# Text, letter, schematic Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr no** | Topic Name | Date | Pg.NO. |
| 1. | Write a program to find occurrence of any specific word | 08-08-2022 | 4 |
| 2. | Average word count | 12-08-2022 | 11 |
| 3. | Encrypt the email id from file | 23-08-2022 | 14 |
| 4. | Multiple File Input | 29-08-2022 | 17 |
| 5. | Web Log Analysis - Partitioner Assignment | 01-09-2022 | 24 |
| 6. | Crime Data analysis using Pig | 10-09-2022 | 29 |
| 7. | Web log + Cust Analysis using Pig | 07-09-2022 | 34 |

**INDEX**

# Assignment 1 Big Data

Date :08-08-2022

**Problem Statement:**

##### Write a program to ﬁnd occurrence of any speciﬁc word

This is the File created by me

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;//type casted to Int import org.apache.hadoop.io.Text;

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.mapreduce.Job;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.mapreduce.Mapper;//mapper class import org.apache.hadoop.mapreduce.Reducer;

//import CountWord.SumReducer;

//import CountWord.WordMapper;

public class CountWord {

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.equals("hello")){

**word.toLowerCase();**

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: CountWord <input dir> <output dir>\n"); System.exit(-1);

}

Job job = new Job(); **job.setJarByClass(CountWord.class); job.setJobName("CountWord");**

FileInputFormat.setInputPaths(job, new Path(args[0])); FileOutputFormat.setOutputPath(job, new Path(args[1]));

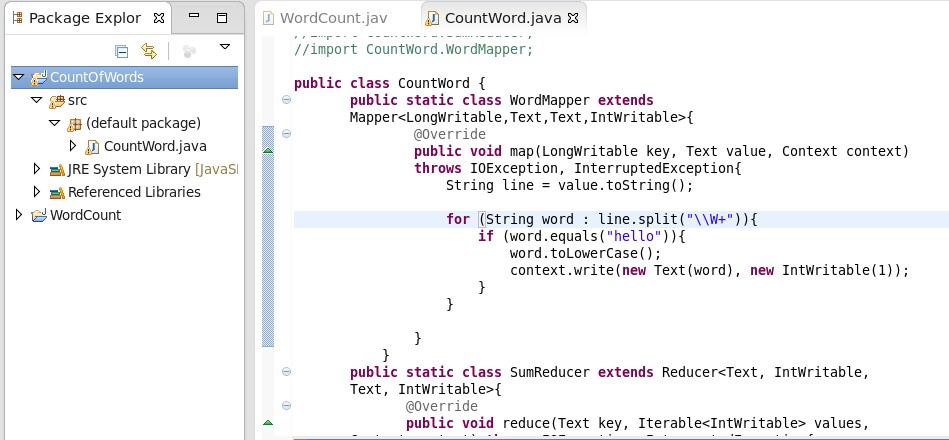
job.setMapperClass(WordMapper.class); job.setReducerClass(SumReducer.class);

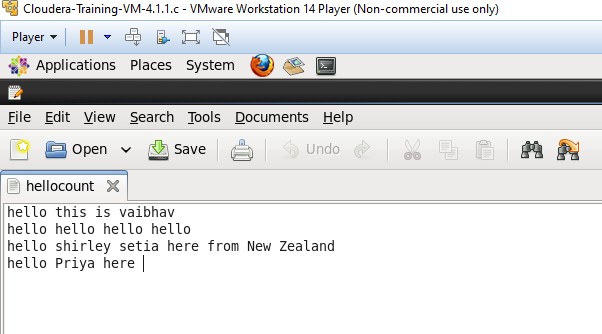
job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);

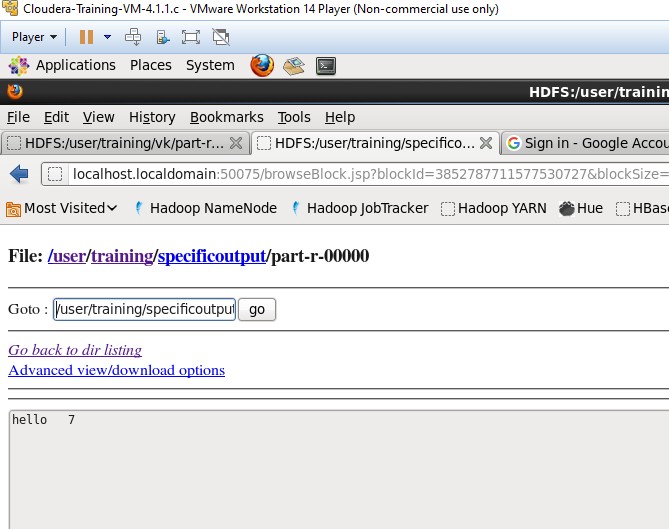
boolean success = job.waitForCompletion(true); System.exit(success ? 0 : 1);

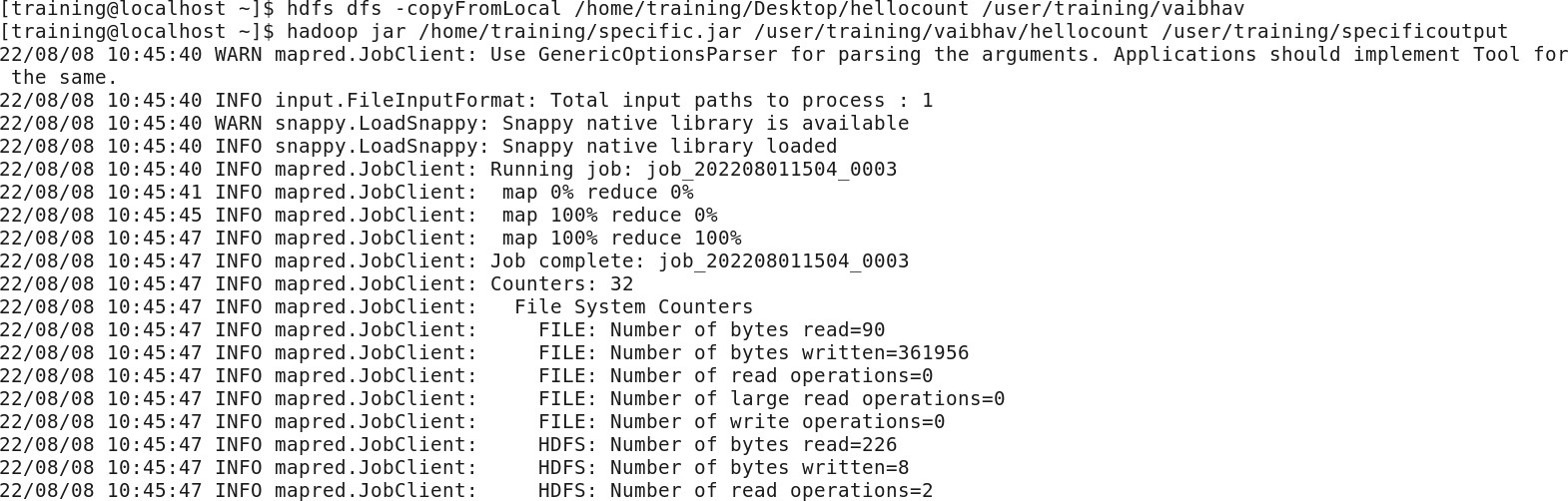
}

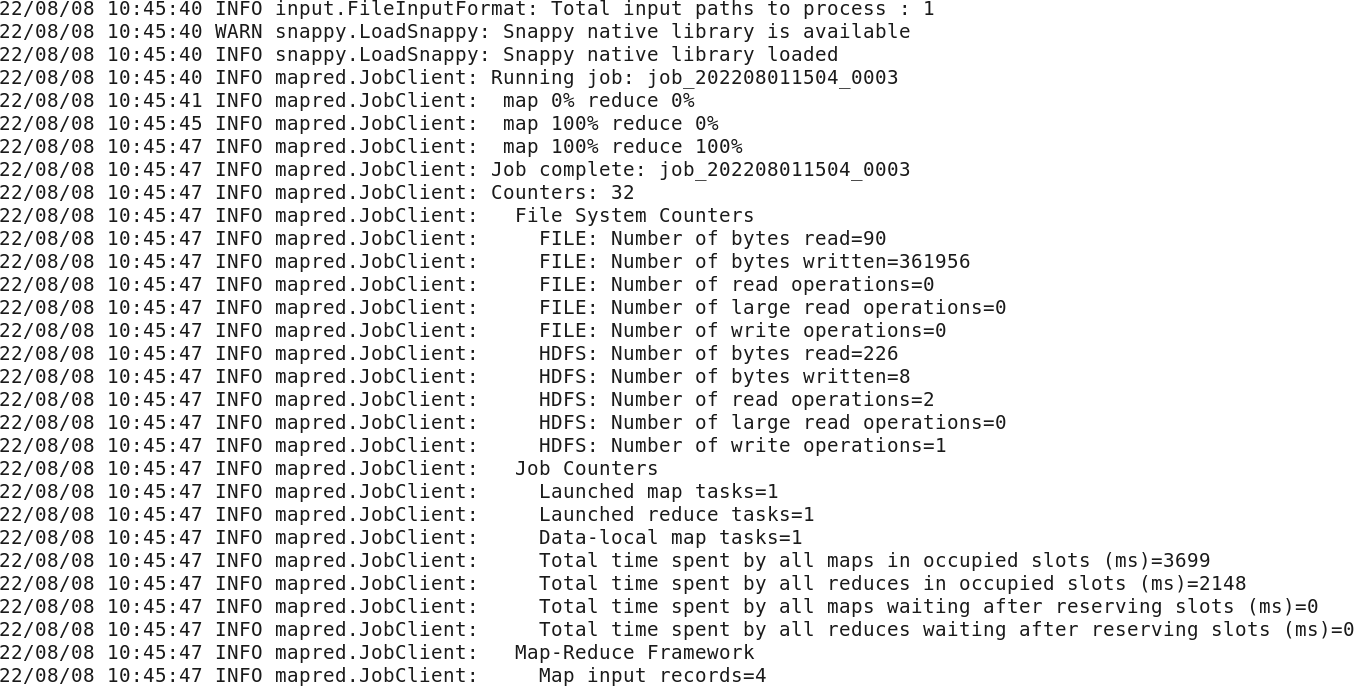
}











***Customer Data***

public class CustomerData {

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.equals("Pilot")){ word.toLowerCase();**

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: CountWord <input dir> <output dir>\n"); System.exit(-1);

}

Job job = new Job(); **job.setJarByClass(CustomerData.class); job.setJobName("CustomerData");**

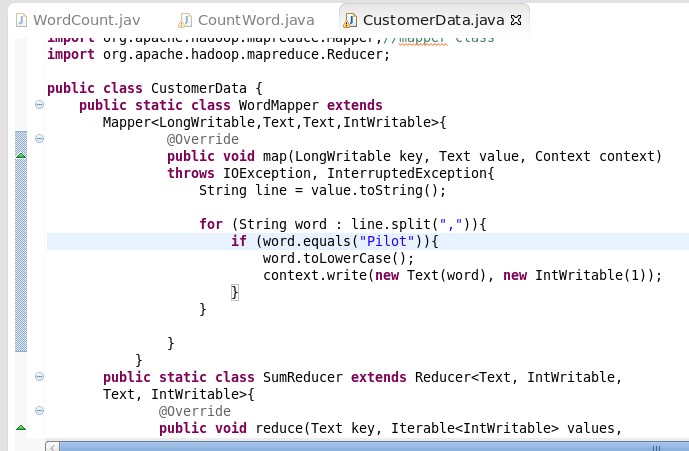
FileInputFormat.setInputPaths(job, new Path(args[0])); FileOutputFormat.setOutputPath(job, new Path(args[1]));

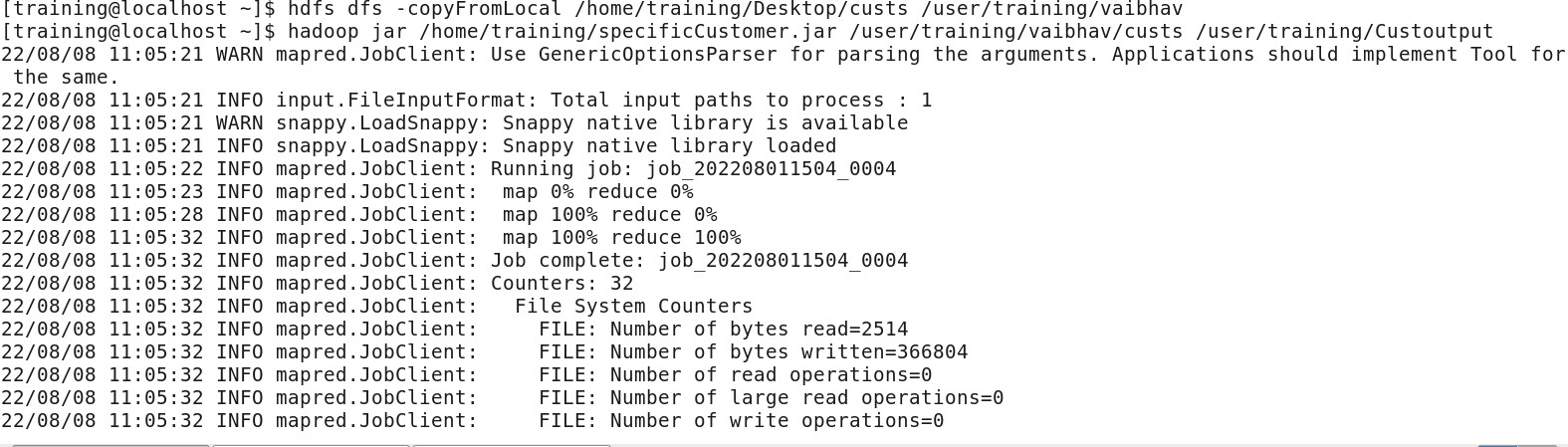
job.setMapperClass(WordMapper.class); job.setReducerClass(SumReducer.class);

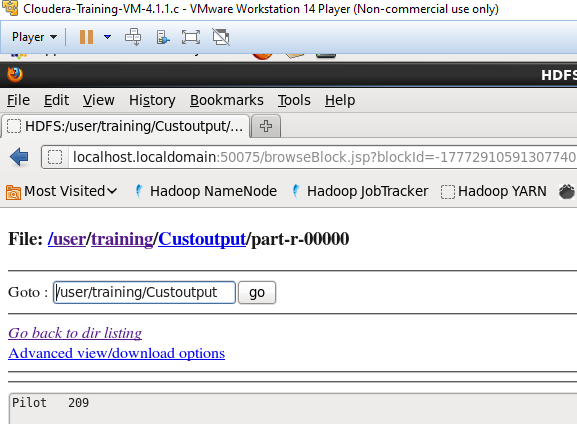
job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);

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

}

}





**Date:12-08-2022**

# Assignment Big Data

Q) Write a program to find average 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**);

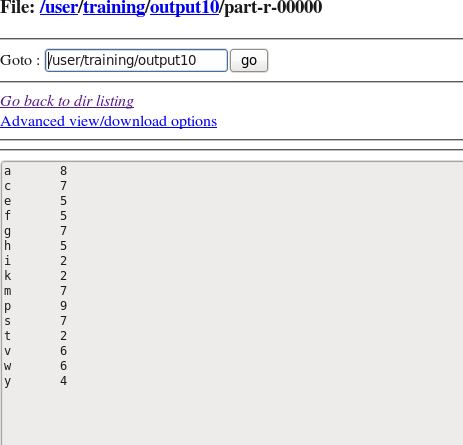
job.setOutputKeyClass(Text.**class**); job.setOutputValueClass(IntWritable.**class**);

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

}

}





**Date : 23-08-2022**

**Assignment**

**Problem Statement :** Encrypt the email id from ﬁle

**Explanation :**

##### Implementing caesar cipher algorithm for this ,

Basically Caesar cipher or Shift Cipher is a Substitution cipher algorithm in which each letter of the plain text (message) is substituted with another letter. In this algorithm, each letter of the Plaintext is shifted a number of positions based on the Key provided.

Here i have provided the **key as 69**.

## CODE:

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));

}

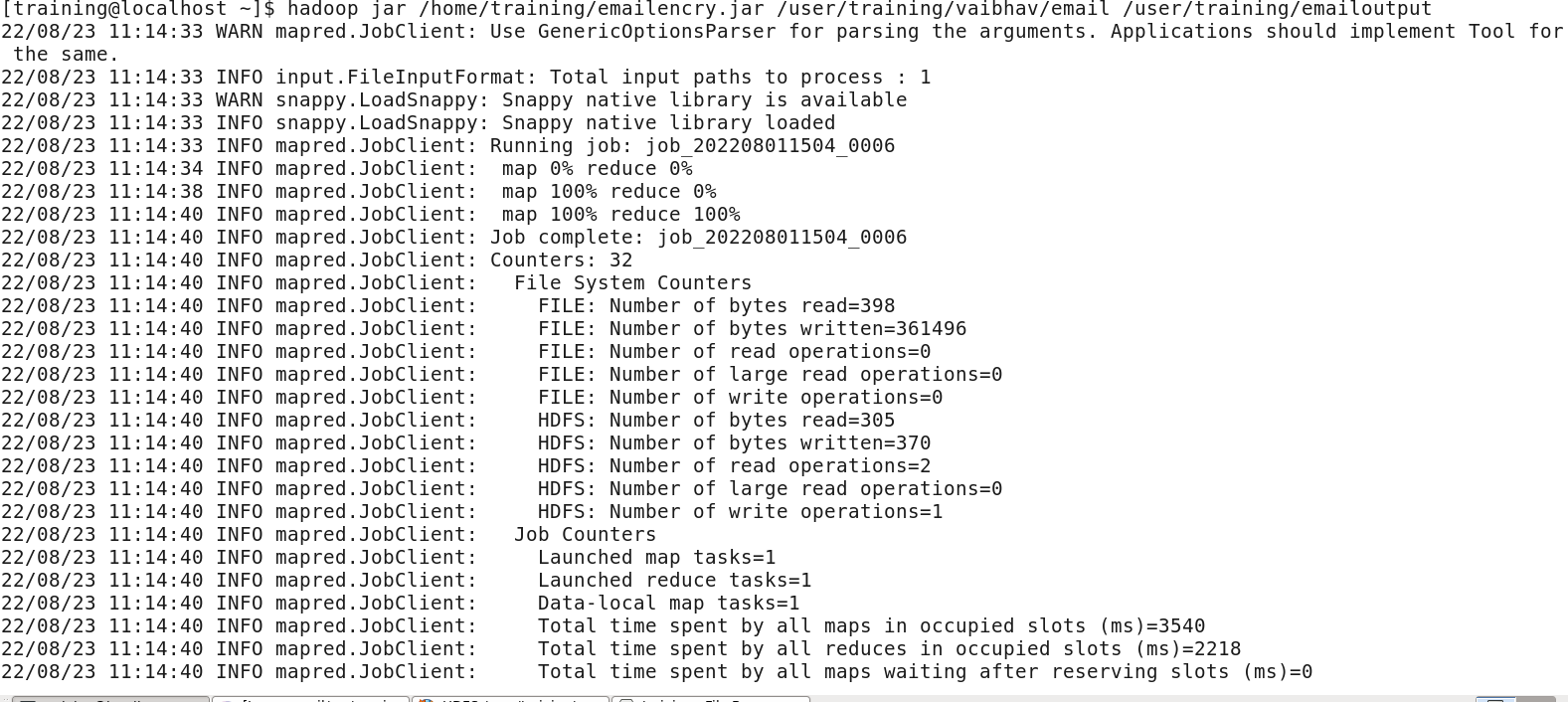
}//end map

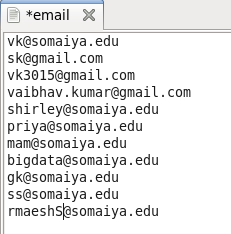
public static void main(String[] args) throws Exception{

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); }}



**Input File :**

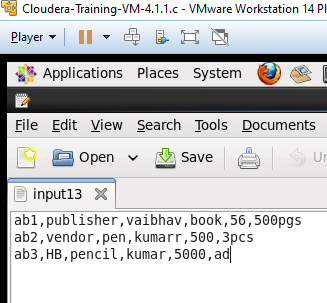
## OUTPUT:

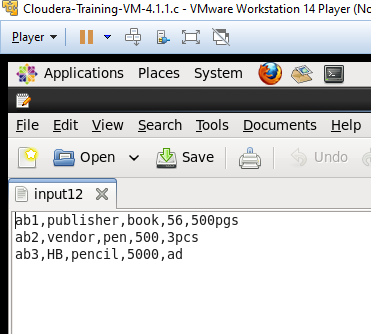
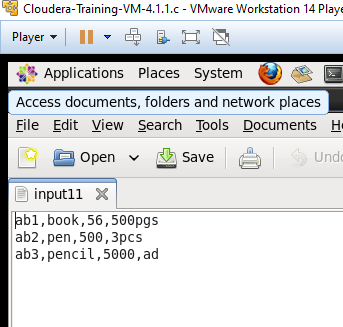


### ASSIGNMENT

**Multiple File Input**

**Problem Statement:** Create 2 or 3 input ﬁles on your own , in which the data is present in different format. Write a program to process the these ﬁles using different map class and perform any one aggerate function like sum, max, min etc. on it.





CODE:

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.lib.input.MultipleInputs; 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.hadoop.util.GenericOptionsParser;

public class MultipleFileInp {

public static class map1 extends Mapper <LongWritable, Text, Text,IntWritable>

{

@Override

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

String line = value.toString(); String[] line1=line.split(","); String category=line1[0];

Text outputKey =new Text(category); 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>

{

@Override

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

String line = value.toString(); String[] line1=line.split(","); String category=line1[0];

Text outputKey =new Text(category); int salary=Integer.parseInt(line1[3]);

IntWritable outputValue=new IntWritable(salary); con.write(outputKey, outputValue);

}

}

public static class map3 extends Mapper <LongWritable, Text, Text,IntWritable>

{

@Override

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

String line = value.toString(); String[] line1=line.split(","); String category=line1[0];

Text outputKey =new Text(category); int salary=Integer.parseInt(line1[4]);

IntWritable outputValue=new IntWritable(salary); con.write(outputKey, outputValue);

}

}

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

public void reduce(Text category,Iterable<IntWritable> total\_sal,Context con)throws IOException,InterruptedException{

int sum =0;

for(IntWritable value:total\_sal){ sum+=value.get();

}

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

}

}

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

Conﬁguration c = new Conﬁguration();

String[]ﬁles = new GenericOptionsParser(c,args).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");

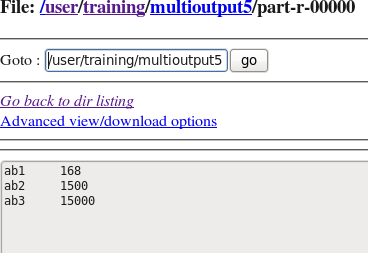
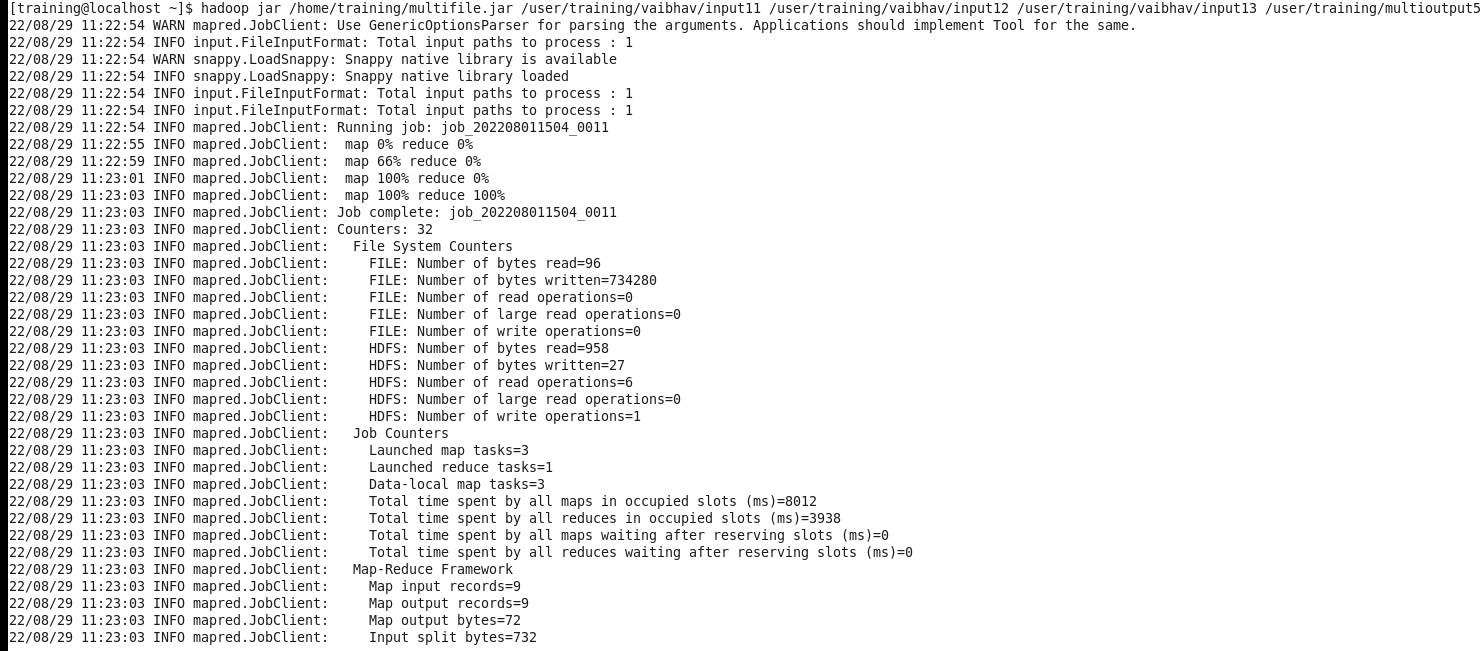
j.setJarByClass(MultipleFileInp.class); j.setMapperClass(map1.class); j.setMapperClass(map2.class); j.setMapperClass(map3.class); j.setReducerClass(red.class); j.setMapOutputKeyClass(Text.class); j.setMapOutputValueClass(IntWritable.class);

MultipleInputs.addInputPath(j, p1, TextInputFormat.class, map1.class); MultipleInputs.addInputPath(j, p2, TextInputFormat.class, map2.class);

MultipleInputs.addInputPath(j, p3, TextInputFormat.class, map3.class); FileOutputFormat.setOutputPath(j, p4); System.exit(j.waitForCompletion(true)?0:1);

}

}



**MAX**

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.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.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;

public class MultipleFileInp {

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 name = line1[0];

Text outputKey = new Text(name); int cost = Integer.parseInt(line1[2]);

IntWritable outputValue = new IntWritable(cost); 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 name = line1[0];

Text outputKey = new Text(name); int cost = Integer.parseInt(line1[3]);

IntWritable outputValue = new IntWritable(cost); 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 name = line1[0];

Text outputKey = new Text(name); int cost = Integer.parseInt(line1[4]);

IntWritable outputValue = new IntWritable(cost); con.write(outputKey, outputValue);

}

}

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

{

public void reduce(Text name, Iterable<IntWritable>total\_cost, Context con) throws IOException, InterruptedException

{

**int sum = 0; int max = 0;**

**for (IntWritable value : total\_cost) { if (value.get() > max) {**

**max = value.get();**

**}**

**}**

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

}

}

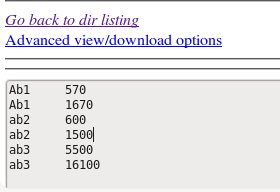
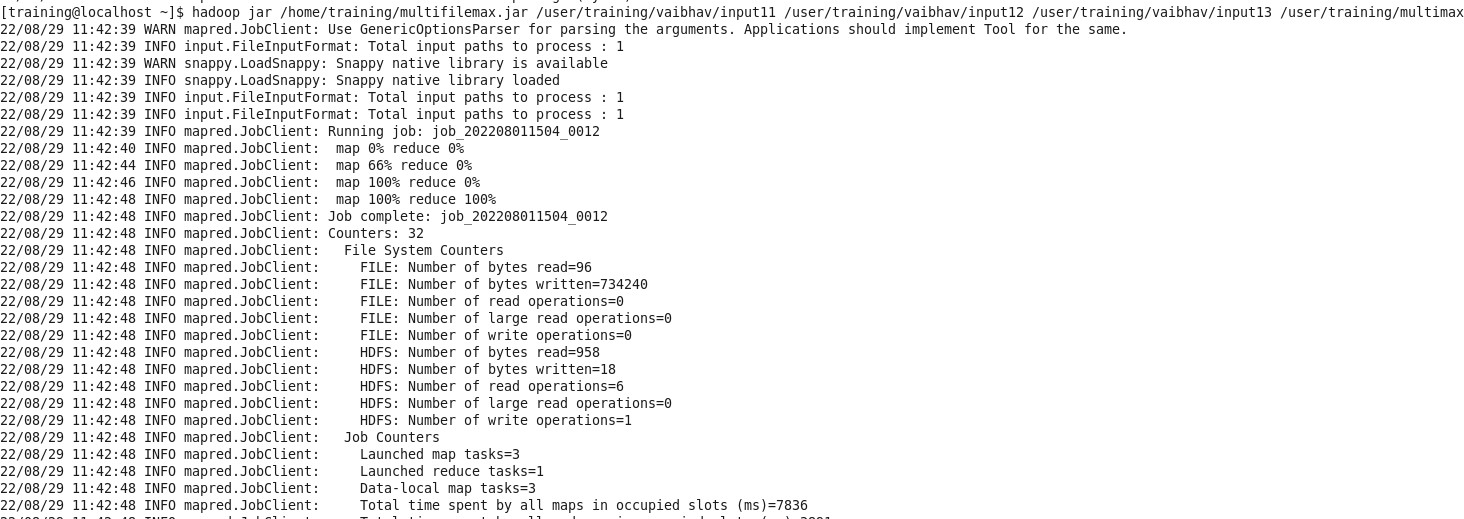
public static void main(String[] args) throws Exception{ Conﬁguration c = new Conﬁguration();

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]); Path p4 = new Path(ﬁles[3]); Job j = new Job(c,"multiple");

j.setJarByClass(MultipleFileInp .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,Map3.class); FileOutputFormat.setOutputPath(j,p4); System.exit(j.waitForCompletion(true)?0:1);}}



## Web Log Analysis - Partitioner Assignment

Date : 01-09-22

For Odd Roll No:

**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 % numReduceTasks ;

}

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

{

return 2% numReduceTasks;

}

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

{

return 3% numReduceTasks;

}

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

{

return 4% numReduceTasks;

}

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

{

return 5% numReduceTasks;

}

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

{

return 6% numReduceTasks;

}

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

{

return 7% numReduceTasks;

}

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

{

return 8% numReduceTasks;

}

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

{

return 9% numReduceTasks;

}

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

{

return 10% numReduceTasks;

}

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

{

return 11% numReduceTasks;

}

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

{

return 12% numReduceTasks;

}

else{

return 0;}}}

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

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

System.exit(0);}@Override

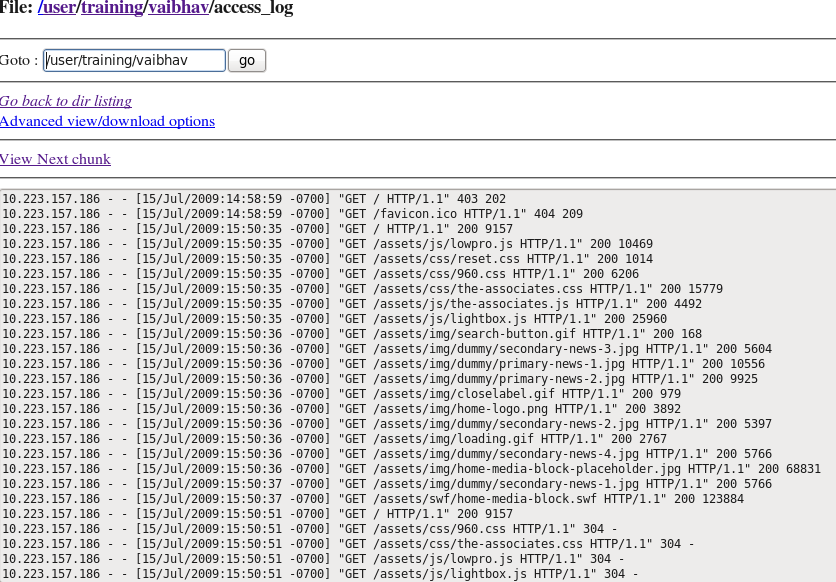
public int run(String[] arg) throws Exception { Configuration conf = getConf() ;

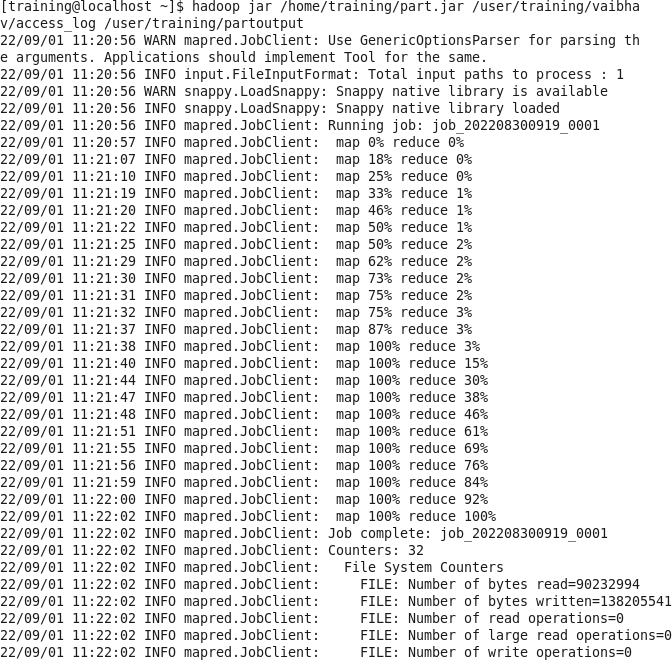
Job job = new Job (conf ,"Partitioner2" ) ; job.setJarByClass(multiFilesPart.class ) ; FileInputFormat.setInputPaths (job ,new Path ( arg [ 0 ])); FileOutputFormat.setOutputPath (job ,new Path ( arg [ 1 ])); job.setMapperClass(Map.class ) ; job.setMapOutputKeyClass ( Text.class ) ; job.setMapOutputValueClass (IntWritable.class ) ;

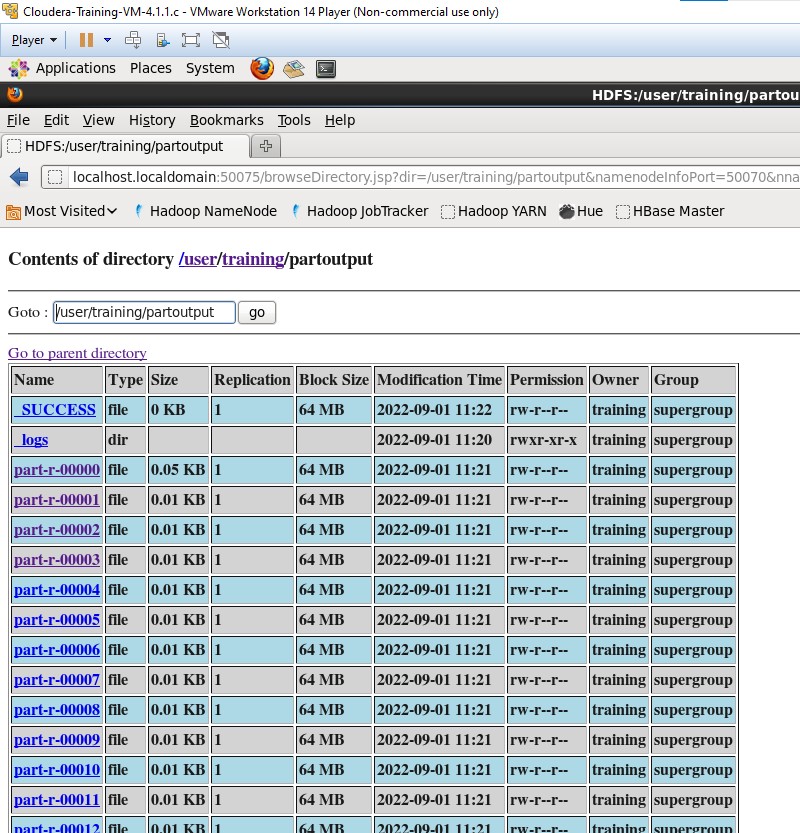
// set partitioner statement job.setPartitionerClass (CaderPartitioner.class ) ; job.setReducerClass(SumReducer.class); job.setNumReduceTasks(13) ;

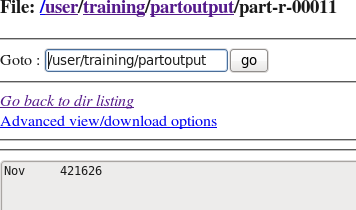
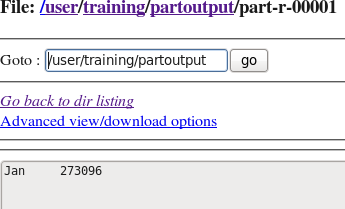
job.setInputFormatClass (TextInputFormat.class ) ; job.setOutputFormatClass(TextOutputFormat.class ) ; job.setOutputKeyClass ( Text.class ) ; job.setOutputValueClass ( Text.class ) ; System.exit(job.waitForCompletion ( true) ? 0 : 1); return 0 ;

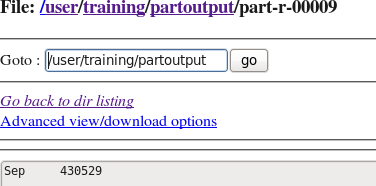
}



}





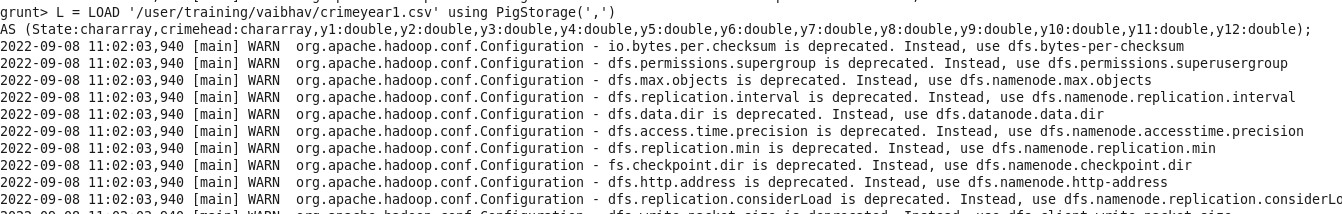


Assignment

Crime Data analysis using Pig

1. To find the total number of crimes which occurred in all the states in the year 2006.

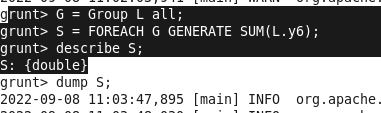
L = LOAD '/user/training/vaibhav/crimeyear1.csv' using PigStorage(',') AS

(State:chararray,crimehead:chararray,y1:double,y2:double,y3:double,y4:double,y5:double,y6:do uble,y7:double,y8:double,y9:double,y10:double,y11:double,y12:double);

grunt> G = Group L all;

grunt> S = FOREACH G GENERATE SUM(L.y6);

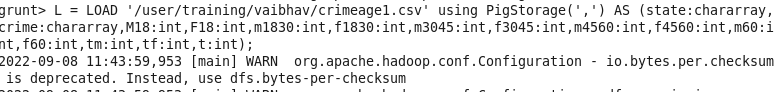
grunt> describe S; S: {double}





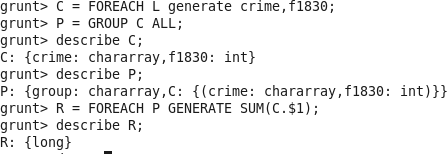
1. Calculate the number of females (age between 18-30 years) who were victims in different crimes in different states.

L = LOAD '/user/training/vaibhav/crimeage1.csv' using PigStorage(',') AS (state:chararray,crime:chararray,M18:int,F18:int,m1830:int,f1830:int,m3045:int,f3045:int,m4560:i nt,f4560:int,m60:int,f60:int,tm:int,tf:int,t:int);



C = FOREACH L generate crime,f1830; P = GROUP C ALL;

R = FOREACH P GENERATE SUM(C.$1);



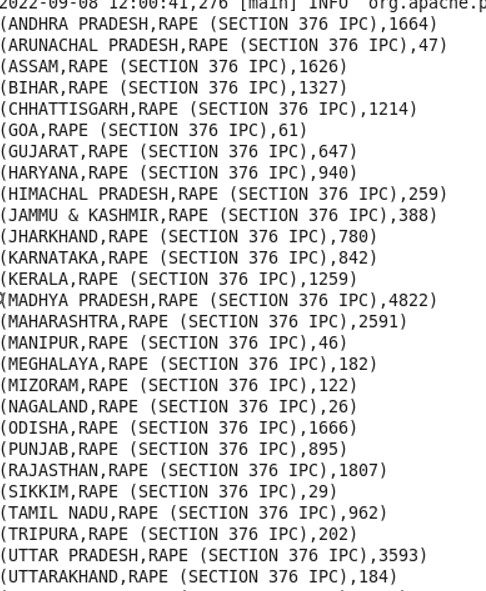
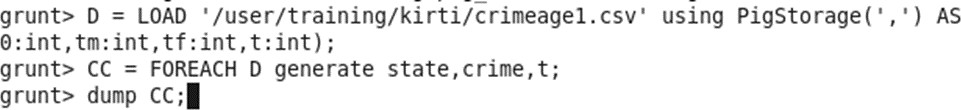


1. Which state have highest rape cases across all years.

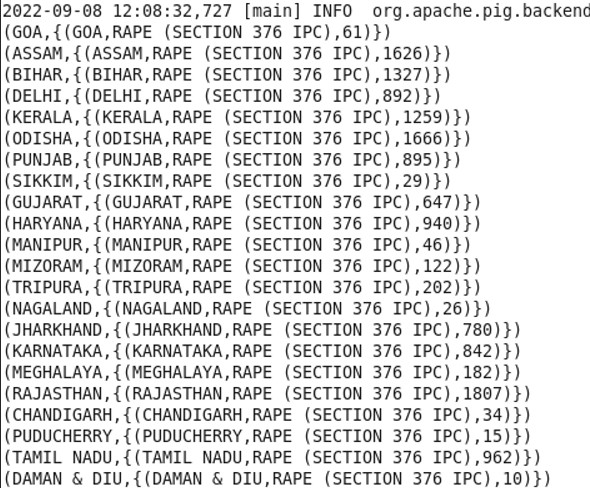
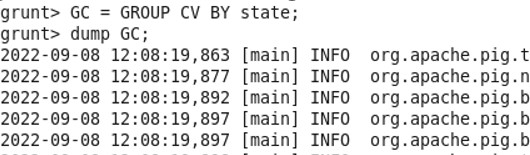
YC = FILTER D BY (crime matches '.\*(RAPE).\*'); dump YC;

CV = FOREACH YC generate state,crime,t; dump CV;

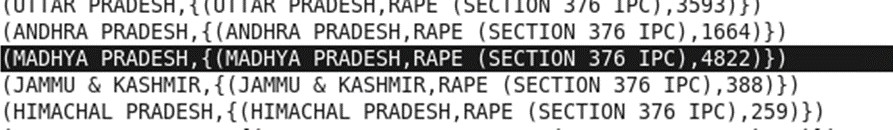
GC = GROUP CV BY state;







Output:



6. Calculate the number of males (age between 45-60 years) who were victims in “Kidnapping and abduction” in different states.

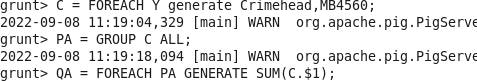
L1 = LOAD '/user/training/vaibhav/crimeage1.csv' using PigStorage(',') AS

(State,Crimehead,MB18,FB18,MB1830,FB1830,MB3035,FB3045,MB4560,FB4560,MA60,FA60, TOTALMALE,TOTALFEMALE);

Y = FILTER L1 BY (Crimehead matches '.("KIDNAPPING AND ABDUCTION").');

C = FOREACH Y generate Crimehead,MB4560; PA = GROUP C ALL;

QA = FOREACH PA GENERATE SUM(C.$1);





**Assignment**

Web log + Cust Analysis using Pig

Date:07-09-2022

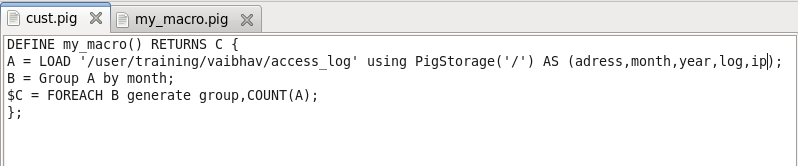
###### Find Out the hit count per month

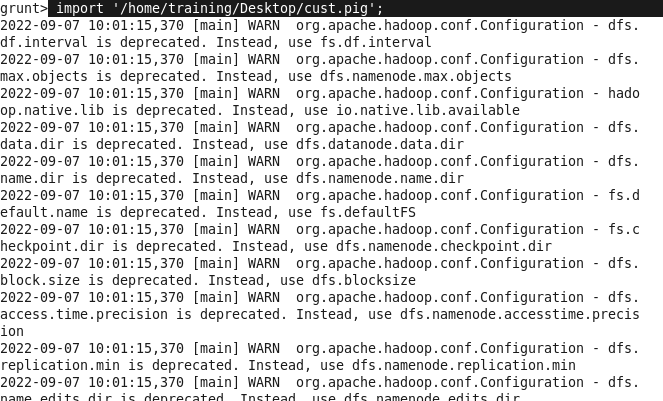
DEFINE my\_macro() RETURNS C {

A = LOAD '/user/training/vaibhav/access\_log' using PigStorage('/') AS (adress,month,year,log,ip);

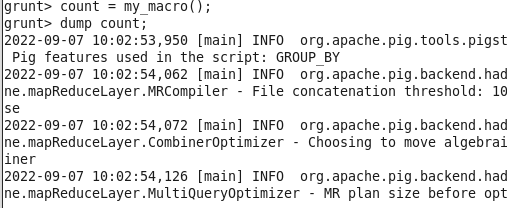
B = Group A by month;

$C = FOREACH B generate group,COUNT(A);

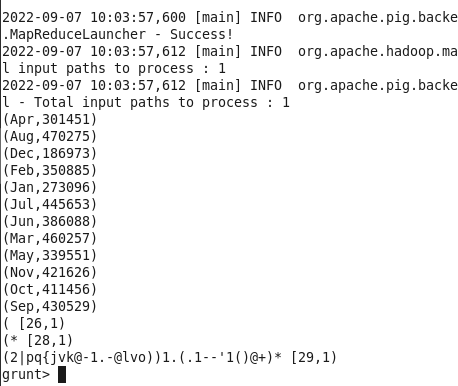
};

import ‘/desktop/cust.pig’; Count = my\_macro();

dump count;



**Output:**



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

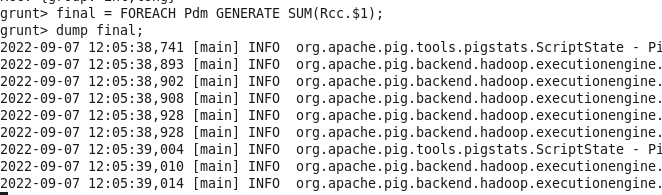
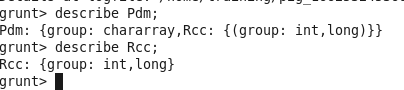
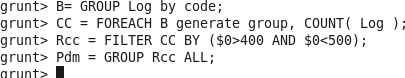
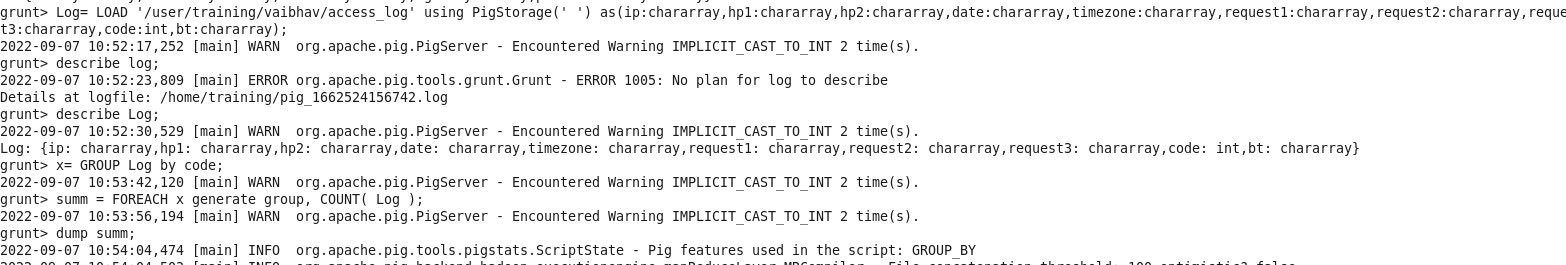
Log= LOAD '/user/training/vaibhav/access\_log' using PigStorage(' ') as(ip:chararray,hp1:chararray,hp2:chararray,date:chararray,timezone:chararray,request1:chararr ay,request2:chararray,request3:chararray,code:int,bt:chararray);

B= GROUP Log by code;

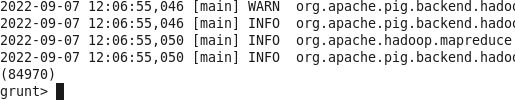
CC = FOREACH B generate group, COUNT( Log ); Rcc = FILTER CC BY ($0>400 AND $0<500);

Pdm = GROUP Rcc ALL;

ﬁnal = FOREACH Pdm GENERATE SUM(Rcc.$1);



Output:



#### Customer dataset:

**Find out age group-wise customer count**

L = LOAD '/user/training/vaibhav/custs' using PigStorage(',') AS (id,fname,lname,age,profession); X = FILTER L BY (age>30 AND age<=40);

G = GROUP X BY age;

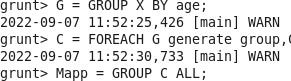
C = FOREACH G generate group,COUNT(X); Mapp = GROUP C ALL;

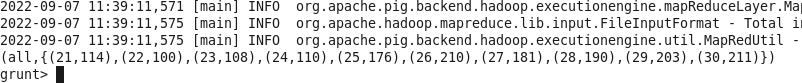
describe Mapp; dump Mapp;

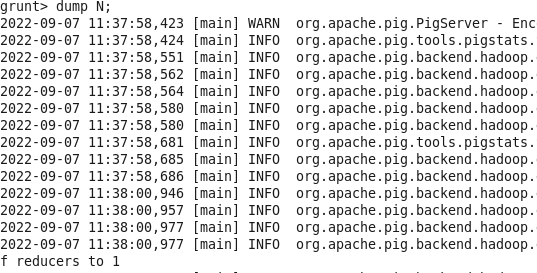
N = FOREACH Mapp GENERATE SUM(C.$1);

dump N;









#### Output:

