

***Department of Data Science & Technology***

***Certificate***

***Master in Computer Applications***

***Trimester IV (2021 – 22)***

***This is to certify that Ms. Saayli Yadav Roll No. 47 of MCA, has satisfactorily completed the practical course “ Big Data Analytics” prescribed by the College for the Partial Fulfillment of the Degree by the Somaiya Vidyavihar University, during the academic year 2021-23.***

***Signature of the Faculty-Incharge Signature of the Programme***

***(Dr. Bharati Wukkadada W) Coordinator***

***(Dr. Bharati Wukkadada W)***

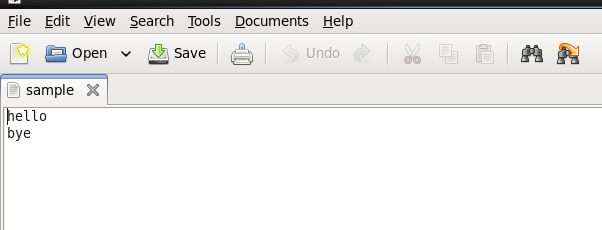
***Date of Examination Signature of the External Examiner/s***

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Contents** | **Date Of Submission** | **Page Number** |
| 1 | Word count | 08- Aug-2022 | 3 |
| 2 | Write a program to find average word length | 16-Aug-2022 | 8 |
| 3 | Encrypt the email | 23-Aug-2022 | 13 |
| 4 | 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. | 29-Aug-2022 | 20 |
| 5 | Find Out the hit count per month | 30-Aug-2022 | 31 |
| 6 | Write a MapReduce program to find profession-wise (choose any 5 professions) of customer count from the "cust" dataset. | 02-Sep-2022 | 40 |
| 7 | Find hit per month using pig | 07-Sep-2022 | 49 |
| 8 | Crime data analysis using pig | 10-Sep-2022 | 53 |

**PRACTICAL – 1**

**Assignment – word Count**



Program

**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** word\_count {

**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(word\_count.**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**);

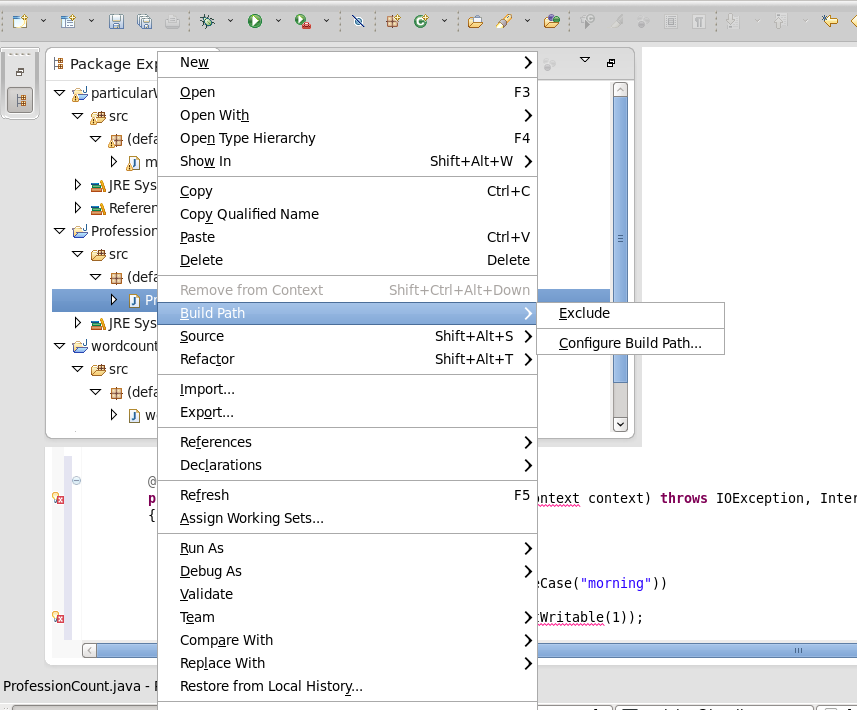
**boolean** success = job.waitForCompletion(**true**);

System.*exit*(success ? 0 : 1);

}

}

Create jar



Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

In terminal

Text, letter

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

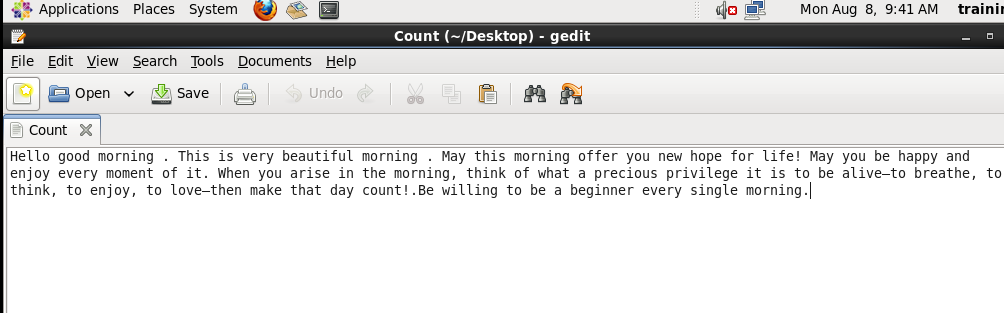
Graphical user interface, text, application, email

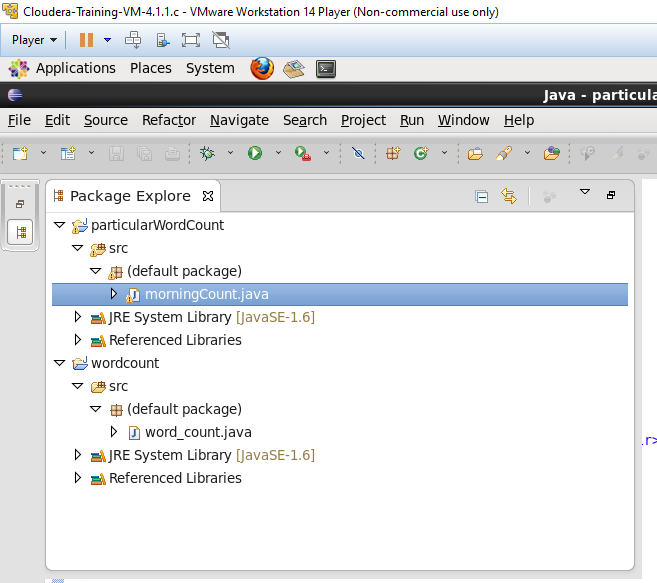
Description automatically generated

**PRACTICAL – 2**

**Assignment - Specific Word Count**

Create Text file





Program

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.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;

//import morningCount.SumReducer;

//import morningCount.WordMapper;

public class morningCount {

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 && word.equalsIgnoreCase("morning"))

{

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(morningCount.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);

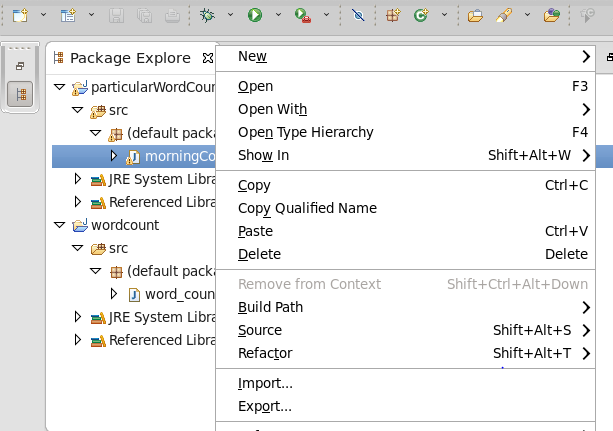
boolean success = job.waitForCompletion(true);

System.exit(success ? 0 : 1);

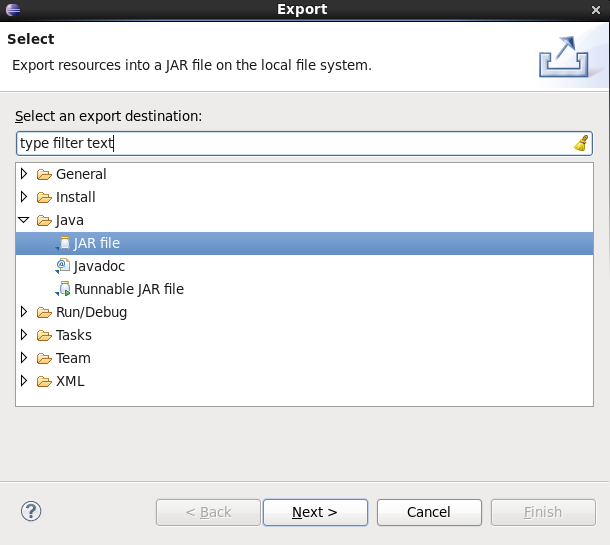
}

}

Create jar

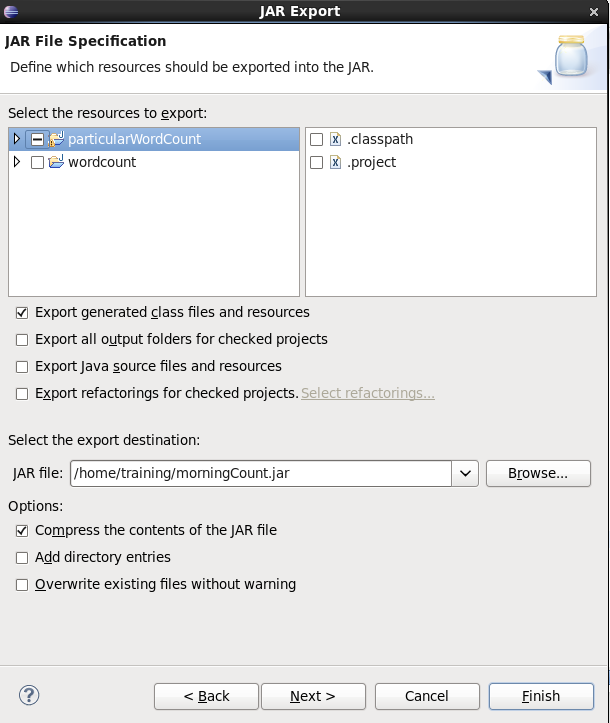


Right click on file -> export



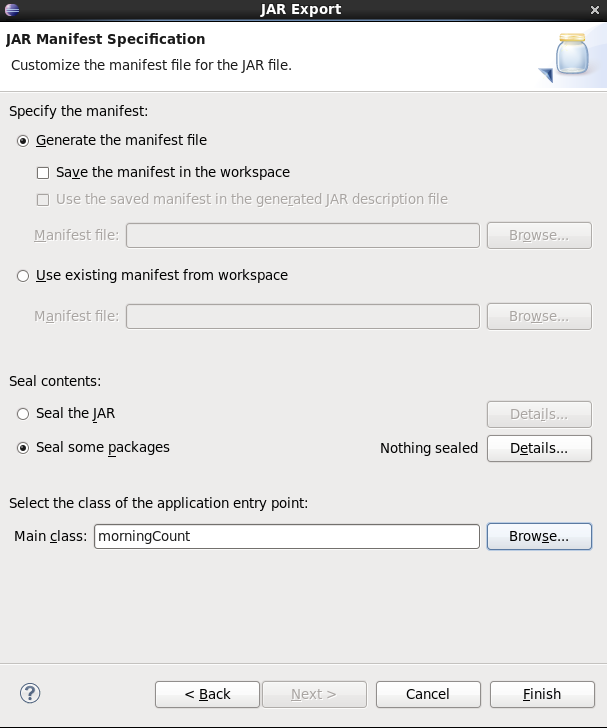
Java -> jar file

Next ->



Give jar name

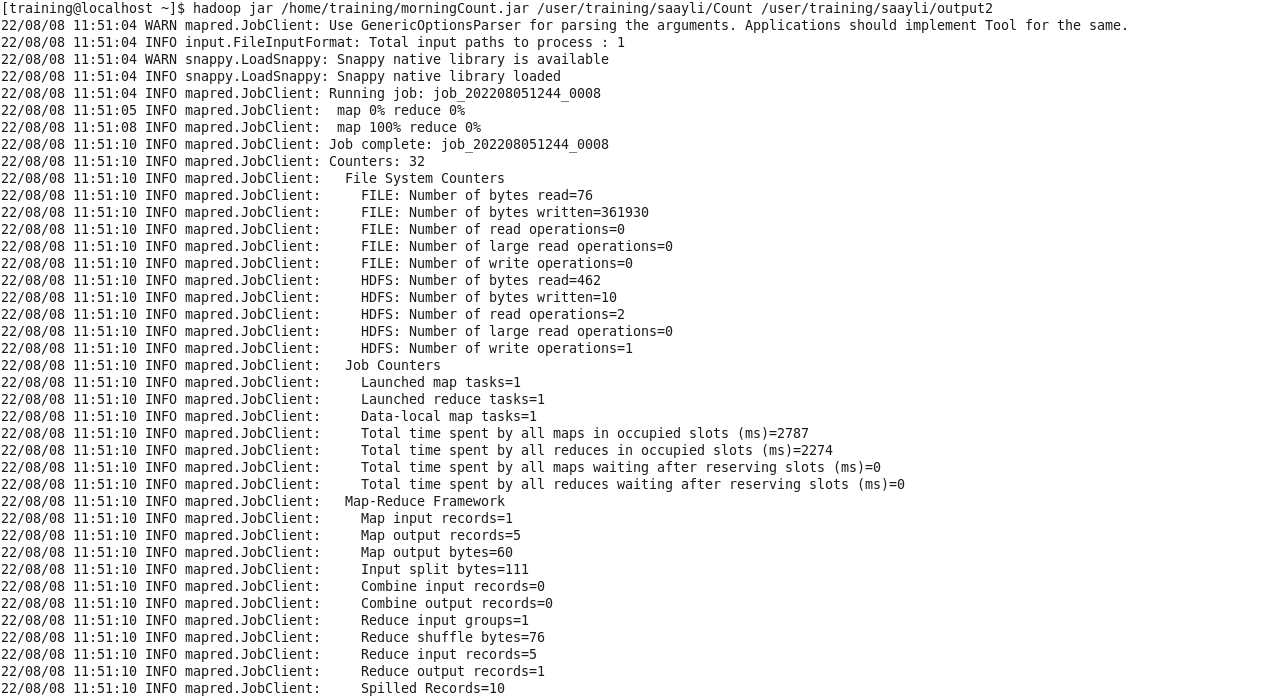
Next



Select main class

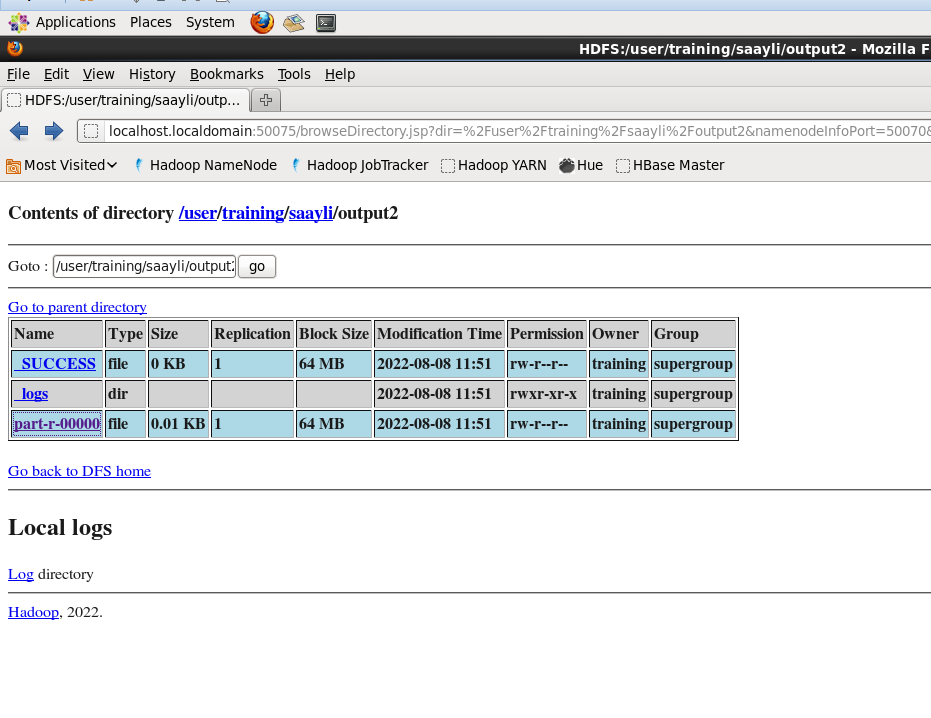
Finish

Then in terminal

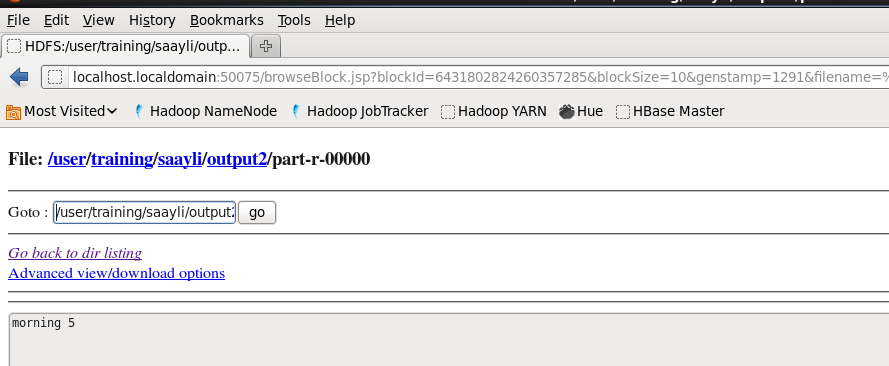




Go to hdfs



Click on part - r- 0000



**PRACTICAL – 3**

**Assignment : 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**);

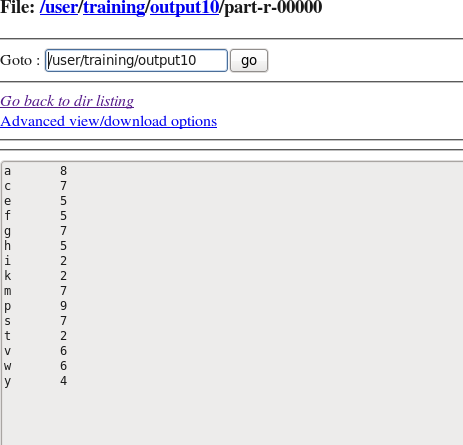
job.setOutputKeyClass(Text.**class**); job.setOutputValueClass(IntWritable.**class**);

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

}

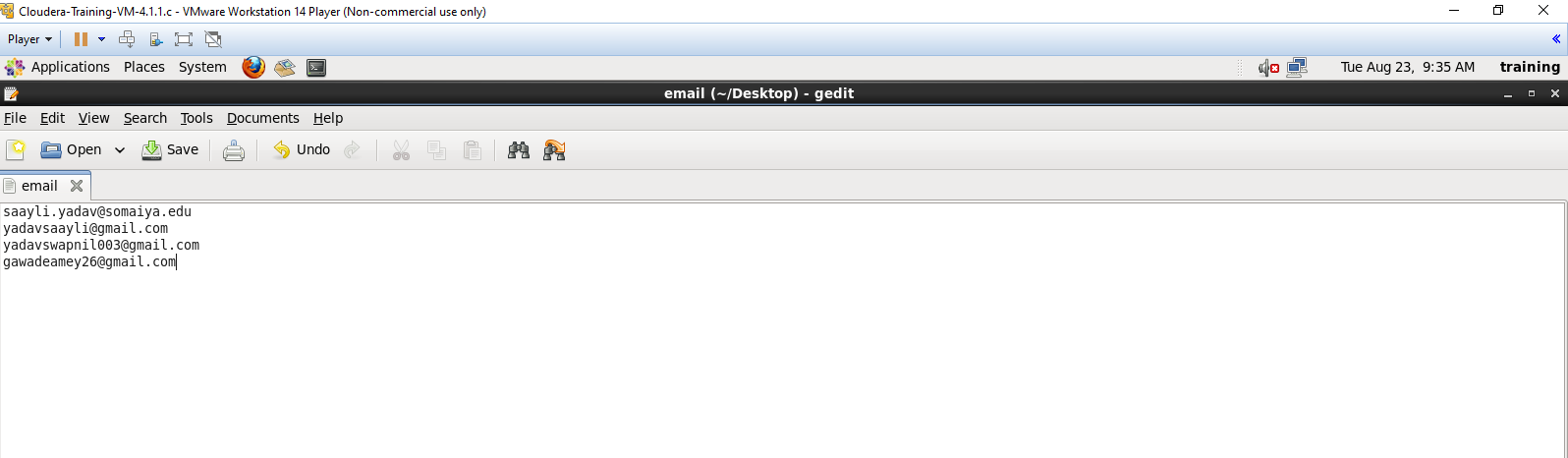
}



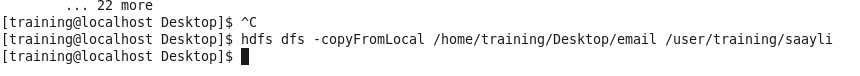


**PRACTICAL – 4**

**Assignment : Encrypt the emails using only mapper**



In terminal



Copy the jar files from classroom to vmware desktop

Graphical user interface, application

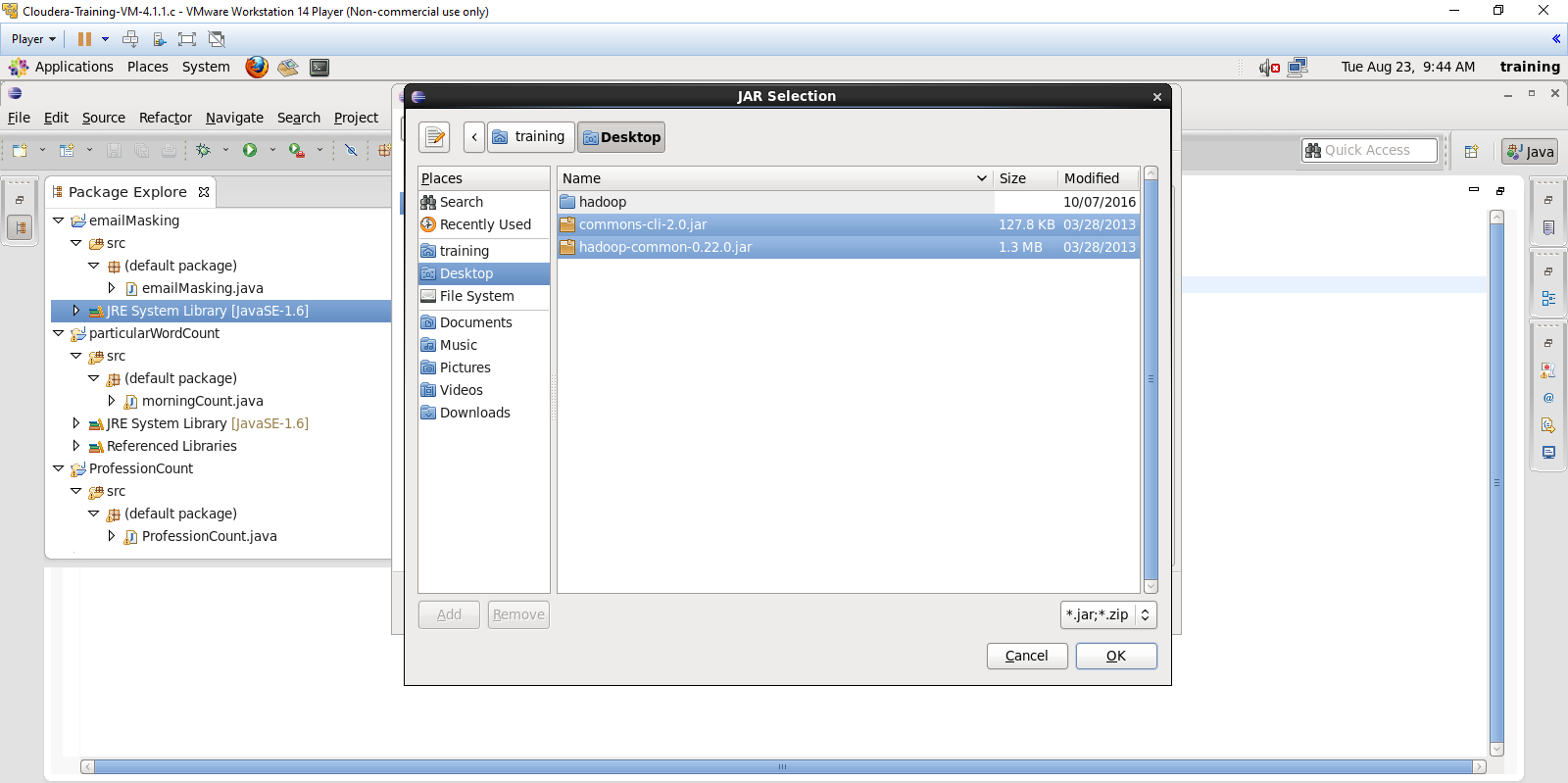
Description automatically generated

Create the java file

Graphical user interface, text, application, Word

Description automatically generated

Right click on file -> JRE -> Build Path -> configure path -> add external jar files -> add the jar files



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.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;

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

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

**class** emailMasking {

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

{

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

InterruptedException

{

String text = value.toString();

String result="";

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

{

**if** (Character.*isUpperCase*(text.charAt(i)))

{

**char** ch = (**char**)(((**int**)text.charAt(i) +

3 - 65) % 26 + 65);

result= result+ch;

}

**else**

{

**char** ch = (**char**)(((**int**)text.charAt(i) +

3 - 97) % 26 + 97);

result= result+ch;

}

}

String line1 = result + ".com";

String value1 = "";

Text outputKey = **new** Text(line1);

con.write(outputKey,**new** Text(value1));

}

}

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

{

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

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

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

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

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

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

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

j.setNumReduceTasks(0);

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

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

FileInputFormat.*addInputPath*(j,input);

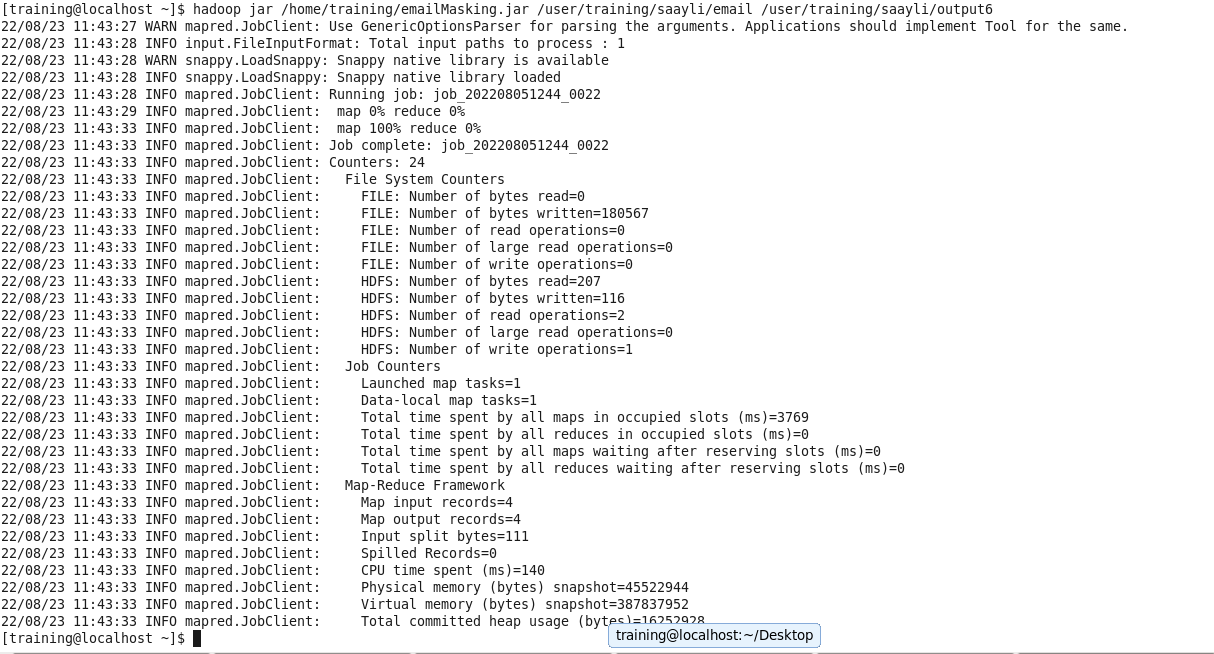
FileOutputFormat.*setOutputPath*(j,output);

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

}

}

Logic of code: ceasar cipher



Output:

Graphical user interface, text, application, email

Description automatically generated

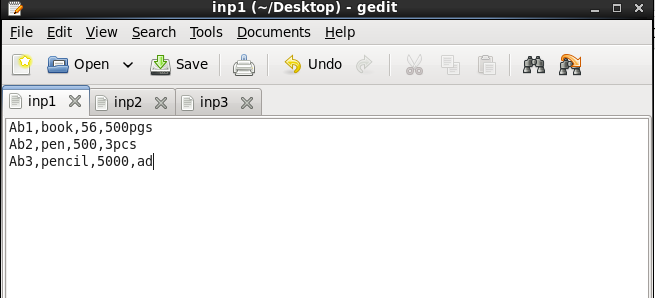
Graphical user interface, text, application, email

Description automatically generated

**PRACTICAL – 5**

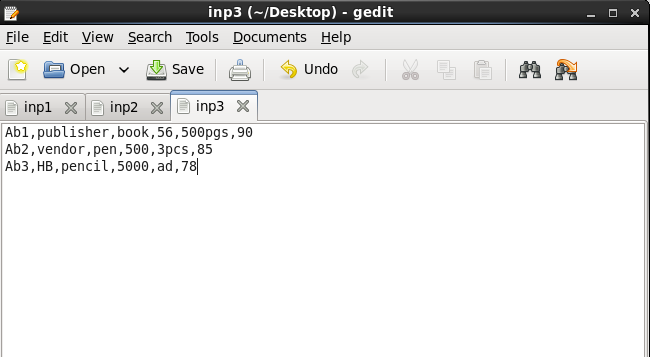
**Assignment :** **Create 2 or 3 input files on your own , in which the data is present in different format. Write a program to process the these files using different map class and perform any one aggerate function like sum, max, min etc. on it.**

Input file

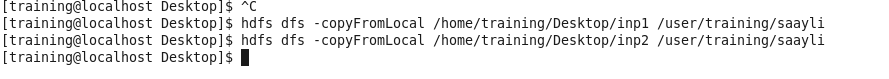


Graphical user interface, text, application, email

Description automatically generated



Copy to the folder



Graphical user interface, text, application

Description automatically generated

To find minimum

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.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;

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

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

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

**import** java.util.Arrays;

**public** **class** maxMultipleInput {

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

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

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** price = Integer.*parseInt*(line1[3]);

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

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** price = Integer.*parseInt*(line1[5]);

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

con.write(outputKey, outputValue);

}

}

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

{

**public** **void** reduce(Text name, Iterable<IntWritable> price, Context con) **throws** IOException, InterruptedException

{

// float sum = 0;

// for(IntWritable value : price)

// {

// sum += value.get();

// }

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

// int max = Integer.MIN\_VALUE;

**int** min = 9999;

**for**(IntWritable value: price)

{

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

}

con.write(name, **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(maxMultipleInput.**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);

}

}

}

Ouput :

Graphical user interface, text, application, email

Description automatically generated

To find Maximum

**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.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;

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

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

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

**import** java.util.Arrays;

**public** **class** maxMultipleInput {

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

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

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** price = Integer.*parseInt*(line1[3]);

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

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** price = Integer.*parseInt*(line1[5]);

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

con.write(outputKey, outputValue);

}

}

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

{

**public** **void** reduce(Text name, Iterable<IntWritable> price, Context con) **throws** IOException, InterruptedException

{

// float sum = 0;

// for(IntWritable value : price)

// {

// sum += value.get();

// }

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

// int max = Integer.MIN\_VALUE;

**int** min = 0;

**for**(IntWritable value: price)

{

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

}

con.write(name, **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(maxMultipleInput.**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);

}

}

}

Output :

Graphical user interface, text, application, email

Description automatically generated

**PRACTICAL – 6**

**Assignment : Find Out the hit count per month**

Input File

Text

Description automatically generated

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** countHitInMonth **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{

// **TODO** Auto-generated method stub

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

System.*exit*(0);

}

@Override

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

// **TODO** Auto-generated method stub

Configuration conf = getConf() ;

Job job = **new** Job (conf ,"countHitInMonth" ) ;

job.setJarByClass(countHitInMonth.**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 ;

}

}

Output :

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

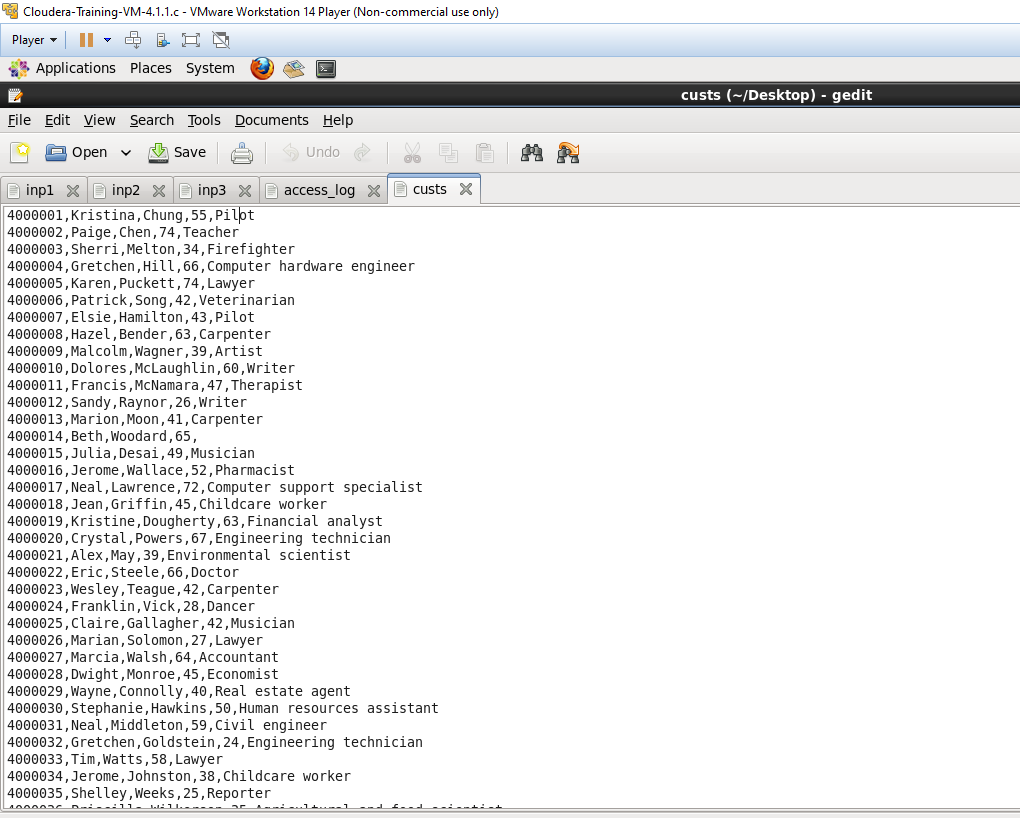
**PRACTICAL – 7**

**Assignment : Write a MapReduce program to find profession-wise (choose any 5 professions) of customer count from the "cust" dataset.**

Copy the custs data to hdfs



Input file



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** customerProfessionCount **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[word.length-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("Pilot"))

{

**return** 1 % numReduceTasks ;

}

**else** **if** (key.toString().equals("Teacher"))

{

**return** 2% numReduceTasks;

}

**else** **if** ( key.toString().equals("Lawyer") )

{

**return** 3% numReduceTasks;

}

**else** **if** ( key.toString().equals("Artist") )

{

**return** 4% numReduceTasks;

}

**else** **if** ( key.toString().equals("Therapist") )

{

**return** 5% numReduceTasks;

}

**else**{

**return** 0;

}

}

}

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

// **TODO** Auto-generated method stub

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

System.*exit*(0);

}

@Override

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

// **TODO** Auto-generated method stub

Configuration conf = getConf() ;

Job job = **new** Job (conf ,"customerProfessionCount" ) ;

job.setJarByClass(customerProfessionCount.**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(6) ;

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 ;

}

}

Output :

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated



**PRACTICAL – 8**

**Assignment : Find Out the hit count per month using pig**

Input file :

Text

Description automatically generated

Pig file

Text

Description automatically generated

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

Graphical user interface, text, application, email

Description automatically generated

Q . Count age wise from custs

Pig File :

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

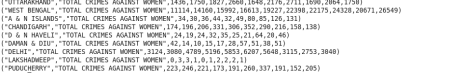
**PRACTICAL – 9**

**Assignment : Solve crime dataset using pig. Solve any four questions**

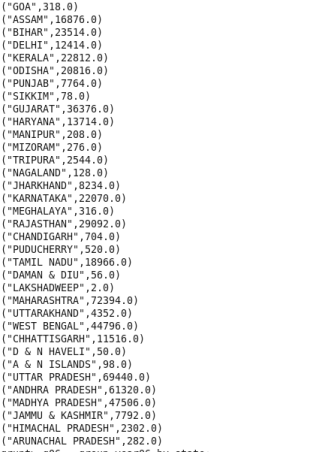
1. To find the total number of crimes which occurred in all the states in the year 2006

year06 = load '/user/training/saayli/crimeyear.csv' using PigStorage(',') as

(state,crime,y\_2001,y\_2002,y\_2003,y\_2004,y\_2005,y\_2006,y\_2007,y\_2008,y\_2009,y\_2010); 



47 - Saayli Yadav

47 - Saayli Yadav

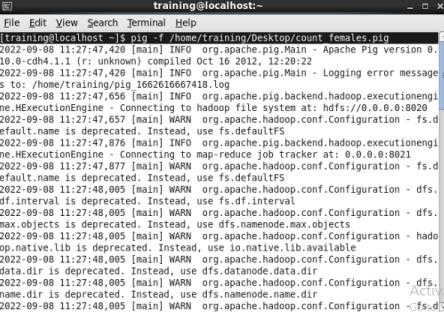
2. Calculate the number of females (age between 18-30 years) who were victims in different crimes in different states.

FOREACH crime2006 generate group, SUM( crimeage.$5 );

crimeage = LOAD '/user/training/input/crimeage' using PigStorage(',') AS (STATE:chararray,crimehead:chararray,mb18yr:chararray,fb18yr:int,mb18\_30yr:int,fb18\_30yr:int ,mb30\_45yr:int,fb30\_45yr:int,mb45\_60yr:int,fb45\_60yr:int,mb60\_abyr:int,fb60\_abyr:int,totalmale :int,totalfemale:int,grandtotal:int);

crimef1830 = GROUP crimeage by (STATE,crimehead);

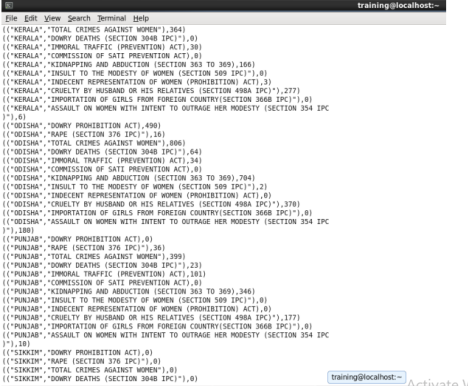
cf1830 = FOREACH crimef1830 generate group, SUM( crimeage.$5 );

Output:

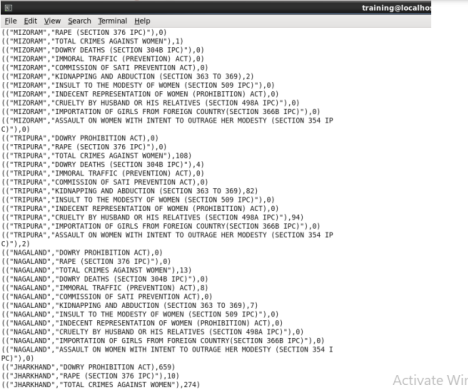
47 - Saayli Yadav

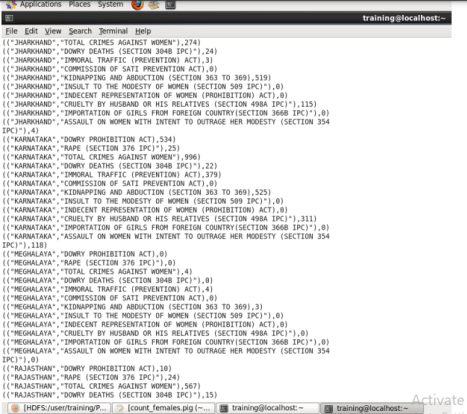
47 - Saayli Yadav

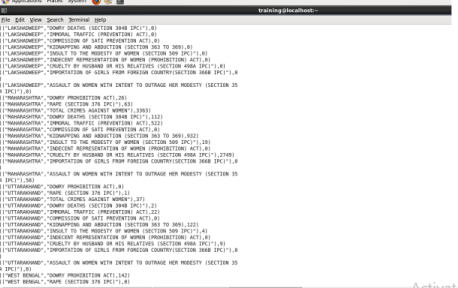
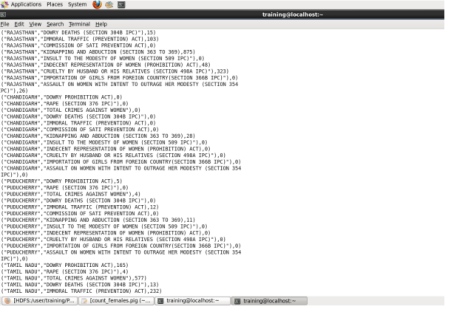
47 - Saayli Yadav

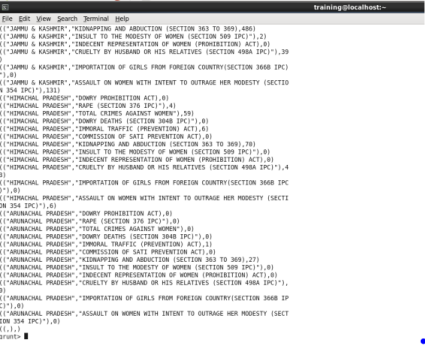
47 - Saayli Yadav

47 - Saayli Yadav

47 - Saayli Yadav

47 - Saayli Yadav

47 - Saayli Yadav

4. Find crime wise which state is having highest count



Output :

47 - Saayli Yadav