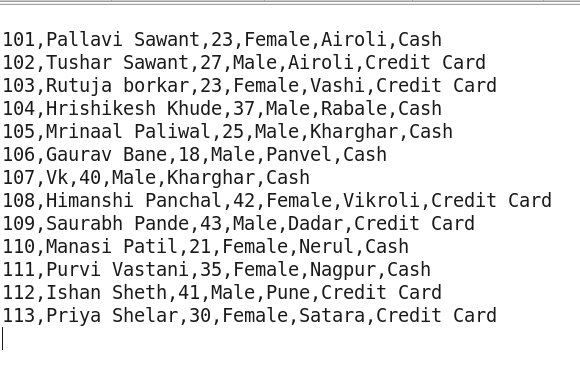
**DATASET:**

****

**Q) Find payment type wise gender count of the customer)**

**CODE:**

DEFINE paygender() RETURNS C{

log=load '/user/training/Pallavi/cust' using PigStorage(',') AS (id:int,name:chararray,age:int,gender:chararray,city:chararray,pay:chararray);

gr=group log by (pay,gender);

$C=foreach gr generate group,COUNT(log.$3);

};

**TERMINAL:**

pig -f /home/training/Desktop/paygender.pig

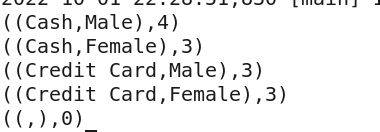
pig

import '/home/training/Desktop/paygender.pig';

Aa =paygender();

Dump aa;

**OUTPUT:**



**Q)Find highest age and lowest age of customers according to city**

**CODE:**

DEFINE highlowage() RETURNS C{

log=load '/user/training/Pallavi/cust' using PigStorage(',') AS (id:int,name:chararray,age:int,gender:chararray,city:chararray,pay:chararray);

gr=group log by city;

$C=foreach gr generate group,(MAX(log.$2),MIN(log.$2));

};

**TERMINAL:**

pig -f /home/training/Desktop/highlowage.pig

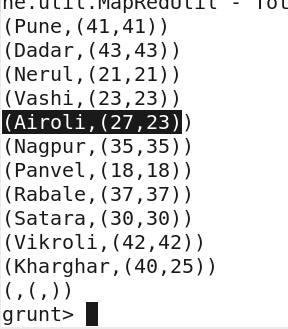
pig

import '/home/training/Desktop/highlowage.pig';

Aa =highlowage();

Dump aa;

**OUTPUT:**



**Q) Find the average age,minimum age,maximum age**

log = LOAD '/user/training/Pallavi/custs' using PigStorage(',') AS ( custid:int, firstname:chararray, lastname:chararray, age:int, profession:chararray);

B= group log all;

maxage1 = FOREACH B GENERATE MAX (log.age),MIN(log.age),AVG(log.age);

**Q) Specific Word Count:**

**Code:**

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

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

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

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

//import org.apache.hadoop.util.\*;

public class specificwordcount {

public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

String find = "Hello";

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {

String line = value.toString();

String temp = "";

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

temp = tokenizer.nextToken();

if(temp.equals(find)){

word.set(temp);

output.collect(word, one);

}

}

}

}

public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {

int sum = 0;

while (values.hasNext()) {

sum += values.next().get(); }

output.collect(key, new IntWritable(sum));

} }

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

JobConf conf = new JobConf(specificwordcount.class);

conf.setJobName("wordcount");

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

conf.setMapperClass(Map.class);

conf.setCombinerClass(Reduce.class);

conf.setReducerClass(Reduce.class);

conf.setInputFormat(TextInputFormat.class);

conf.setOutputFormat(TextOutputFormat.class);

FileInputFormat.setInputPaths(conf, new Path(args[0]));

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

JobClient.runJob(conf);

} }

**Q) Average Word Count in file:**

**Code:**

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

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

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

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

public class wordcount {

public static class WordMapper extends

Mapper<LongWritable,Text,Text,IntWritable>{

@Override

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

throws IOException, InterruptedException{

String line = value.toString();

for (String word : line.split("\\W+")){

if (word.length()>0){

context.write(new Text(word), new IntWritable(1));

}

}

}

}

public static class SumReducer extends Reducer<Text, IntWritable,Text, IntWritable>{

@Override

public void reduce(Text key, Iterable<IntWritable> values,

Context context) throws IOException, InterruptedException{

int wordCount = 0;

for (IntWritable value: values){

wordCount += value.get();

}

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

}

}

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

if(args.length != 2){

System.out.printf("Usage: WordCount <input dir> <output dir>\n");

System.exit(-1);

}

Job job = new Job();

job.setJarByClass(wordcount.class);

job.setJobName("WordCount");

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

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

job.setMapperClass(WordMapper.class);

job.setReducerClass(SumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

boolean success = job.waitForCompletion(true);

System.exit(success ? 0 : 1);

} }

**Q) Find the firstname with highest occurrence**

**Code:**

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.io.Text;

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

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

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

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

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.commons.cli.Options;

import org.apache.hadoop.util.GenericOptionsParser;

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

public class fname {

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

{

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

{

String line=value.toString().split(",")[1];

Text outputKey=new Text(line);

IntWritable outputValue=new IntWritable(1);

con.write(outputKey, outputValue);

}

}

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

{

public void reduce(Text line,Iterable<IntWritable>max\_fname,Context con)throws IOException,InterruptedException

{

int max = 0;

for(IntWritable value:max\_fname)

{

max+= Math.max(max, value.get());

}

con.write(line, new IntWritable(max));

}

}

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

Configuration c = new Configuration();

GenericOptionsParser parser = new GenericOptionsParser(c,args);

String[] files = parser.getRemainingArgs();

Path p1 = new Path(files[0]);

Path p4 = new Path(files[1]);

Job j = new Job(c,"multiple");

j.setJarByClass(fname.class);

j.setMapperClass(Map.class);

j.setReducerClass(Reduce.class);

j.setOutputKeyClass(Text.class);

j.setOutputValueClass(IntWritable.class);

j.setMapOutputKeyClass(Text.class);

j.setMapOutputValueClass(IntWritable.class);

MultipleInputs.addInputPath(j,p1,TextInputFormat.class,Map.class);

FileOutputFormat.setOutputPath(j,p4);

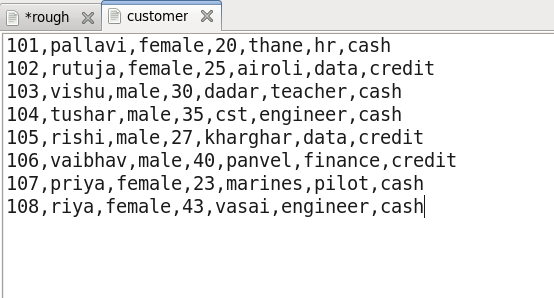
System.exit(j.waitForCompletion(true)?0:1);

}

}

**Q) Write a Map Reduce program to find age group wise(20-30,31-45) of customer count**

**DATASET:**

****

**CODE:**

import java.io.\*;

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.\*;

public class agegrp\_wise\_count extends Configured implements Tool

{

//map class

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

{

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

throws IOException, InterruptedException {

String line = value.toString();

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

String code=word[3];

context.write(new Text (code), new IntWritable (1));

}

}

//Reducer class

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

{public int count = 0;

@Override public void reduce (Text code, Iterable<IntWritable> values, Context context)

throws IOException, InterruptedException

{

int wordCount = 0;

for (IntWritable value : values)

{

wordCount += value.get();

wordCount+=1;

}

count+=wordCount;

context.write(code, new IntWritable (count));

}

}

//Partitioner class

public static class CaderPartitioner extends Partitioner<Text,IntWritable>{

public int getPartition ( Text key , IntWritable value , int numReduceTasks )

{

String num = key.toString();

int val = Integer.parseInt(num);

if ( numReduceTasks == 0 )

{

return 0 ;

}

if (val>=20 & val<=30)

{

return 1 % numReduceTasks ;

}

else if ( val>=31 & val<=45 )

{

return 2% numReduceTasks;

}

else{

return 0;

}

}

}

public int run(String[] arg) throws Exception

{

// TODO Auto-generated method stub

Configuration conf = getConf();

Job job = new Job (conf ,"Partitioner" ) ;

job.setJarByClass(agegrp\_wise\_count.class ) ;

FileInputFormat.setInputPaths (job ,new Path ( arg [ 0 ]));

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

job.setMapperClass(Map.class ) ;

job.setMapOutputKeyClass ( Text.class ) ;

job.setMapOutputValueClass (IntWritable.class ) ;

// set partitioner statement

job.setPartitionerClass (CaderPartitioner.class ) ;

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(5) ;

job.setInputFormatClass (TextInputFormat.class ) ;

job.setOutputFormatClass(TextOutputFormat.class ) ;

job.setOutputKeyClass ( Text.class ) ;

job.setOutputValueClass ( Text.class ) ;

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

return 0 ;

}

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

{

// TODO Auto-generated method stub

int res=ToolRunner.run(new Configuration(),new agegrp\_wise\_count (), args);

System.exit(0);

}

}

**Q) Map Reduce Program to find sum from two files**

import java.io.IOException;

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.io.Text;

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

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

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

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

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

//

import org.apache.commons.cli.Options;

import org.apache.hadoop.util.GenericOptionsParser;

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

public class sum\_max\_min {

//public static class Map extends Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT>

public static class Map1 extends Mapper<LongWritable, Text, Text, IntWritable>{

//public void map1(KEYIN key, VALUEIN value,Context context)

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

// TODO Auto-generated method stub

String line=value.toString();

String[] line1=line.split(",");

String name=line1[0];

Text outputKey=new Text(name);

int num=Integer.parseInt(line1[2]);

IntWritable outputValue=new IntWritable(num);

con.write(outputKey, outputValue);

}

}

//public static class Map extends Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT>

public static class Map2 extends Mapper<LongWritable, Text, Text, IntWritable>{

//public void map1(KEYIN key, VALUEIN value,Context context)

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

// TODO Auto-generated method stub

/\*String line=value.toString();

String[] line1=line.split(",");

String name=line1[0];

int num=Integer.parseInt(line1[2]);

//super.map(key, value, con);

con.write(new Text(name), new IntWritable(num));

//super.map(key, value, con);\*/

String line=value.toString();

String[] line1=line.split(",");

String name=line1[0];

Text outputKey=new Text(name);

int num=Integer.parseInt(line1[2]);

IntWritable outputValue=new IntWritable(num);

con.write(outputKey, outputValue);

}

}

//public static class Reduce extends Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT>

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

{

@Override

//public void reduce(KEYIN arg0, Iterable<VALUEIN> arg1,Context arg2)throws IOException, InterruptedException {

public void reduce(Text name, Iterable<IntWritable>sum\_num,Context con)throws IOException, InterruptedException {

// TODO Auto-generated method stub

int sum=0;

for(IntWritable value:sum\_num){

sum+=value.get();

}

con.write(name, new IntWritable(sum));

}

}

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

Configuration c=new Configuration();

GenericOptionsParser parser = new GenericOptionsParser(c,args);

String[] files = parser.getRemainingArgs();

Path p1 = new Path(files[0]);

Path p2 = new Path(files[1]);

Path p3 = new Path(files[2]);

Job j = new Job(c,"multiple");

j.setJarByClass(sum\_max\_min.class);

j.setMapperClass(Map1.class);

j.setMapperClass(Map2.class);

j.setReducerClass(Reduce.class);

j.setOutputKeyClass(Text.class);

j.setOutputValueClass(IntWritable.class);

j.setMapOutputKeyClass(Text.class);

j.setMapOutputValueClass(IntWritable.class);

MultipleInputs.addInputPath(j,p1,TextInputFormat.class,Map1.class);

MultipleInputs.addInputPath(j, p2, TextInputFormat.class,Map2.class);

FileOutputFormat.setOutputPath(j,p3);

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

**Q) Map Reduce Program to find Min from three files**

**import** java.io.IOException;

**import** java.util.\*;

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

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

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

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

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

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

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

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

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

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

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

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

//

**import** org.apache.commons.cli.Options;

**import** org.apache.hadoop.util.GenericOptionsParser;

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

**public** **class** assignment4 {

**public** **static** **class** Map1 **extends** Mapper<LongWritable,Text,Text,IntWritable>

{

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

{

String line=value.toString();

String[] line1=line.split(",");

String gender=line1[0];

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

**int** salary=Integer.*parseInt*(line1[2]);

IntWritable outputValue=**new** IntWritable(salary);

con.write(outputKey, outputValue);

}

}

**public** **static** **class** Map2 **extends** Mapper<LongWritable,Text,Text,IntWritable>

{

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

{

String line=value.toString();

String[] line1=line.split(",");

String gender=line1[0];

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

**int** salary=Integer.*parseInt*(line1[2]);

IntWritable outputValue=**new** IntWritable(salary);

con.write(outputKey, outputValue);

}

}

**public** **static** **class** Map3 **extends** Mapper<LongWritable,Text,Text,IntWritable>

{

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

{

String line=value.toString();

String[] line1=line.split(",");

String gender=line1[0];

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

**int** salary=Integer.*parseInt*(line1[2]);

IntWritable outputValue=**new** IntWritable(salary);

con.write(outputKey, outputValue);

}

}

**public** **static** **class** Red **extends** Reducer<Text,IntWritable,Text,IntWritable>

{

**public** **void** reduce(Text gender,Iterable<IntWritable>min\_marks,Context con)**throws** IOException,InterruptedException

{

**int** min=100;

**for**(IntWritable value:min\_marks)

{

min = Math.*min*(min, value.get());

}

con.write(gender, **new** IntWritable(min));

}

}

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

Configuration c = **new** Configuration();

GenericOptionsParser parser = **new** GenericOptionsParser(c,args);

String[] files = parser.getRemainingArgs();

Path p1 = **new** Path(files[0]);

Path p2 = **new** Path(files[1]);

Path p3 = **new** Path(files[2]);

Path p4 = **new** Path(files[3]);

Job j = **new** Job(c,"multiple");

j.setJarByClass(assignment4.**class**);

j.setMapperClass(Map1.**class**);

j.setMapperClass(Map2.**class**);

j.setMapperClass(Map3.**class**);

j.setReducerClass(Red.**class**);

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

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

MultipleInputs.*addInputPath*(j,p1,TextInputFormat.**class**,Map1.**class**);

MultipleInputs.*addInputPath*(j, p2, TextInputFormat.**class**,Map2.**class**);

MultipleInputs.*addInputPath*(j, p3, TextInputFormat.**class**,Map2.**class**);

FileOutputFormat.*setOutputPath*(j,p4);

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

}

}

hdfs dfs -mkdir /user/training/Pallavi

hdfs dfs -copyFromLocal /home/training/Desktop/input /user/training/Pallavi

hdfs dfs -copyFromLocal /home/training/Desktop/input /user/training/Pallavi/input

hadoop jar /home/training/wordcount.jar /user/training/Pallavi/input /user/training/Pallavi/output

hadoop jar /home/training/assignment4.jar /user/training/Pallavi/Input11 /user/training/Pallavi/Input22 /user/training/Pallavi/Input33 /user/training/Pallavi/MultipleOutputFile-1

**DELETE FILES FROM HDFS**

hdfs dfs -rmr /user/training/filename

pig -f /home/training/Desktop/paygender.pig

pig

import '/home/training/Desktop/paygender.pig';

Aa =paygender();

Dump aa;

101,pallavi,female,20,thane,hr,cash

102,rutuja,female,25,airoli,data,credit

103,vishu,male,30,dadar,teacher,cash

104,tushar,male,35,cst,engineer,cash

105,rishi,male,27,kharghar,data,credit

106,vaibhav,male,40,panvel,finance,credit

107,priya,female,23,marines,pilot,cash

108,riya,female,43,vasai,engineer,cash

109,pallavi,female,20,thane,hr,cash

109,tushar,male,45,seawoods,hr,cash