0]);

String callEndTime = parts[3];

String callStartTime = parts[2];

long duration = toMillis(callEndTime) - toMillis(callStartTime);

durationInMinutes.set((int)(duration/(1000 \* 60)));

context.write(phoneNumber,durationInMinutes);

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

private long toMillis(String date)

{

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date dateFrm = null;

try

{ dateFrm = format.parse(date); }

catch (ParseException e)

{ e.printStackTrace(); }

return dateFrm.getTime();

}

}

public static class ReduceClass extends Reducer<Text,IntWritable,Text,IntWritable>

{

private IntWritable result = new IntWritable();

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

long sum = 0;

for (IntWritable val : values)

{ sum += val.get(); }

if (sum >= 60)

{

result.set((int) sum);

context.write(key, result);

}

}

}

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

{

Configuration conf = new Configuration();

conf.set("mapreduce.output.textoutputformat.separator",",");

Job job = Job.getInstance(conf, "STD Calls");

job.setJarByClass(STDCalls.class);

job.setMapperClass(MapClass.class);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

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

}

}

**Problem 4** . Program using partitioner to find out max sal in three age group

import java.io.\*;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

import org.apache.hadoop.util.Tool;

public class MyPartitioner extends Configured implements Tool

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAPPER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static class MapClass extends Mapper<LongWritable, Text, Text, Text>

{

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

{

try

{

String[] str = value.toString().split(",");

String gender = str[3];

context.write(new Text(gender), new Text(value));

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*REDUCER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static class ReducerClass extends Reducer<Text, Text, Text, IntWritable>

{

public int max = -1;

private Text outputKey = new Text();

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

{

max = -1;

for (Text val : values)

{

String[] str = val.toString().split(",");

if(Integer.parseInt(str[4]) > max)

{

max = Integer.parseInt(str[4]);

String mykey = str[3] + "," + str[1] + "," + str[2] ;

outputKey.set(mykey);

}

}

context.write(outputKey, new IntWritable(max));

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*PARTITIONER\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static class CaderPartitioner extends Partitioner<Text, Text>

{

@Override

public int getPartition(Text key, Text value, int numReduceTask)

{

String[] str = value.toString().split(",");

int age = Integer.parseInt(str[2]);

if(age <= 20)

return 0;

else if (age > 20 && age <= 30)

return 1;

else

return 2;

}

}

@Override

public int run(String[] arg0) throws Exception

{

// TODO Auto-generated method stub

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

{

Configuration conf = new Configuration();

conf.set("mapreduce.output.textoutputformat.separator",",");

Job job = Job.getInstance(conf, "STD Calls");

job.setJarByClass(MyPartitioner.class);

job.setMapperClass(MapClass.class);

job.setPartitionerClass(CaderPartitioner.class);

job.setReducerClass(ReducerClass.class);

job.setNumReduceTasks(3);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

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

}

}

**Problem 5** . Program for word count

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class WordCount

{

public static class mapClass extends Mapper<LongWritable, Text, Text, IntWritable>

{

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

{

String line = value.toString();

StringTokenizer st = new StringTokenizer(line);

try

{

while(st.hasMoreTokens())

{

context.write(new Text((st.nextToken()).toLowerCase()), new IntWritable(1));

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

}

}

public static class reduceClass extends Reducer<Text, IntWritable, Text, IntWritable>

{

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

{

int sum=0;

for(IntWritable val : values)

{

sum+=val.get();

}

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

}

}

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

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Word Count");

job.setJarByClass(WordCount.class);

job.setMapperClass(mapClass.class);

job.setCombinerClass(reduceClass.class);

job.setReducerClass(reduceClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

}

}

**Problem 6** . Program for reduce side join

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

import org.apache.hadoop.mapreduce.\*;

public class ReduceJoin

{

public static class CustsMapper extends Mapper<LongWritable, Text , Text, Text>

{

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

{

String record = value.toString();

String[] parts = record.split(",");

context.write(new Text(parts[0]), new Text("custs\t" + parts[1]));

}

}

public static class TnxMapper extends Mapper<LongWritable, Text, Text, Text>

{

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

{

String record = value.toString();

String[] parts = record.split(",");

context.write(new Text(parts[2]), new Text("txns\t" + parts[3]));

}

}

public static class ReduceJoinClass extends Reducer<Text, Text, Text, Text>

{

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

{

String name="";

float total=(float) 0.0;

int count = 0;

for(Text text:value)

{

String[] parts = text.toString().split("\t");

if(parts[0].equals("txns"))

{

total+=Float.parseFloat(parts[1]);

count+=1;

}

else if (parts[0].equals("custs"))

{

name=parts[1].toString();

}

}

context.write(new Text(name), new Text(String.format("%d\t%f", count, total)));

}

}

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf,"Reduce Side Join ");

job.setJarByClass(ReduceJoin.class);

job.setReducerClass(ReduceJoinClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class, CustsMapper.class);

MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, TnxMapper.class);

FileOutputFormat.setOutputPath(job, new Path(args[2]));

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

}

}

**Problem 7** . Program for map side join

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.net.URI;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.Mapper;

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

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**public** **class** Mapperjoin

{

**public** **static** **class** mymapper **extends** Mapper<LongWritable, Text, Text, Text>

{

**private** Map<String, String > abMap = **new** HashMap<String, String>();

**private** Map<String, String > abMap1 = **new** HashMap<String, String>();

**private** Text outputKey = **new** Text();

**private** Text outputValue = **new** Text();

**protected** **void** setup (Context context) **throws** IOException, InterruptedException

{

**super**.setup(context);

URI[] files = context.getCacheFiles();

Path p = **new** Path((files[0]));

Path p1 = **new** Path((files[1]));

**if**(p.getName().equals("salary.txt"))

{

BufferedReader reader = **new** BufferedReader(**new** FileReader(p.toString()));

String line = reader.readLine();

**while**(line != **null**)

{

String[] tokens = line.split(",");

String emp\_id = tokens[0];

String emp\_sal = tokens[1];

abMap.put(emp\_id, emp\_sal);

line = reader.readLine();

}

reader.close();

}

**if**(p1.getName().equals("desig.txt"))

{

BufferedReader reader = **new** BufferedReader(**new** FileReader(p.toString()));

String line = reader.readLine();

**while**(line != **null**)

{

String[] tokens = line.split(",");

String emp\_id = tokens[0];

String emp\_desig = tokens[1];

abMap1.put(emp\_id, emp\_desig);

line = reader.readLine();

}

reader.close();

}

**if**(abMap.isEmpty())

**throw** **new** IOException("Error: Unable to load Salary file");

**if**(abMap1.isEmpty())

**throw** **new** IOException("Error: Unable to load Designation file");

}

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

{

String row = value.toString();

String[] tokens = row.split(",");

String emp\_id = tokens[0];

String salary = abMap.get(emp\_id);

String desig = abMap1.get(emp\_id);

String Sal\_Desig = salary + "," + desig;

outputKey.set(row);

outputValue.set(Sal\_Desig);

}

**public** **static** **void** main(String[] args) **throws** IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = **new** Configuration();

conf.set("mapreduce.output.textoutputformat.seperatot", ",");

Job job = Job.*getInstance*(conf);

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

job.setJobName("Map side join");

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

job.addCacheFile(**new** Path("salary.txt").toUri());

job.addCacheFile(**new** Path("desig.txt").toUri());

job.setNumReduceTasks(0);

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

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

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

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

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

}

}

}

**Problem 8**. Program for key value to Text conversion

import java.io.IOException;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class KeyValueToText

{

public static class keyvaluemapper extends Mapper<Text, Text, Text, Text>

{

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

{

context.write(key, value);

}

}

public static void main(String[] args ) throws IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = new Configuration();

conf.set("mapreduce.input.keyvaluelinerecordreader.key.value.separator", "#");

Job job = Job.getInstance(conf,"Break string into two part i.e. key and value");

job.setJarByClass(KeyValueToText.class);

job.setMapperClass(keyvaluemapper.class);

job.setNumReduceTasks(0);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setInputFormatClass(KeyValueTextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

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

}

}

**Problem 9**. Program for text to sequence conversion

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class TextToSequence

{

public static class SequenceMapper extends Mapper<LongWritable, Text, LongWritable, Text>

{

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

{

String line = value.toString();

String[] tokens = line.split("\t");

context.write(new LongWritable(Long.parseLong(tokens[0])), new Text(tokens[1]));

}

}

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf,"Convert Text to Sequence");

job.setJarByClass(KeyValueToText.class);

job.setMapperClass(SequenceMapper.class);

job.setNumReduceTasks(0);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(SequenceFileOutputFormat.class);

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

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

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

}

}

**Problem 10**. Program for sequence to text conversion

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class SequenceToText

{

public static class SequenceMapper extends Mapper<LongWritable, Text, LongWritable, Text>

{

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

{

context.write(key, value);

}

}

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf,"Convert Sequence to Text");

job.setJarByClass(KeyValueToText.class);

job.setMapperClass(SequenceMapper.class);

job.setNumReduceTasks(0);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

job.setInputFormatClass(SequenceFileInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

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

}

}

**Problem 11**. Program for text to Null Conversion

import java.io.IOException;

import org.apache.hadoop.conf.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.io.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.\*;

import org.apache.hadoop.mapreduce.lib.output.\*;

public class TextToNull

{

public static class SequenceMapper extends Mapper<LongWritable, Text, LongWritable, Text>

{

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

{

String line = value.toString();

String[] tokens = line.split("\t");

context.write(new LongWritable(Long.parseLong(tokens[0])), new Text(tokens[1]));

}

}

public static void main(String[] args) throws IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = new Configuration();

Job job = Job.getInstance(conf,"Convert Text to Null");

job.setJarByClass(KeyValueToText.class);

job.setMapperClass(SequenceMapper.class);

job.setNumReduceTasks(0);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(NullOutputFormat.class);

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

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

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

}

}

**Problem 12**. Program to find out the customer I.D for the customer who has spent the maximum amount in a month.

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

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.Reducer.Context;

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

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Top5m {

public static class MyMapper extends Mapper<LongWritable,Text,Text,Text>{

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

String arr[]=value.toString().split(";");

context.write(new Text("top 1"),new Text(arr[1]+","+arr[8]));

}

}

public static class MyReducer extends Reducer<Text,Text,Text,Text>{

String ss;

TreeMap tm=new TreeMap();

public void reduce(Text key,Iterable<Text> value,Context context) throws IOException, InterruptedException{

for(Text num:value)

{

String str[]=num.toString().split(",");

int vol=Integer.parseInt(str[1]);

String id=str[0];

tm.put(vol, id);

if(tm.size()>1){

tm.remove(tm.firstKey());

}

}

context.write(key, new Text(tm.toString()));

}

}

public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

Configuration c=new Configuration();

Job job=Job.getInstance(c,"sss");

job.setJarByClass(Top5m.class);

job.setMapperClass(MyMapper.class);

job.setReducerClass(MyReducer.class);

//job.setNumReduceTasks(0);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileSystem.get(c).delete(new Path(args[1]),true);

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

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

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

}

}

**Problem 13**. Program to Find out the top 4 product being sold.

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

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.Reducer.Context;

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

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class topp {

public static class MyMapper extends Mapper<LongWritable,Text,Text,Text>{

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

String arr[]=value.toString().split(";");

context.write(new Text("top 1"),new Text(arr[5]+","+arr[8]));

}

}

public static class MyReducer extends Reducer<Text,Text,Text,Text>{

String ss;

TreeMap tm=new TreeMap();

public void reduce(Text key,Iterable<Text> value,Context context) throws IOException, InterruptedException{

for(Text num:value)

{

String str[]=num.toString().split(",");

int vol=Integer.parseInt(str[1]);

String id=str[0];

tm.put(vol, id);

if(tm.size()>4){

tm.remove(tm.firstKey());

}

}

context.write(key, new Text(tm.toString()));

}

}

public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

Configuration c=new Configuration();

Job job=Job.getInstance(c,"sss");

job.setJarByClass(topp.class);

job.setMapperClass(MyMapper.class);

job.setReducerClass(MyReducer.class);

//job.setNumReduceTasks(0);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileSystem.get(c).delete(new Path(args[1]),true);

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

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

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

}

}

**Problem 13**. Program to Find out the top 5 viable products and the top 5 product subclass for the age group A, B, C etc..... Data should be taken for all the 4 months

import java.io.IOException;

import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.Reducer.Context;

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

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class partition1 {

public static class MyMapper extends Mapper<LongWritable,Text,Text,Text>{

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

String arr[]=value.toString().split(";");

context.write(new Text(arr[2]),new Text(arr[5]+","+arr[7]+","+arr[8]));

}

}

public static class pa extends Partitioner<Text,Text>{

@Override

public int getPartition(Text key, Text arg1, int arg2) {

if(key.toString().contains("A")){

return 0;

}

if(key.toString().contains("B")){

return 1;

}

if(key.toString().contains("C")){

return 2;

}if(key.toString().contains("D")){

return 3;

}if(key.toString().contains("E")){

return 4;

}if(key.toString().contains("F")){

return 5;

}if(key.toString().contains("G")){

return 6;

}if(key.toString().contains("H")){

return 7;

}if(key.toString().contains("I")){

return 8;

}if(key.toString().contains("J")){

return 9;

}if(key.toString().contains("K")){

return 10;

}

else{

return 11;

}}

}

public static class MyReducer extends Reducer<Text,Text,Text,Text>{

String ss;int abc=0;int max=0;int vol1=0;

TreeMap tm=new TreeMap();

public void reduce(Text key,Iterable<Text> value,Context context) throws IOException, InterruptedException{

for(Text num:value)

{

String str[]=num.toString().split(",");

int vol=Integer.parseInt(str[1]);

int vol1=Integer.parseInt(str[2]);

String id=str[0];

// abc=vol1-vol;

// max=Math.max(abc, max);

tm.put((vol1-vol), id);

if(tm.size()>5){

tm.remove(tm.firstKey());

}

}

context.write(key, new Text(tm.toString()));

}

}

public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

Configuration c=new Configuration();

Job job=Job.getInstance(c,"sss");

job.setJarByClass(partition1.class);

job.setMapperClass(MyMapper.class);

job.setPartitionerClass(pa.class);

job.setReducerClass(MyReducer.class);

job.setNumReduceTasks(12);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

FileSystem.get(c).delete(new Path(args[1]),true);

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

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

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

}

}