## Count of each word

**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.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 wrodcount\_class

{

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(wrodcount\_class.class);

job.setJobName("Word Count");

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);

}

}

**Count of Specific Word**

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 cust\_class {

    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(",")){

                //if (word.length()>0){

                if (word.equals("Firefighter")){

                    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(cust\_class.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);

    }

}

**Each word length**

**import** java.io.IOException;

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

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

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

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

**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.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** Average {

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

{

@Override

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

**throws** IOException, InterruptedException{

        Text firstLetter = **new** Text();

        IntWritable wordLength = **new** IntWritable();

**char**[] a1 = **null**;

String line = value.toString().toLowerCase();

**for**(String word:line.split(" ")){

**if** (word.length() > 0) {

        firstLetter.set(String.*valueOf*(word.charAt(0)));

        wordLength.set(word.length());

        context.write(firstLetter, wordLength);

    }

}

}

}

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

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

**throws** IOException,InterruptedException{

**int** sum =0;

**int** count =0;

**int** Average =0;

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

          sum += val.get();

           count = count+1;

                       }

      Average = sum/count;

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

}

}

**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(Average.**class**);

job.setJobName("Word Count");

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);

}

}

**MULTIPLE INPUT**

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Reducer;

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

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

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

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

import org.apache.hadoop.util.GenericOptionsParser;

import org.apache.commons.cli.Options;

//import replace.MapforWordCount;

public class MultiInput {

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[3];

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[2];

Text outputKey=new Text(gender);

int salary=Integer.parseInt(line1[3]);

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> total\_sal,Context con)

throws IOException,InterruptedException

{

int sum=0;

for(IntWritable value:total\_sal)

{

sum+=value.get();

}

con.write(gender, 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(MultiInput.class);

j.setMapperClass(Map1.class);

j.setMapperClass(Map2.class);

j.setReducerClass(Red.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);

}

}

**Encrypt Email**

import java.io.\*;

import java.util.\*;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Job;

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

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

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

import org.apache.hadoop.util.GenericOptionsParser;

public class email {

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

{

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

throws IOException,InterruptedException

{

String strIncremented=new String();

            String x = "";

            String line = value.toString();

            for(int i=0;i<line.length();i++)

            {

                strIncremented+=(char)(line.charAt(i)+69);

            }

            con.write(new Text(x),new Text(strIncremented));

    }

}//end map

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

Job job=new Job();

        job.setJarByClass(email.class);

        job.setJobName("email");

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

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

    job.setMapperClass(emailencrypt.class);

    job.setOutputKeyClass(Text.class);

    job.setOutputValueClass(Text.class);

    boolean success=job.waitForCompletion(true);

    System.exit(success?0:1);

}

}

**Find Out the hit count per month**

**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 multiFilesPart 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[1];

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

//Reducer class

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));}}

//Partitioner class

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

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

   {

   if ( numReduceTasks == 0 )

   {

   return 0 ;

   }

   if ( key.toString().equals("Jan"))

   {

    return 1 ;

   }

   else if (key.toString().equals("Feb"))

   {

    return 2;

   }

   else if ( key.toString().equals("Mar") )

   {

    return 3;

   }

   else if ( key.toString().equals("Apr") )

   {

    return 4;

   }

   else if ( key.toString().equals("May") )

   {

    return 5;

   }

   else if ( key.toString().equals("Jun") )

   {

    return 6;

   }

   else if ( key.toString().equals("Jul") )

   {

    return 7;

   }

   else if ( key.toString().equals("Aug") )

   {

    return 8;

   }

   else if ( key.toString().equals("Sep") )

   {

    return 9;

   }

   else if ( key.toString().equals("Oct") )

   {

    return 10;

   }

   else if ( key.toString().equals("Nov") )

   {

    return 11;

   }

   else if ( key.toString().equals("Dec") )

   {

    return 12;

   }

   else{

    return 0;}}}

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

int res=ToolRunner.run(new Configuration(),new multiFilesPart(), args);

System.exit(0);}@Override

public int run(String[] arg) throws Exception {

Configuration conf = getConf() ;

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

   job.setJarByClass(multiFilesPart.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(SumReducer.class);

**job.setNumReduceTasks(13) ;**

   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 ;

}}

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

**CODE:**

DEFINE paygender() RETURNS C{

log=load '/user/training/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

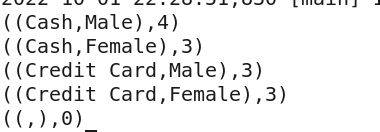
pig

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/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

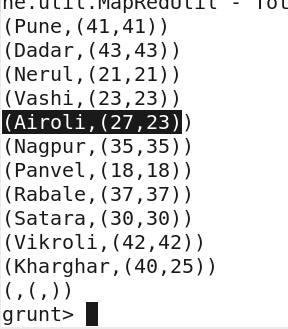
pig

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

Aa =highlowage();

Dump aa;

**OUTPUT:**



# **Web log and cust data analysis using Pig**

### **Problem Statement - Find Out the hit count per month**

### **Find Out the hit count per status messages (2XX, 3XX, 4XX, 5XX)**

**Input File (access\_log)**

**Code 1**

**For month count**

DEFINE hitcount() RETURNS C {

L = load '/user/training/Desktop/access\_log' using PigStorage('/') AS (addr:chararray, month:chararray, year:chararray, prot:chararray, status:chararray);

G = GROUP L BY month;

$C = FOREACH G generate group, COUNT(L);

};

****

**Code 2**

**For status message count**

DEFINE hitcount() RETURNS C {

    L = load '/user/training/Desktop/access\_log' using PigStorage(' ') AS (addr:chararray, month:chararray, year:chararray, prot:chararray, status:chararray, addr1:chararray, month1:chararray, year1:chararray, prot1:chararray, status1:chararray);

    G = GROUP L BY prot1;

    $C = FOREACH G generate group, COUNT(L);

};

**Text

Description automatically generated**

**Logic for 3rd i.e status message count with group all and join**

DEFINE hitcount() RETURNS C {

L = load '/user/training/Desktop/access\_log' using PigStorage(' ') AS (addr:chararray, month:chararray, year:chararray, prot:chararray, status:chararray, addr1:chararray, month1:chararray, year1:chararray, prot1:chararray, status1:chararray);

G = GROUP L BY prot1;

C = FOREACH G generate group, COUNT(L);

B = filter C by group== '200' or group =='206';

F = group B all;

A = foreach F generate all , SUM(B.$1);

x = filter C by group== '301' or group =='302' or group =='304';

y = group x redirection;

z = foreach y generate redirection, SUM(x.$1);

dump z;

p = filter C by group== '400' or group =='401' or group =='403' or group=='404' or group =='405' or group='416';

q = group p clienterror;

r = foreach q generate clienterror , SUM(p.$1);

dump r;

ab = join A by $0 FULL, z by $0;

final = join ab by $0 FULL, r by $0;

 };

Calculate the number of females (age between 18-30 years) who were victims in different crimes in different states.

CODE:

cage = LOAD '/user/training/Somaiya/crimeage' using PigStorage(',') AS (state:chararray,crime:chararray,year:long,mblw18:long,fblw18:long,mbtw18\_30:long,fbtw18\_30:long, mbtw30\_45:long,fbtw30\_45:long,mbtw45\_60:long,fbtw45\_60:long,mabv60:long,fabv60:long,tmale:lon g,tfemale:long);

X = group cage by fbtw18\_30;

Y = FOREACh X generate group, COUNT(cage);

RSLT = LIMIT Y 1;

dump RSLT;