**Practical 1**

**Problem :-**

**Write a map reduce code to count a specific character**

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.mapreduce.Mapper; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.mapreduce.Reducer;

public class specificCount {

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+")){ word=word.toLowerCase();

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

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(specificCount.class); job.setJobName("specificCount");

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);

}

}

Hdfs code:-

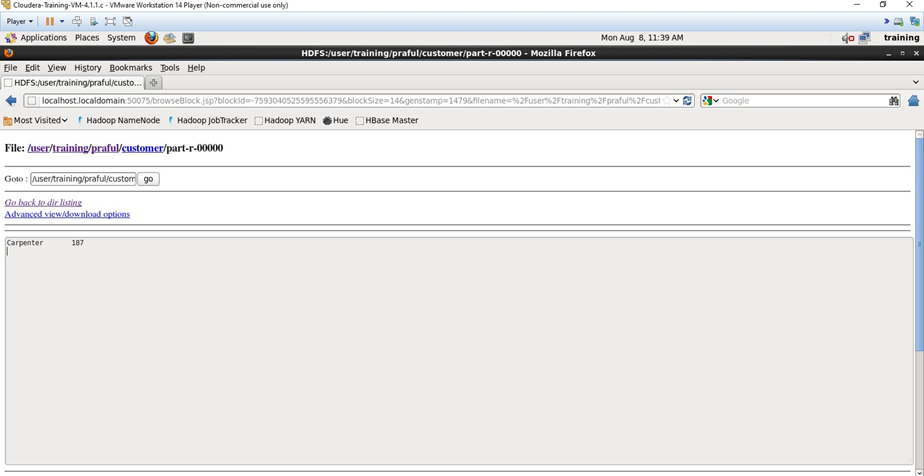
To copy from Local To HDFS:-

hdfs dfs -copyFromLocal /home/training/Desktop/praful/specific\_count /user/training/Praful

To start count jar

$ hadoop jar /home/training/Desktop/praful/custsOM.jar /user/training/praful/custs

/user/training/praful/customer



**2) To Count Specific Character:-**

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.mapreduce.Mapper; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.mapreduce.Reducer;

public class specificCount {

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+")){ word=word.toLowerCase();

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

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(specificCount.class); job.setJobName("specificCount");

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

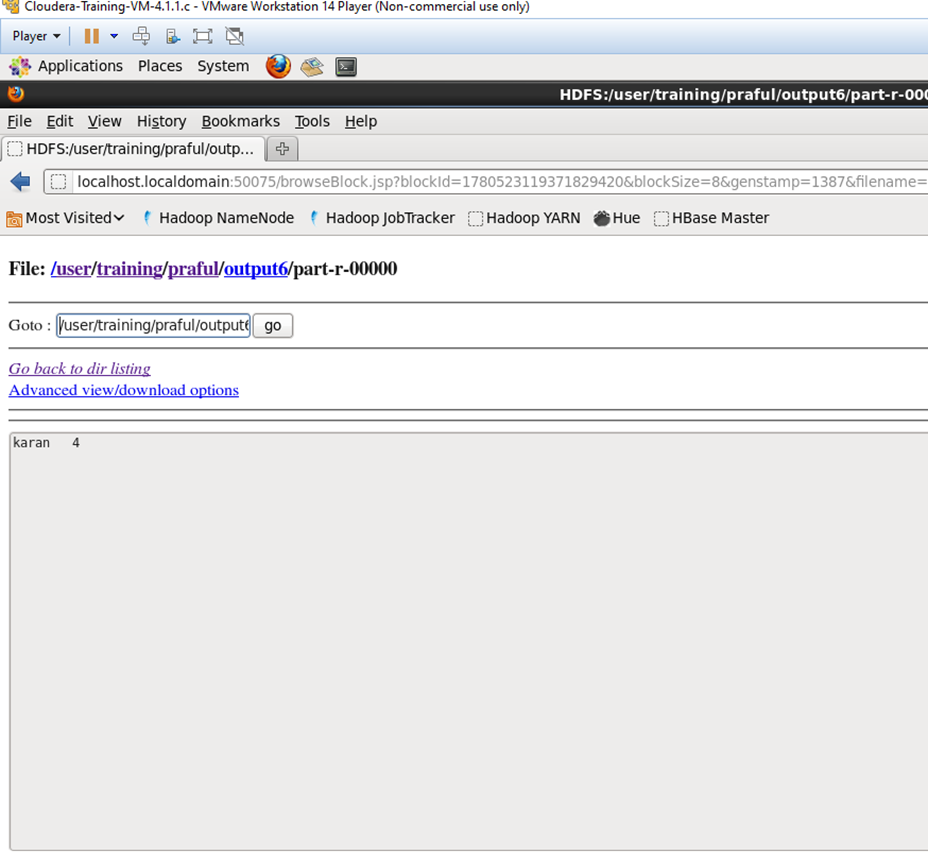
job.setMapperClass(WordMapper.class); job.setReducerClass(SumReducer.class);

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

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

}

}



**PRACTICAL 2**

**PROBLEM:-**

**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);

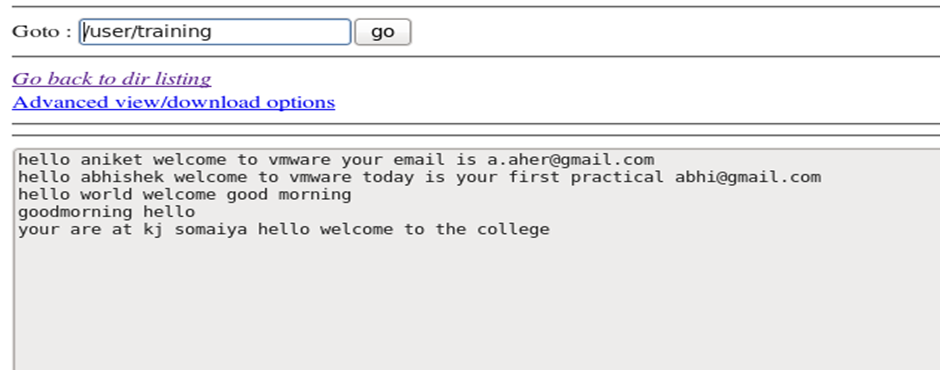
job.setOutputValueClass(IntWritable.class);

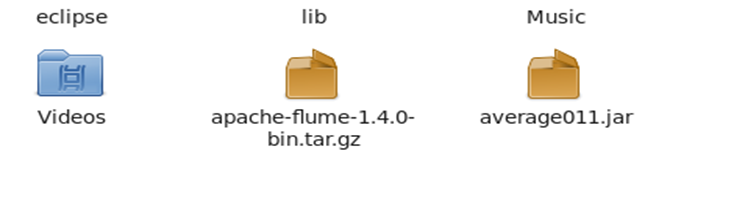
boolean success = job.waitForCompletion(true);

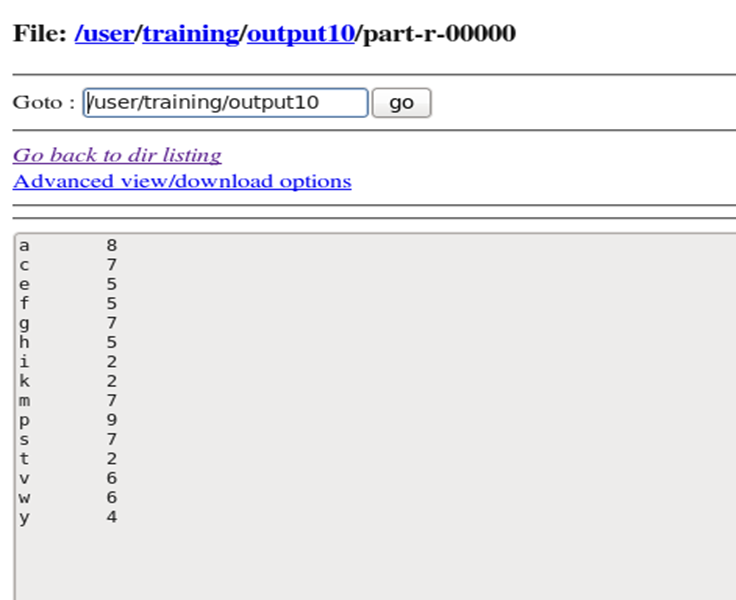
System.*exit*(success ? 0 :1);

}

}







**PRACTICAL 3**

**PROBLEM:-**

**Write a map reduce code to Encrypt Email.**

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.util.GenericOptionsParser;

public class encrypt {

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

{

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

{

line = value.toString();

String line1 = line.replaceAll("[a-z]","1"); Text outputKey = new Text(line1);

Text outputValue = new Text(""); con.write(outputKey, outputValue);

}//end of map()

}//end of mapper class

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

{

Configuration c = new Configuration();

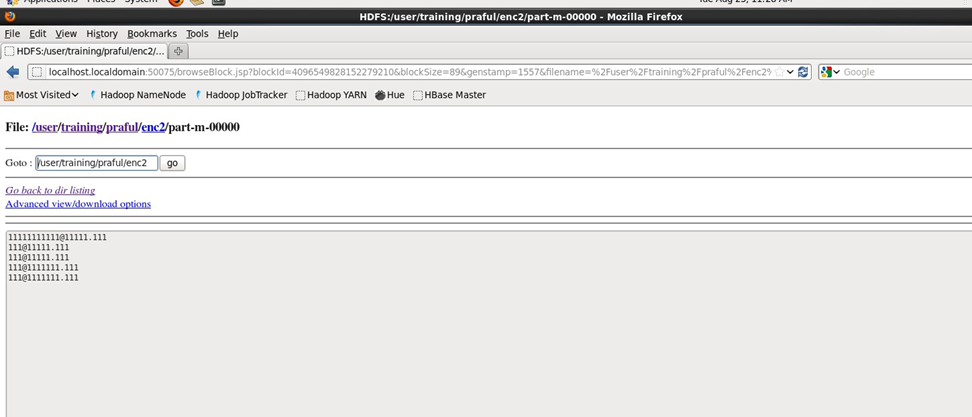
String[] files = new GenericOptionsParser(c,args).getRemainingArgs(); Path input = new Path(files[0]);

Path output = new Path(files[1]); Job j = new Job(c,"FormatEmpFile"); j.setJarByClass(encrypt.class);

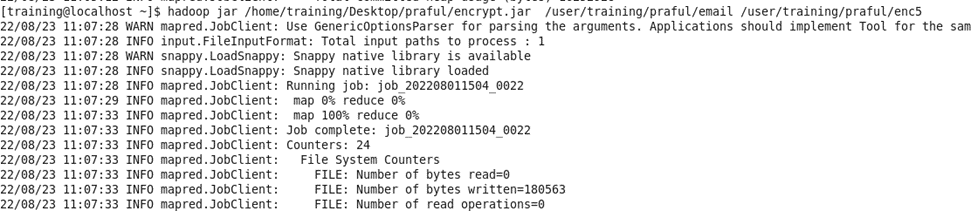
j.setMapperClass(MapForWordCount.class); j.setNumReduceTasks(0); j.setMapOutputKeyClass(Text.class); j.setMapOutputValueClass(Text.class); FileInputFormat.*addInputPath*(j, input); FileOutputFormat.*setOutputPath*(j, output); System.*exit*(j.waitForCompletion(true)?0:1);

}

}



Hadoop CMD:-



Logic :-

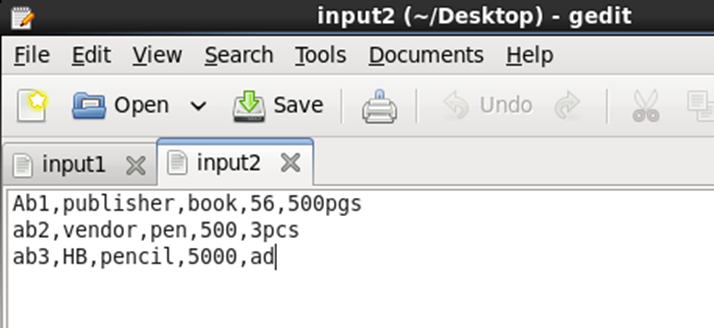
We are converting the the all alphabet[a-z] to 1.

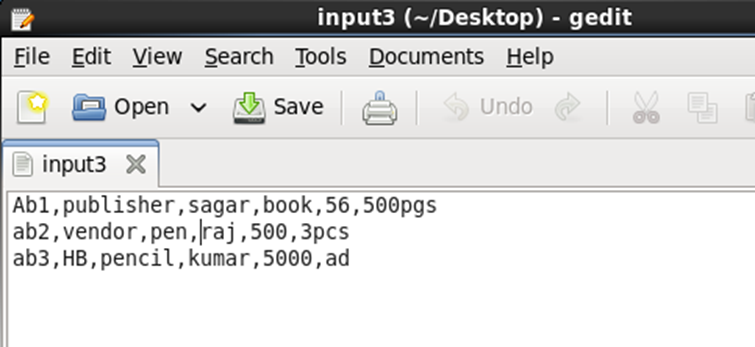
And working on Loop for converting a = 1, b = 2, c = 3 and so On

**Practical 4**

**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.**







Putting the files into hadoop file system





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 MultipleFileInput {

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(MultipleFileInput.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);

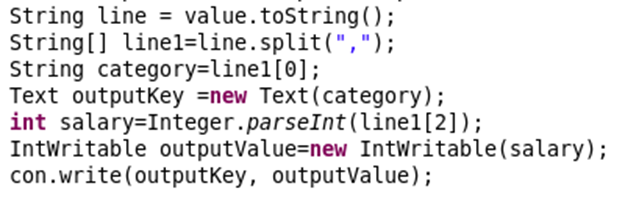
FileOutputFormat.setOutputPath(j, p4);

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

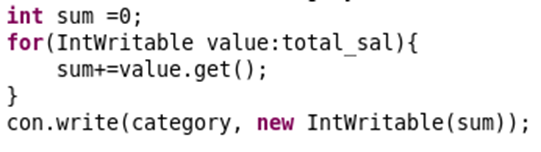
}

}

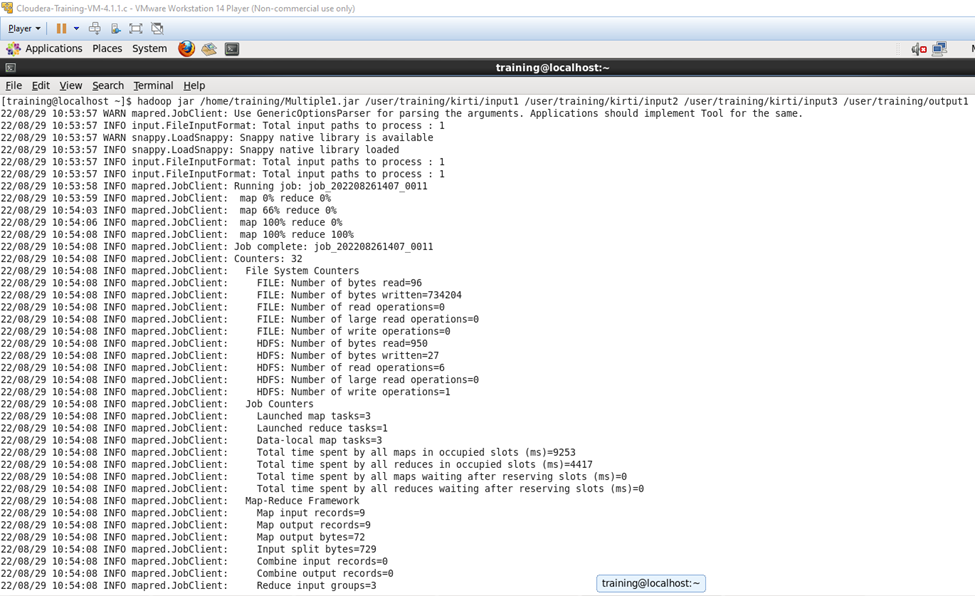
Map Logic



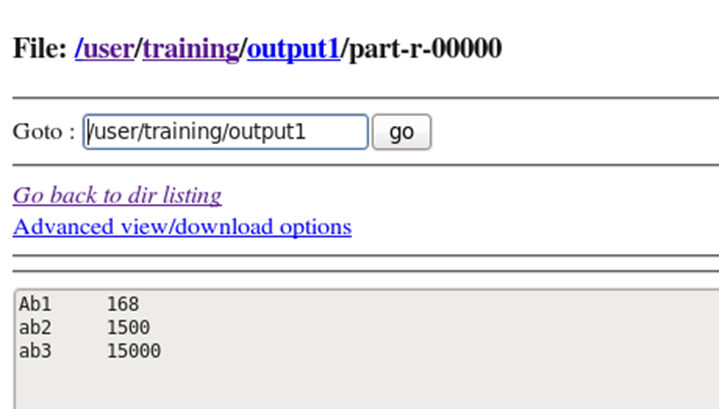
Reducer Logic



Terminal Output:-



Output File:-



**Practical 5**

**Problem**

**Find Out the hit count peí month**

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{

// TODO Auto-generated method stub

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

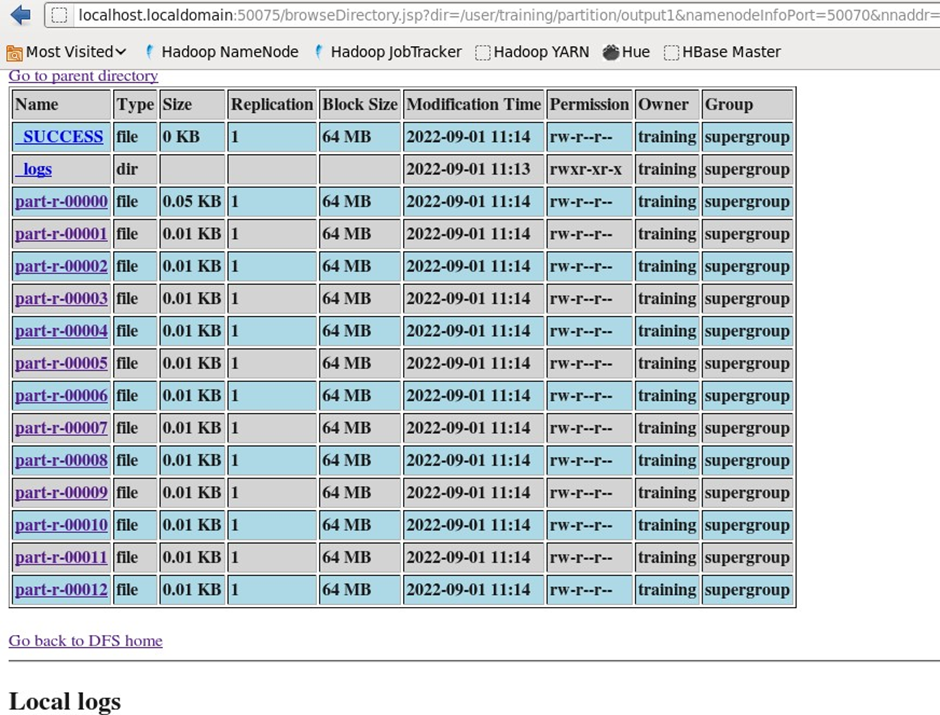
job.setOutputFormatClass(TextOutputFormat.class ) ; job.setOutputKeyClass ( Text.class ) ; job.setOutputValueClass ( Text.class ) ; System.*exit*(job.waitForCompletion ( true) ? 0 : 1); return 0 ;

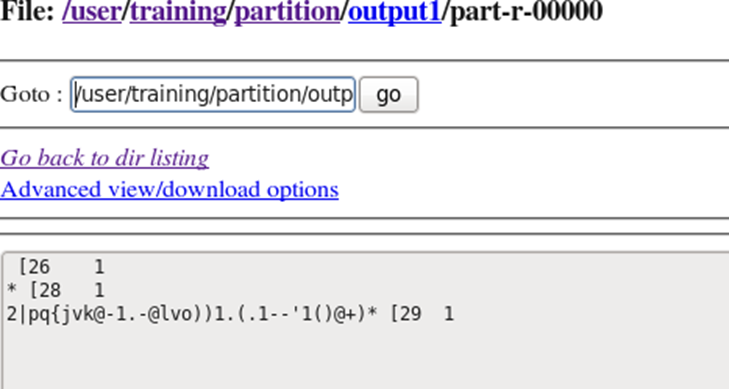
}

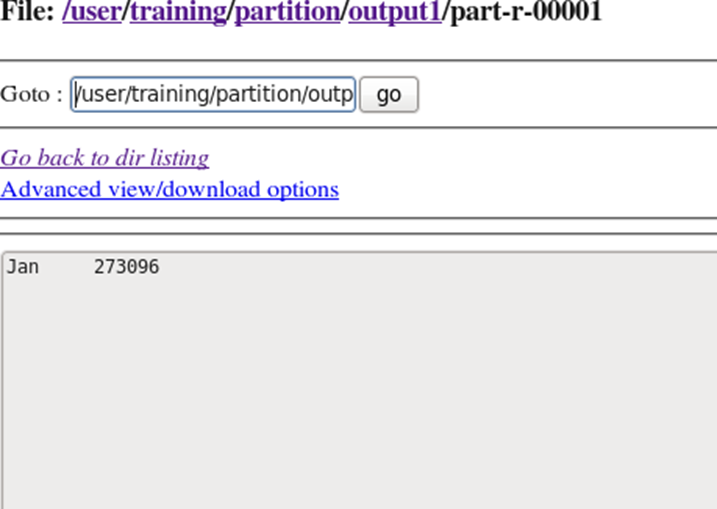
}

# In hdfs :-

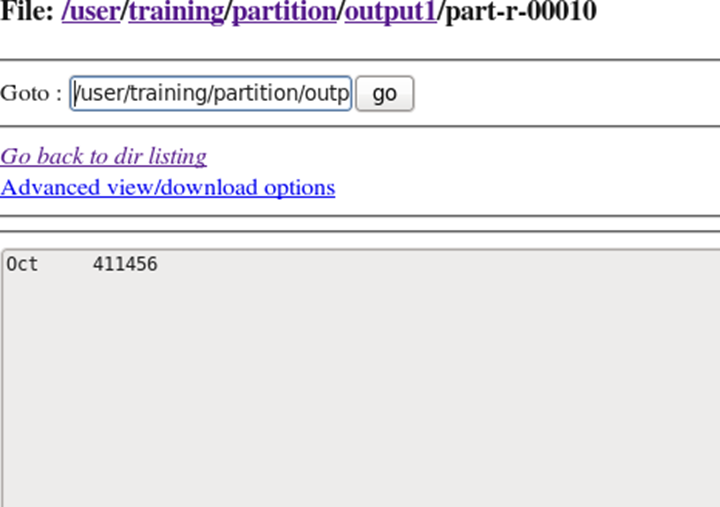
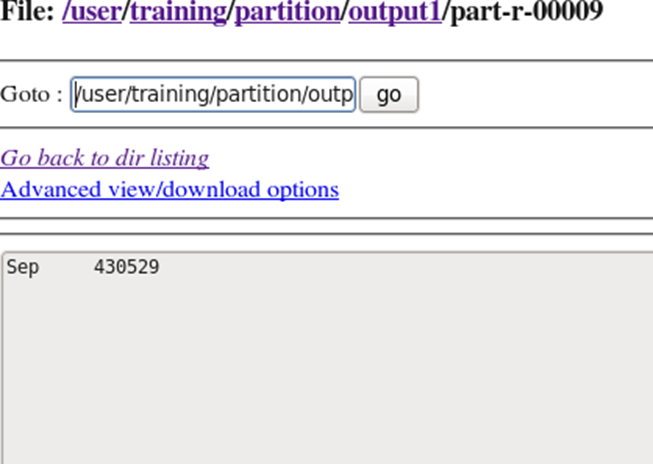
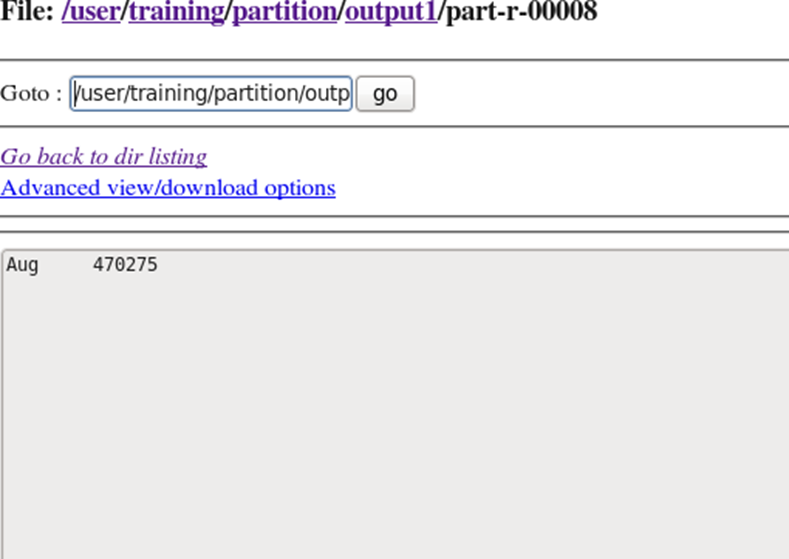
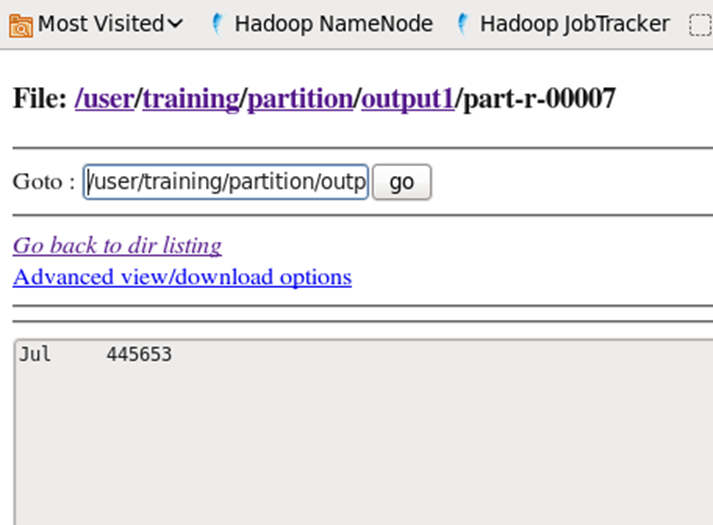
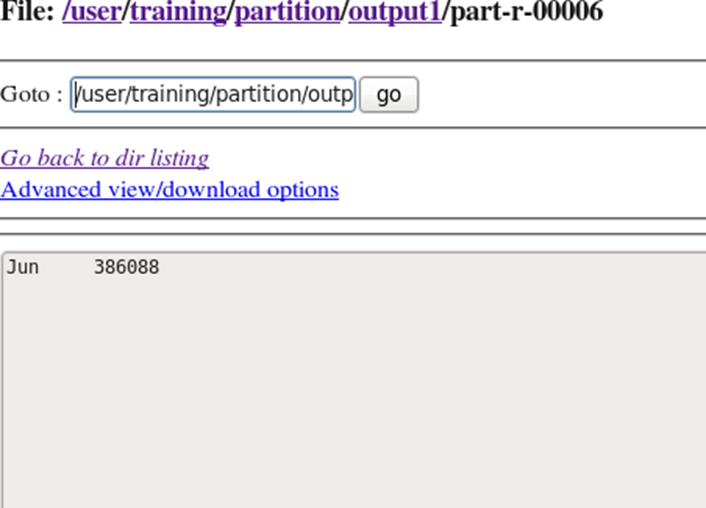
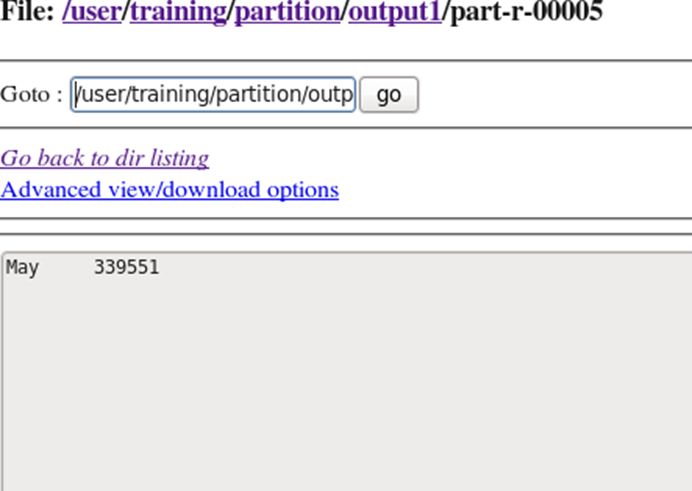
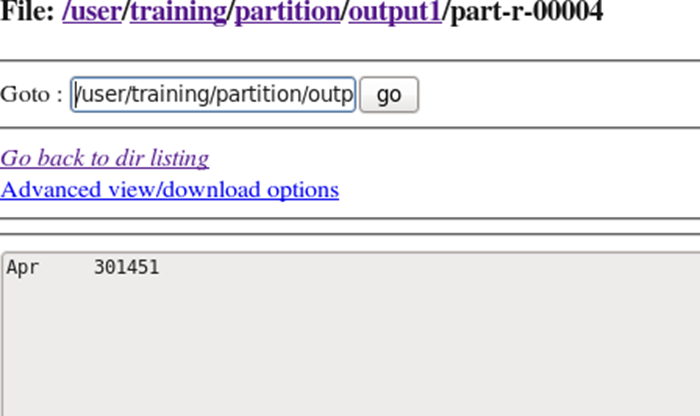
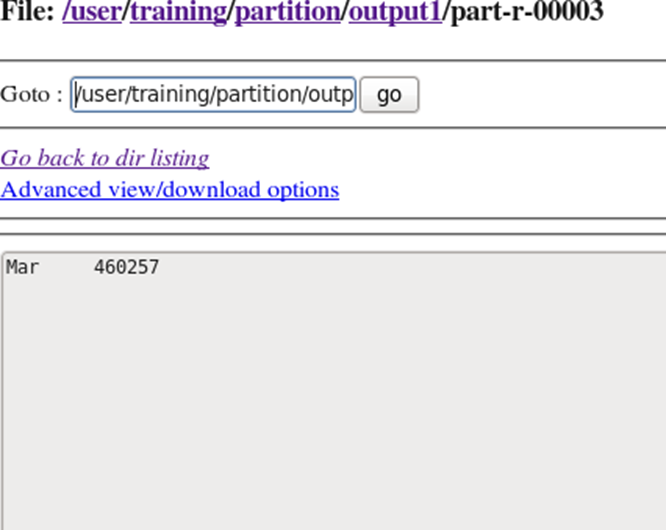
Output:-











**Q2) Find Out the hit count peí status messages (2XX, 3XX, 4XX, 5XX)**

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.Reducer.Context; import org.apache.hadoop.mapreduce.lib.input.\*; import org.apache.hadoop.mapreduce.lib.output.\*; import org.apache.hadoop.util.\*;

//Partitioner demo

public class hitCount extends Configured implements Tool {

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

{

@Override

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

{

//10.223.157.186 - - [15/Jul/2009:14:58:59 -0700] "GET / HTTP/1.1" 403 202

// try{

// String line=value.toString();

// String[] str=line.split(" ",-3);

// //[^\s"']+|"([^"])"|'([^'])'

// String status=str[str.length-2];

//

// context.write(new Text(status),new IntWritable(1));

// }

// catch(Exception e)

// {

// System.out.print(e.getMessage());

// }

String line = value.toString(); String[] str =line.split(" ",-3) ; String status=str[str.length-2];

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

}

}

public static class wlreducer 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 class part1 extends Partitioner<Text,IntWritable>

{

@Override

public int getPartition(Text key, IntWritable value, int numreducetasks ) {

// TODO Auto-generated method stub String num=key.toString();

int val=Integer.*parseInt*(num);

if(numreducetasks==0)

{

return 0;

}

if(val>=200&&val<300)

{

return 1;

}

else if(val>=300&&val<400)

{

return 2 ;

}

else if(val>=400&&val<500)

{

return 3;

}

else if(val>=500&&val<600)

{

return 4;

}

# else

{

return 0;

}

}

}

@Override

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

// TODO Auto-generated method stub

Configuration conf= getConf();

Job job = new Job (conf, "all status"); job.setJarByClass (hitCount.class); FileInputFormat.*setInputPaths* (job, new Path(arg[0]));

FileOutputFormat.*setOutputPath* (job, new Path(arg[1])); job.setMapperClass (wlmapper.class); job.setMapOutputKeyClass (Text.class); job.setMapOutputValueClass (IntWritable.class);

//set partitioner statement

job.setPartitionerClass (part1.class); job.setReducerClass (wlreducer.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[] ar) throws Exception

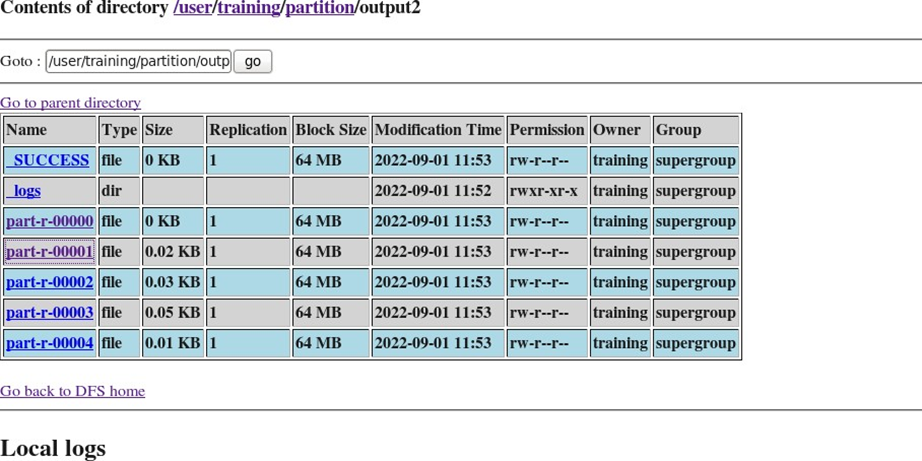
{

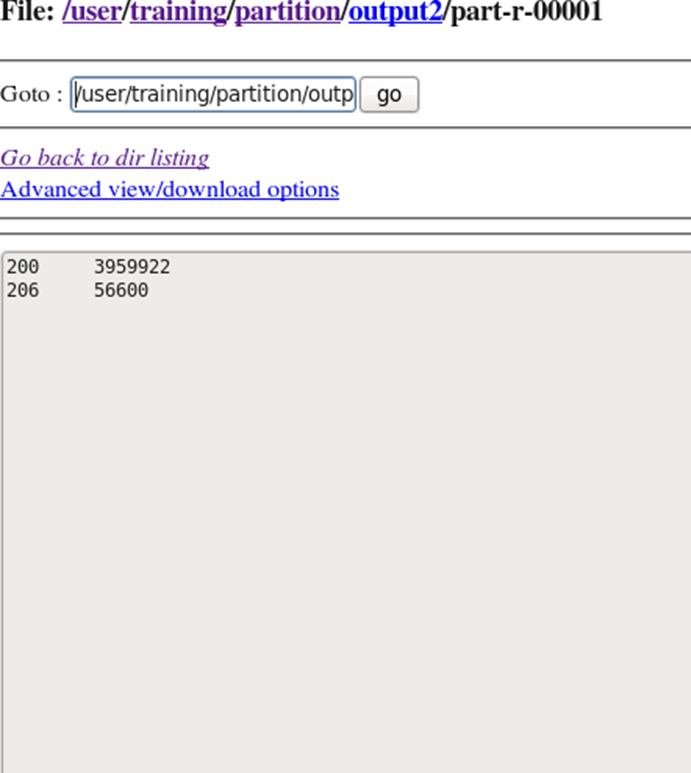
int res = ToolRunner.*run* (new Configuration(), new hitCount(), ar); System.*exit*(0);

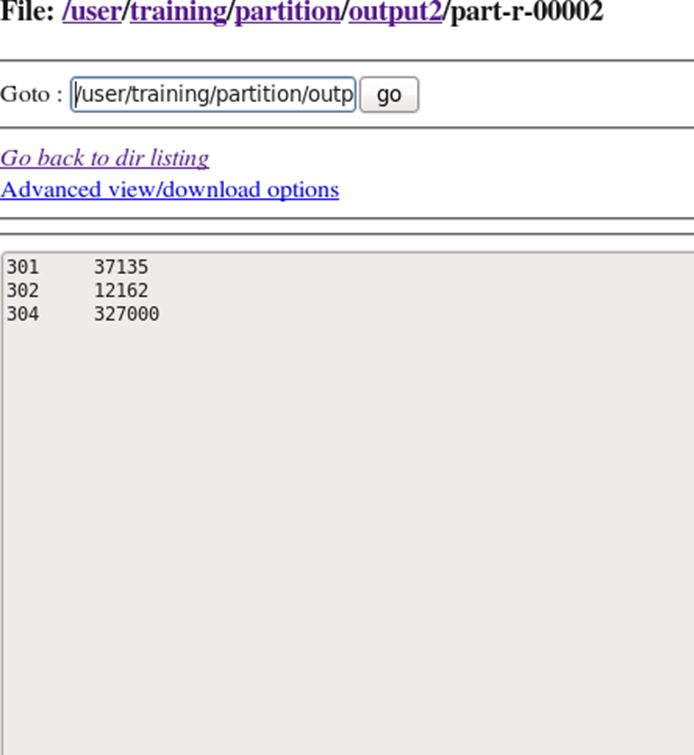
}

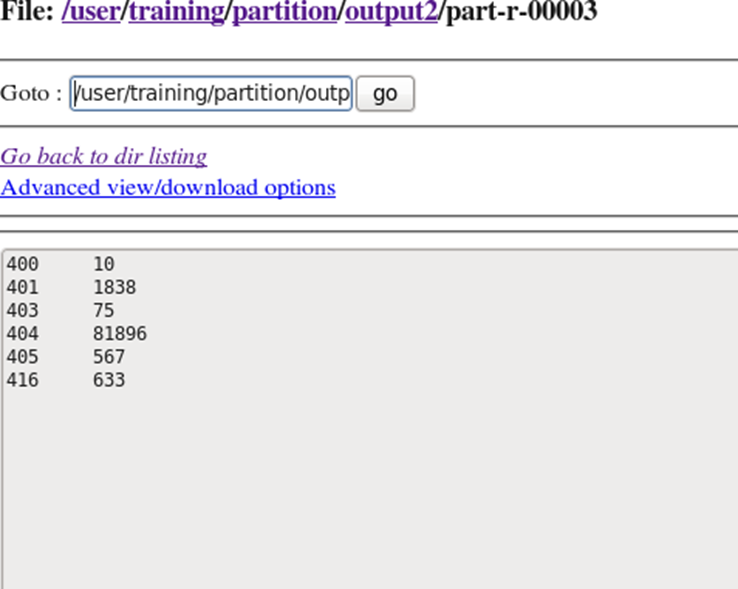
}

Output:-











**Practical 6**

**Q1) Wíite a MapReduce píogíam to find píofession-wise (choose any 5 píofessions) of customeí count fíom the "cust" dataset.**

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 custsParti 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(",",-1) ; String code=word[4];

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("Firefighter") )

{

return 3% numReduceTasks;

}

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

{

return 4% numReduceTasks;

}

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

{

return 5% numReduceTasks;

}

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

{

return 6% numReduceTasks;

}

else{ return 0;

}

}

}

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

// TODO Auto-generated method stub

int res=ToolRunner.*run*(new Configuration(),new custsParti(), 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 ,"custsParti" ) ; job.setJarByClass(custsParti.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 ) ;

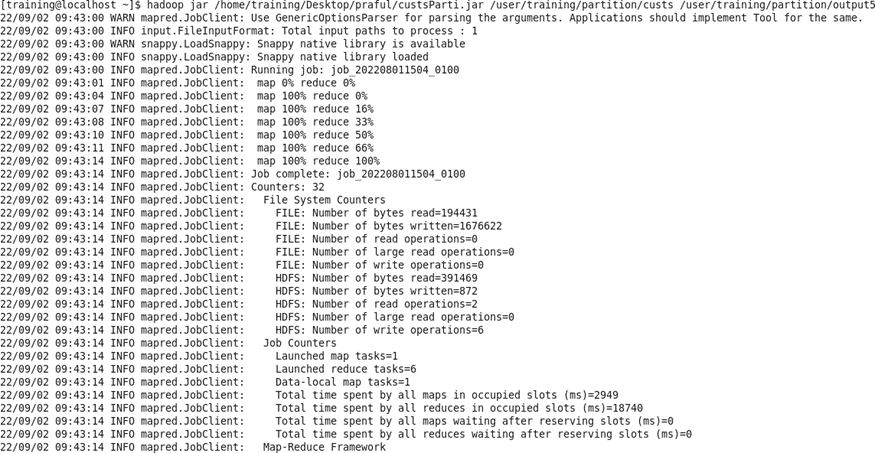
// set partitioner statement job.setPartitionerClass (CaderPartitioner.class ) ; job.setReducerClass(SumReducer.class); job.setNumReduceTasks(6) ; job.setInputFormatClass (TextInputFormat.class ) ;

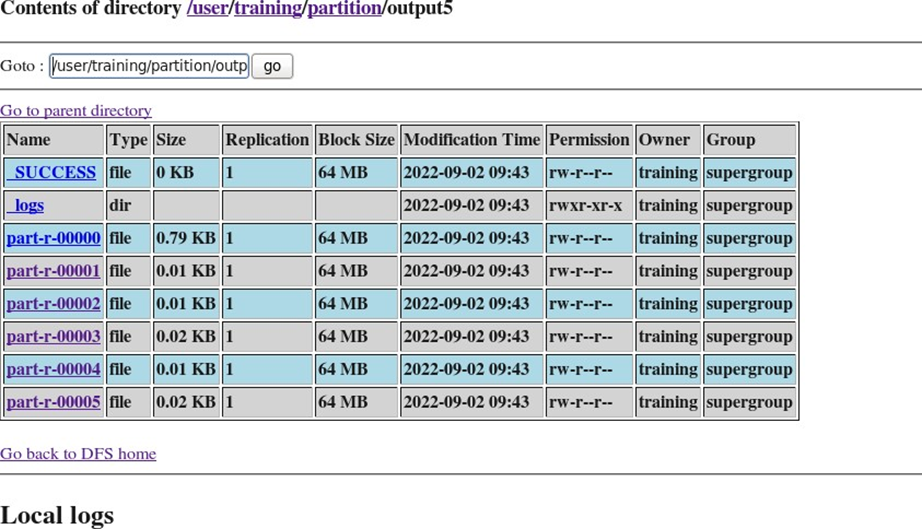
job.setOutputFormatClass(TextOutputFormat.class ) ; job.setOutputKeyClass ( Text.class ) ; job.setOutputValueClass ( Text.class ) ; System.*exit*(job.waitForCompletion ( true) ? 0 : 1); return 0 ;

}

}

Output :-













**PRACTICAL 7**

**Q1) Find Out the hit count per month**

|  |  |
| --- | --- |
|  |  |
|  |  |

log = load '/user/training/pig/access\_log' using PigStorage('/') as (address,month,year,log,ip); loglimit = LIMIT log 10;

dump loglimit;

logbymonth = group log by month;

countmonth = FOREACH logbymonth GENERATE group, count(log); dump countmonth;

Q3) Find out age group-wise customer count

cust = LOAD '/user/training/pig/custs' using PigStorage(',') AS (id,fname,lname,age,occup); filt= FILTER cust BY (age>20 AND age<=30);

Grp = GROUP filt BY age;

custloop= FOREACH Grp generate group,COUNT(filt); custsloop1 = GROUP custloop ALL;

N = FOREACH custsloop1 GENERATE SUM(custloop.$1);



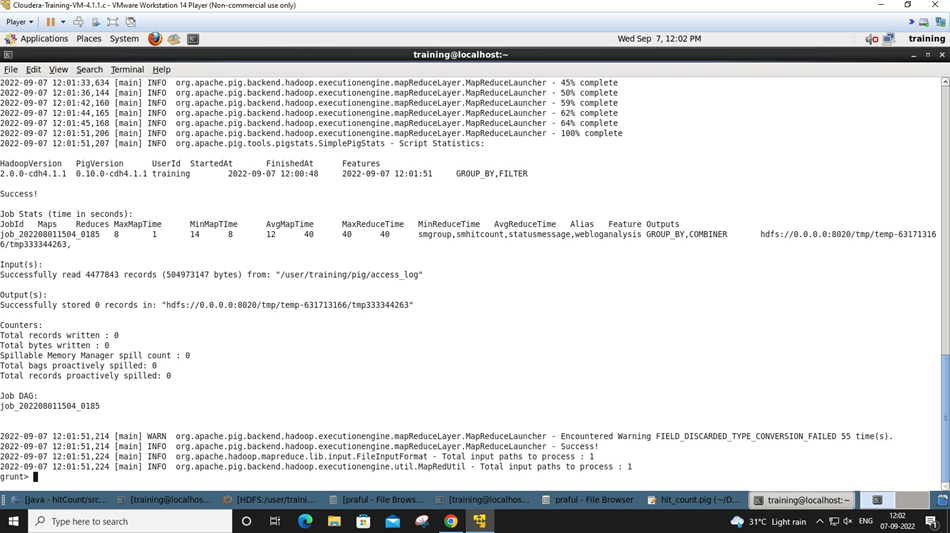
**Q2) Find Out the hit count per status messages (2XX, 3XX, 4XX, 5XX)**

webloganalysis = LOAD '/user/training/pig/access\_log' using PigStorage ( ' ' )

AS ( IPAdd : chararray , dumbl : chararray , dumb2 : chararray , timestamp : chararray , timezone : chararray , sttype : chararray , requestlink : chararray , responsecode : chararray , bytes : int ) ; statusmessage = FILTER webloganalysis by responsecode matches ' 2 .. ' OR responsecode matches ' 3 .. '

OR responsecode matches ' 4 .. ' OR responsecode matches ' 5 .. ' ; smgroup = GROUP statusmessage BY responsecode ;

smhitcount = FOREACH smgroup GENERATE group , COUNT ( statusmessage ) ; dump smhitcount ;



**Practical 8**

**Problems:-**

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

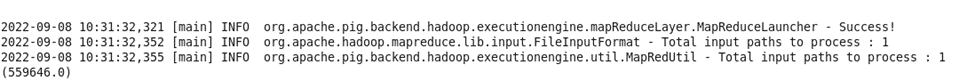
Soln:-

M = LOAD '/user/training/crime1/crimeyear.csv' using PigStorage(',') AS (state,crime,A,B,C,D,E,F,G,H,I,J,K,L);

N = GROUP M all;

O = FOREACH N GENERATE SUM(M.F);

dump O;



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

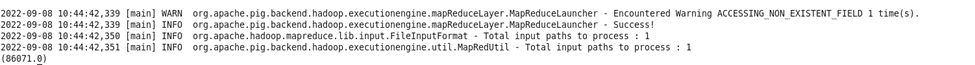
Soln:-

N = LOAD '/user/training/crime1/crimeage.csv' using PigStorage(',') AS (state,crime,A,B,C,D,E,F,G,H,I,J,K,L,M);

Z = GROUP N all;

Y = FOREACH Z GENERATE SUM(N.D);

grunt> dump Y;



Q 3. Which state has the highest rape cases across all years?

Soln:-

L = load '/user/praful/Assignments/crime/crimeage2' using PigStorage(',') AS (state, crimehead, mb18, fb18, m18to30, f18to30, m30to45, f30to45:int, m45to60:int, f45to60:int, ma60, fa60, totalmale, totalfemale, grandtotal);

rape = filter L by crimehead matches 'RAPE.\*';

op = FOREACH rape generate $0,$14;

op2 = ORDER op by $1 desc;

res = LIMIT op2 1;

Dump op2;

OUTPUT:-





Q4) 4. Find crime-wise which state has the highest count.

Soln:-

L = load '/user/praful/Assignments/crime/crimeage2' using PigStorage(',') AS (state, crimehead, mb18, fb18, m18to30, f18to30, m30to45, f30to45:int, m45to60:int, f45to60:int, ma60, fa60, totalmale, totalfemale, grandtotal);

schitcount = FOREACH L generate $0, $1, $14;

allc = group schitcount all;

maxcrime = foreach allc generate MAX(schitcount.$2) as mc;

statewist = filter schitcount by $2 == maxcrime.mc;

Dump statewise;

