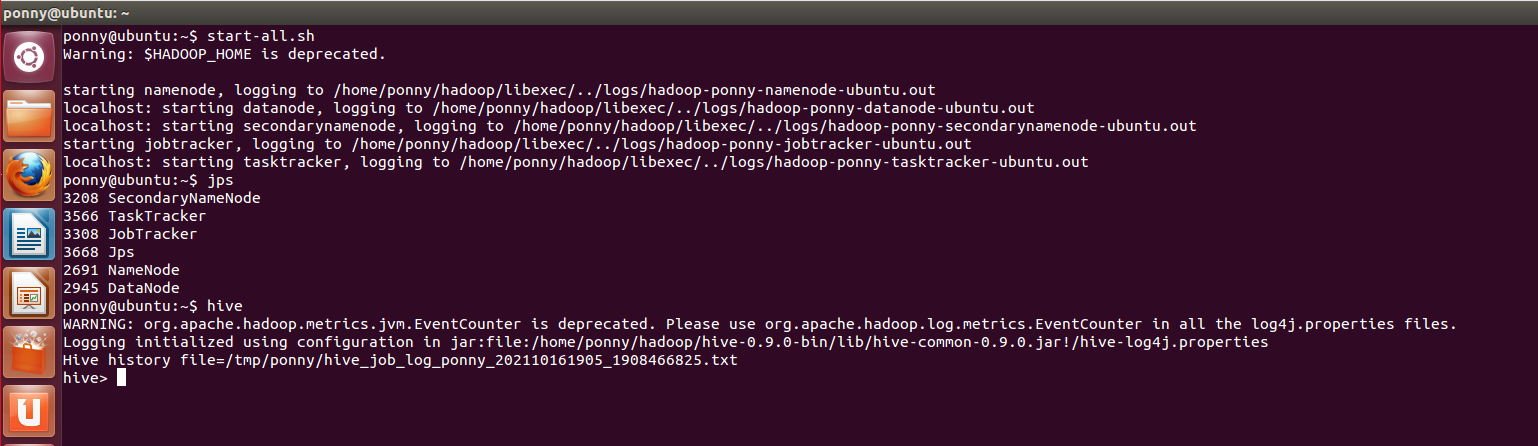
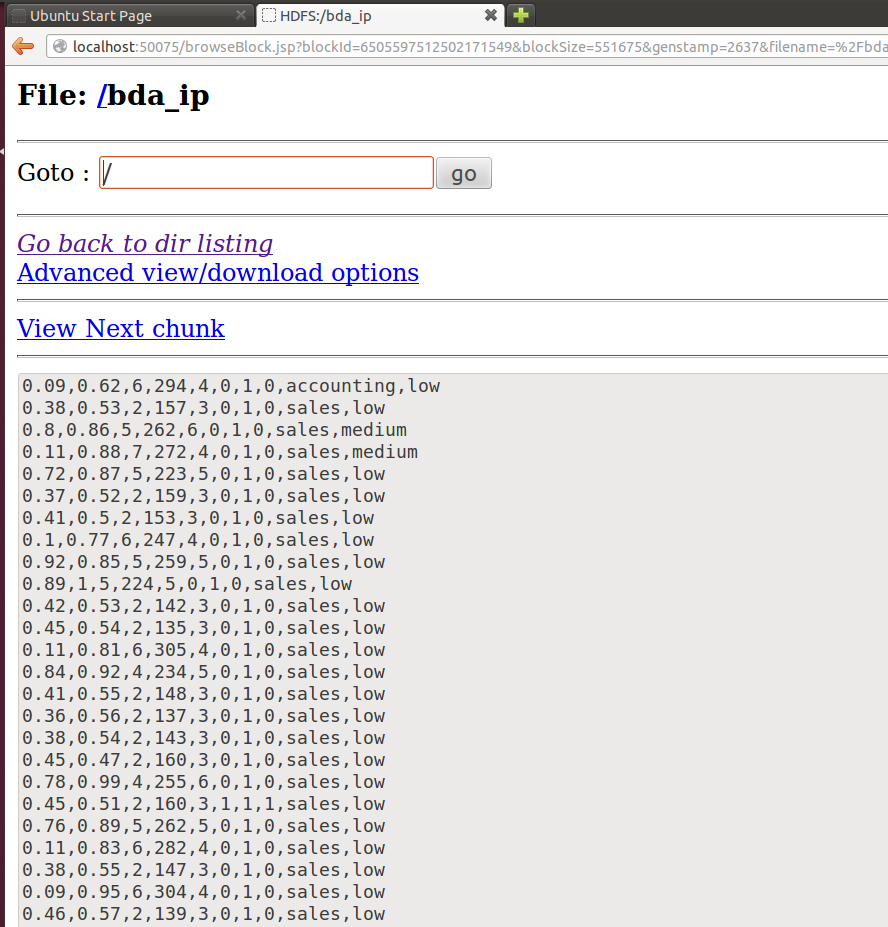
***Hadoop - Employee Turnover Analysis***

**Input:**







**Ques 1) Finding the average number of projects for the departments who left the company.**

**Code:**

package bda;

import java.io.IOException;

import org.apache.hadoop.fs.Path;

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

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

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

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

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

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

public class ques1 {

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

{

IntWritable one = new IntWritable(1);

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

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

if(Integer.parseInt(line[6]) ==1)

{

context.write(new Text(line[8]),new IntWritable(Integer.parseInt(line[2])));

}

}

}

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

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

throws IOException, InterruptedException {

float avg;

int sum = 0;

int count=0;

for (IntWritable val : values) {

count=count+1;

sum += val.get();

}

avg=sum/count;

context.write(key, new FloatWritable(avg));

}

}

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

Configuration conf = new Configuration();

Job job = new Job(conf, "question1");

job.setJarByClass(ques1.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

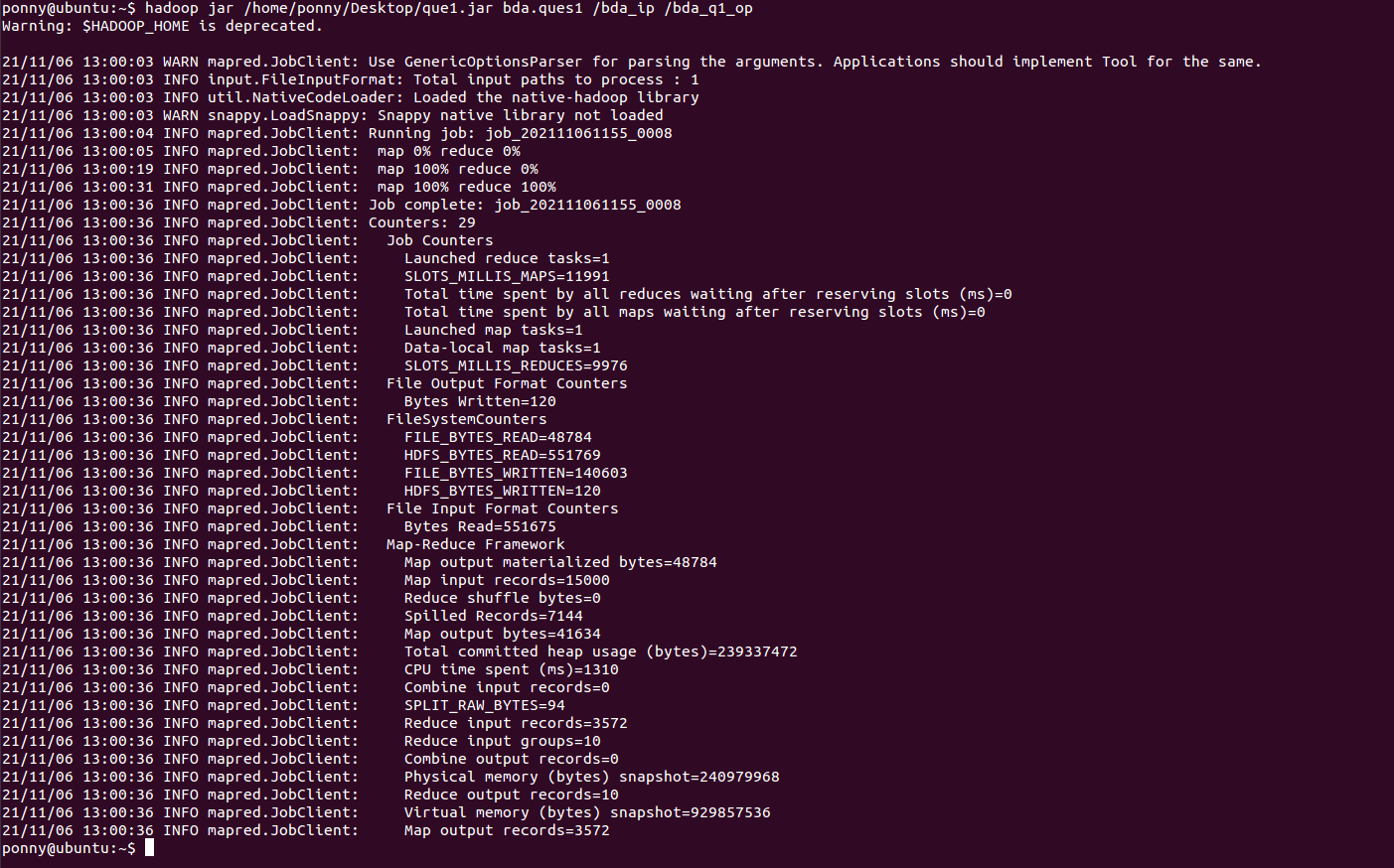
FileOutputFormat.setOutputPath(job, new Path(args[1]));

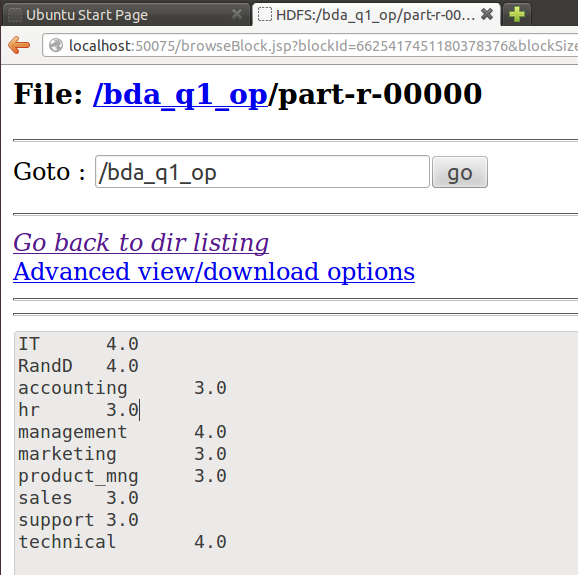
job.waitForCompletion(true);

}

}

**Output:**





**Ques 2)** **Finding the details of the employees with the count who left the company(left==1) with the satisfaction level > 0.5.**

**Code:**

package bda;

import java.io.IOException;

//import java.util.\*;

import org.apache.hadoop.fs.Path;

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

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

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

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

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

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

public class ques2 {

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

{

IntWritable one = new IntWritable(1);

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

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

if (Float.parseFloat(line[0])>0.5 && Integer.parseInt(line[6])==1){

context.write(new Text(line[8]),new Text(line[0]+','+line[2]));

}

}

}

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

int sum = 0;

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

for (Text val : values) {

context.write(key,val);

sum++;

}

}

public void cleanup(Context context) throws IOException, InterruptedException

{

context.write(new Text("Total NUmber of Records: "),new Text(Integer.toString(sum)));

}

}

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

Configuration conf = new Configuration();

Job job = new Job(conf, "question2");

job.setJarByClass(ques2.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class); job.setOutputFormatClass(TextOutputFormat.class);

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

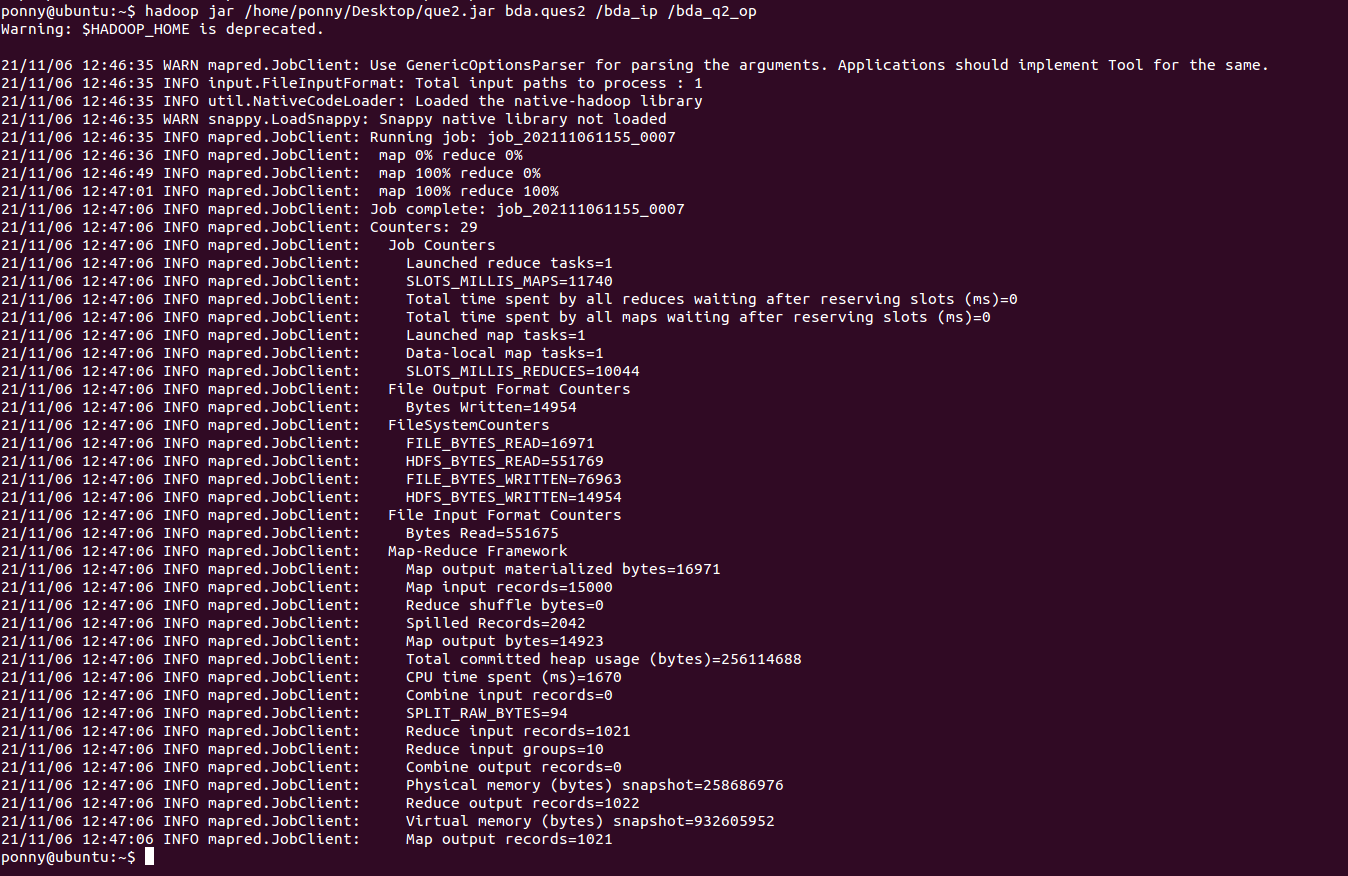
FileOutputFormat.setOutputPath(job, new Path(args[1]));

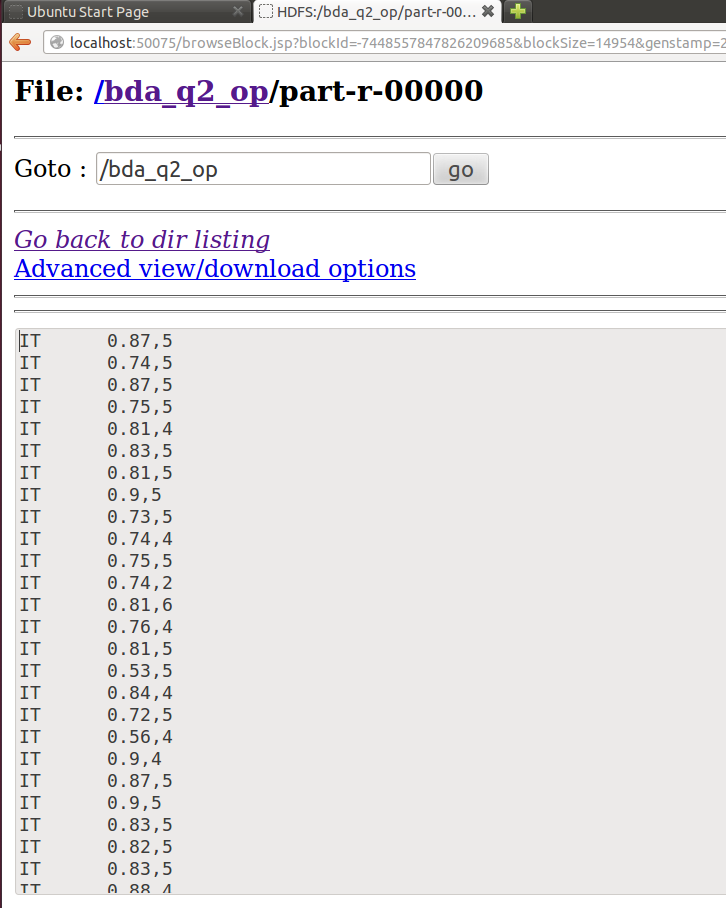
job.waitForCompletion(true);

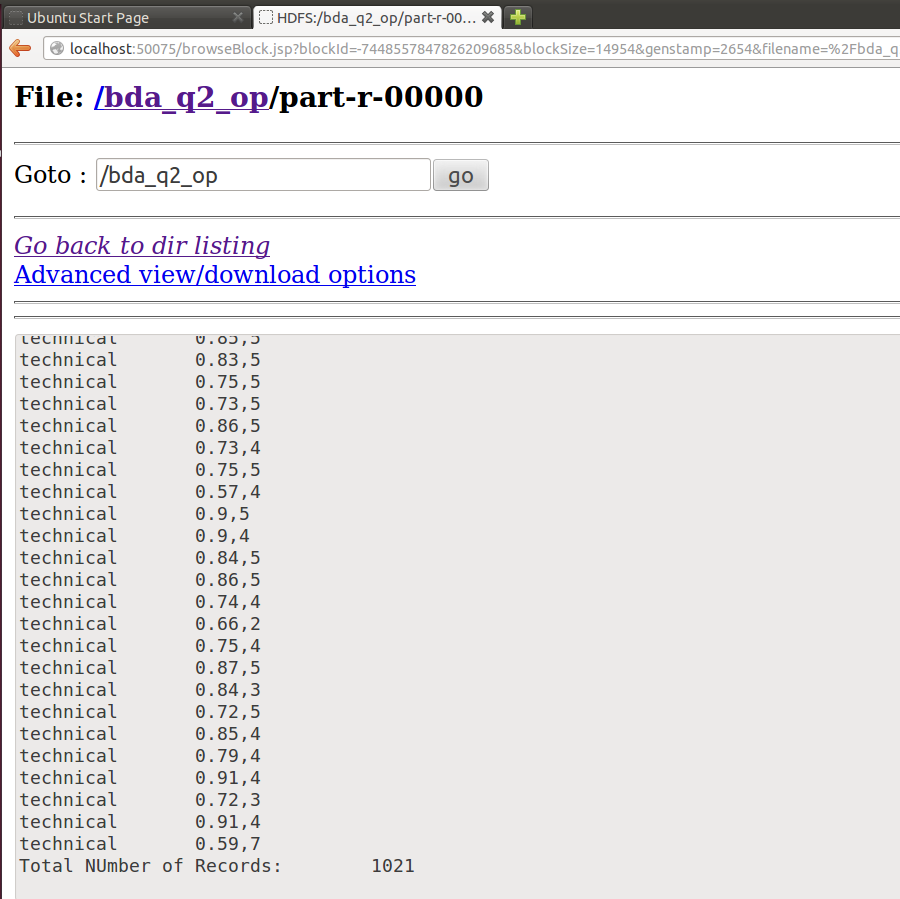
}

}

**Output:**







**Ques 3) Using NLineinput Format for our dataset and dividing into 5000 lines in each mapper.**

**Code:**

package bda;

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

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

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

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

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

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

public class ques3

{

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

{

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

{

context.write(key, value);

}

}

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

{

Configuration conf = new Configuration();

conf.setInt(NLineInputFormat.LINES\_PER\_MAP,5000);

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

job.setJarByClass(ques3.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(Map.class);

job.setInputFormatClass(NLineInputFormat.class);

job.setNumReduceTasks(0);

job.setOutputFormatClass(TextOutputFormat.class);

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

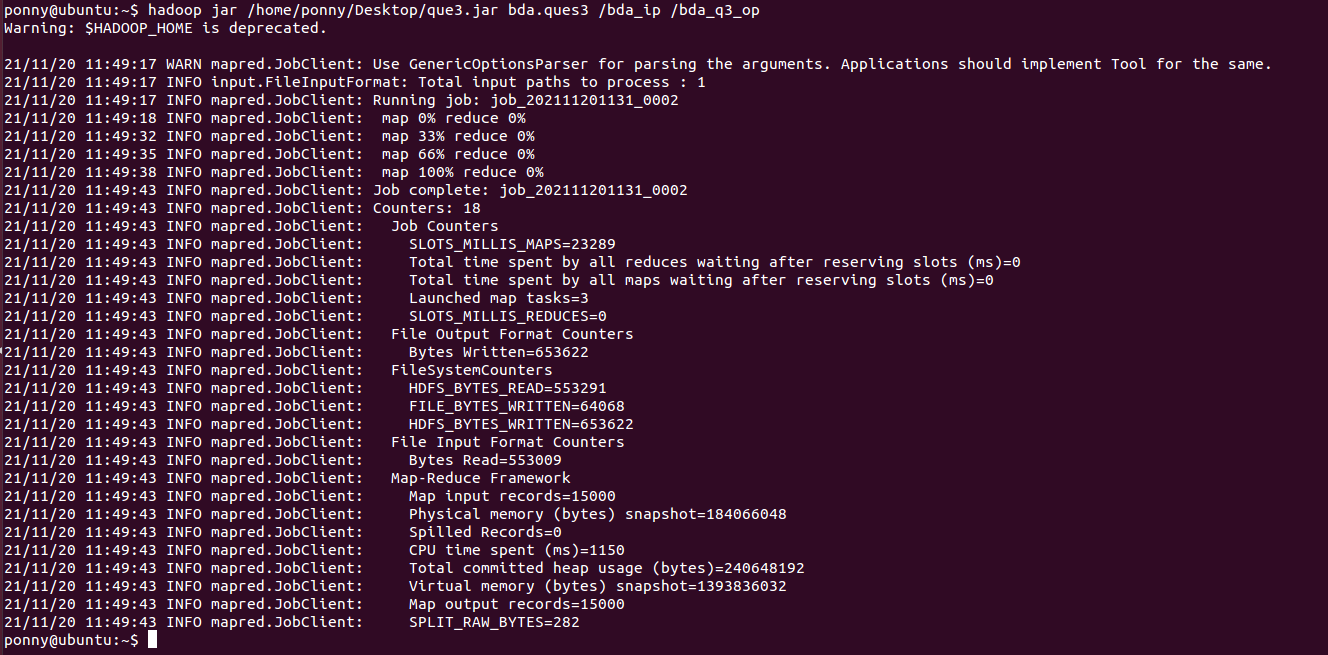
FileOutputFormat.setOutputPath(job, new Path(args[1]));

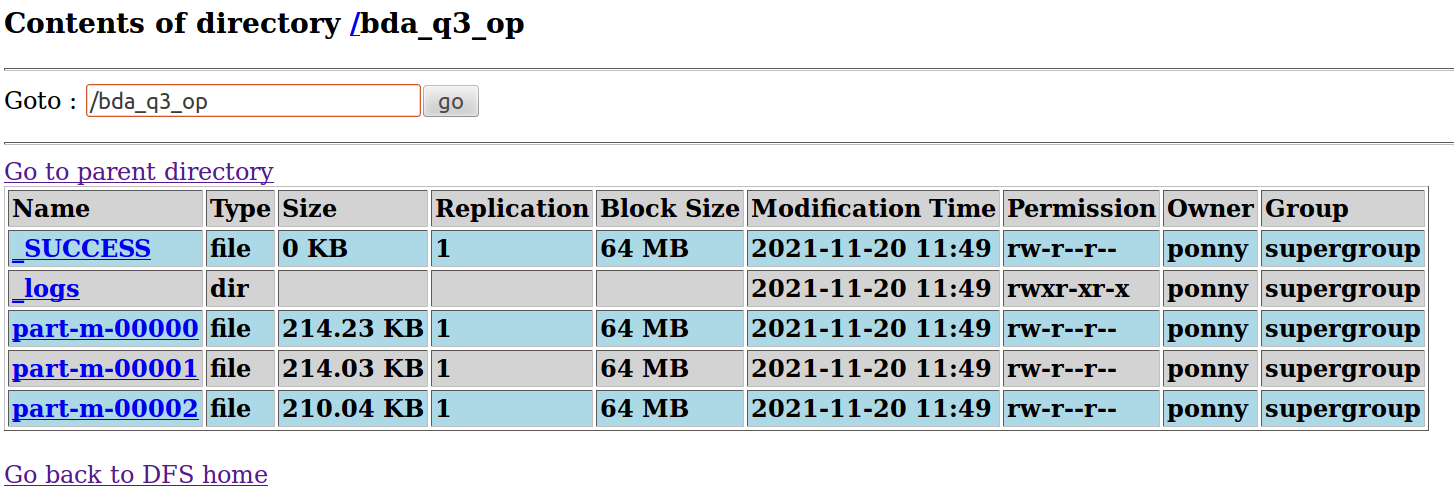
job.waitForCompletion(true);

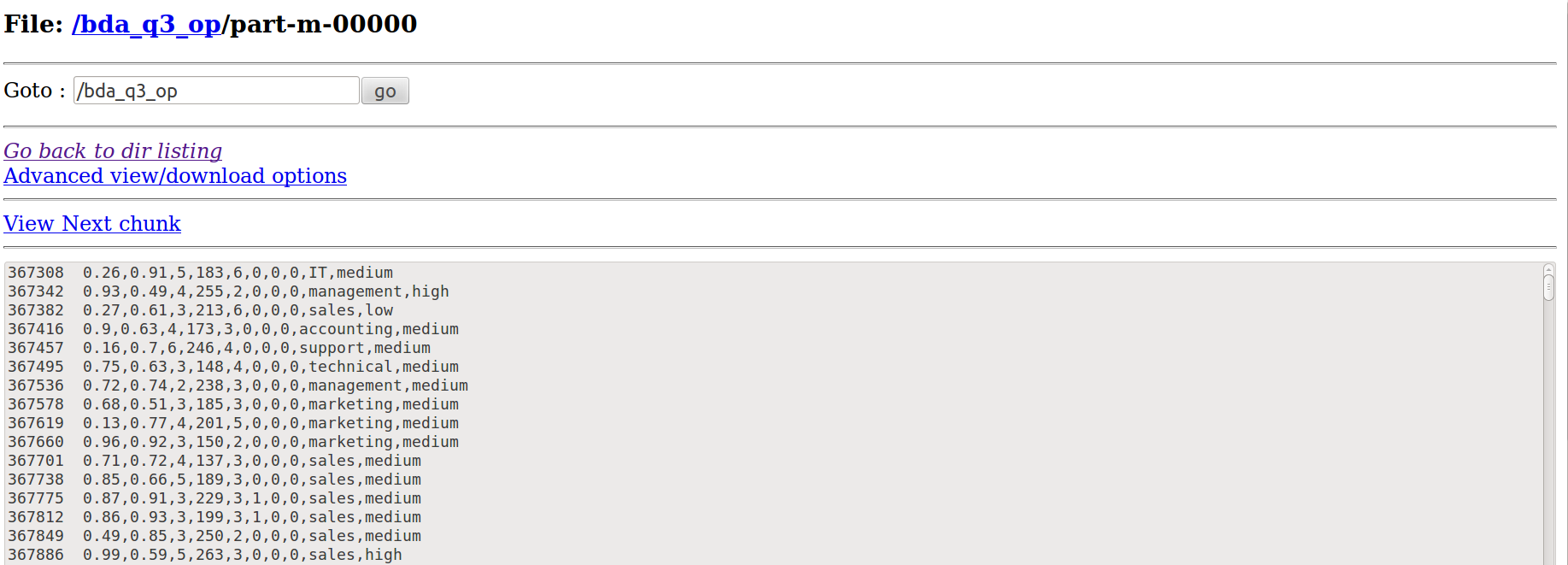
}

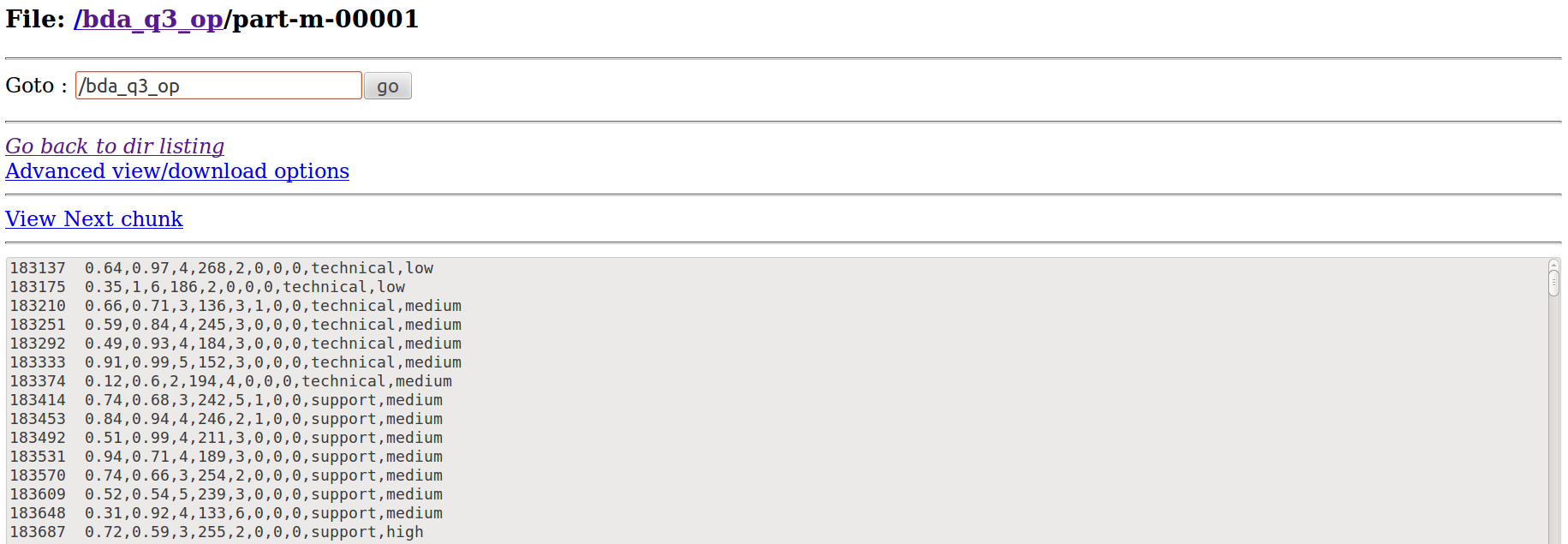
}

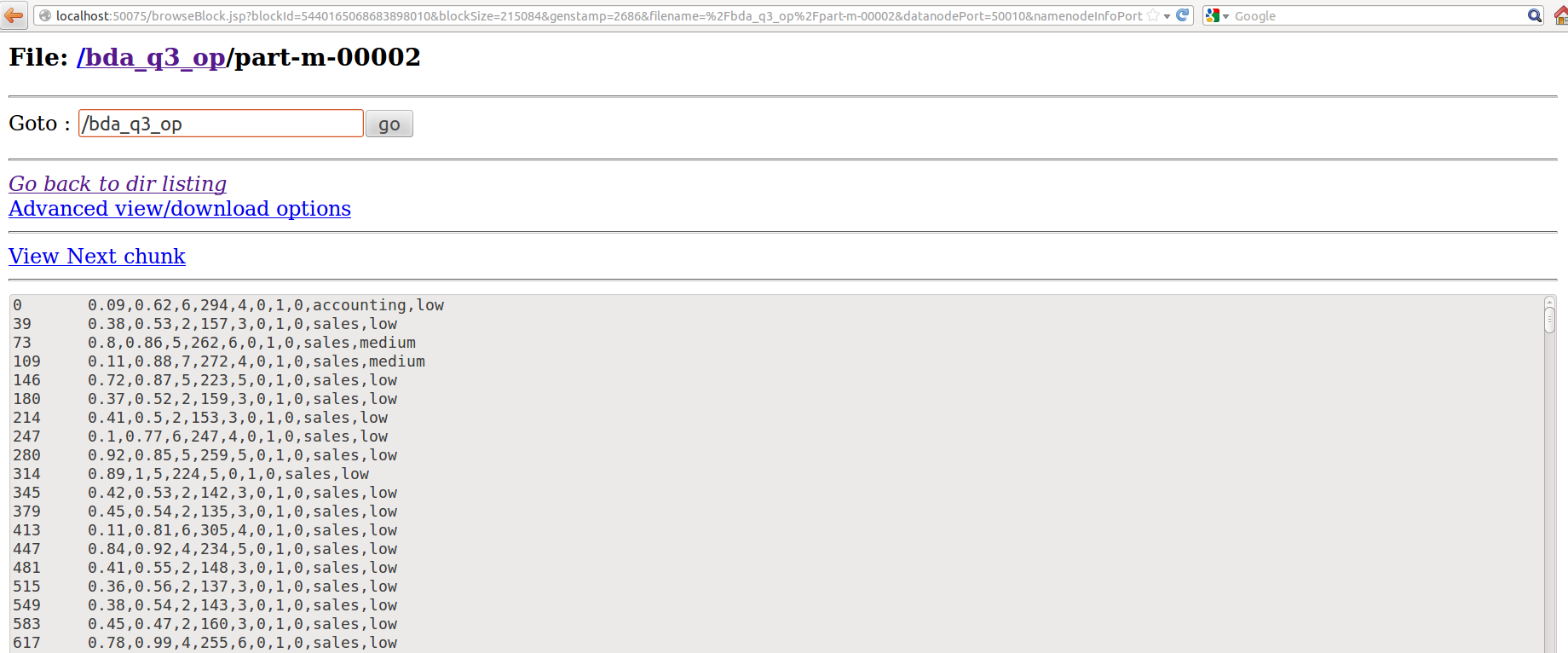
**Output:**











**Ques 4) Partitioning our dataset based on the left attribute. Creating 2 partitions with left = 1(who left the company) and left = 0 and writing the salary details along with the left attribute value. Also displaying the count of the records satisfying the above conditions.**

**Code:**

package bda;

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

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

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

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

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

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

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

public class ques4{

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

{

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

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

context.write(new Text(line[9]),new IntWritable(Integer.parseInt(line[6])));

}

}

public static class lpart extends Partitioner<Text,IntWritable>

{

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

{

if(value.get()== 1)

return 0;

else

return 1;

}

}

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

int sum = 0;

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

throws IOException, InterruptedException {

for (IntWritable val : values) {

sum=sum+1;

context.write(key, val);

}

}

public void cleanup(Context context)throws IOException,InterruptedException

{

context.write(new Text("Total:"),new IntWritable(sum));

}

}

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

Configuration conf = new Configuration();

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

job.setPartitionerClass(lpart.class);

job.setNumReduceTasks(2);

job.setJarByClass(ques4.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

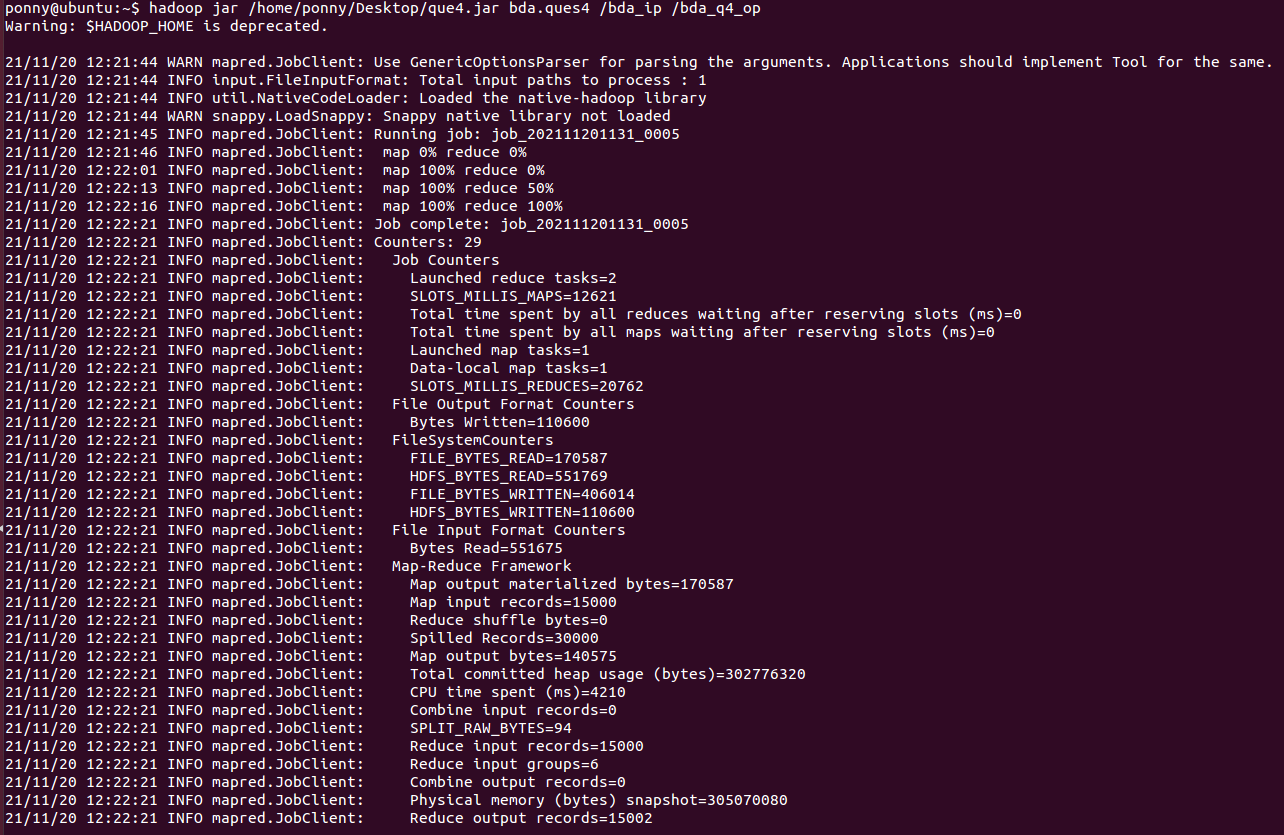
FileOutputFormat.setOutputPath(job, new Path(args[1]));

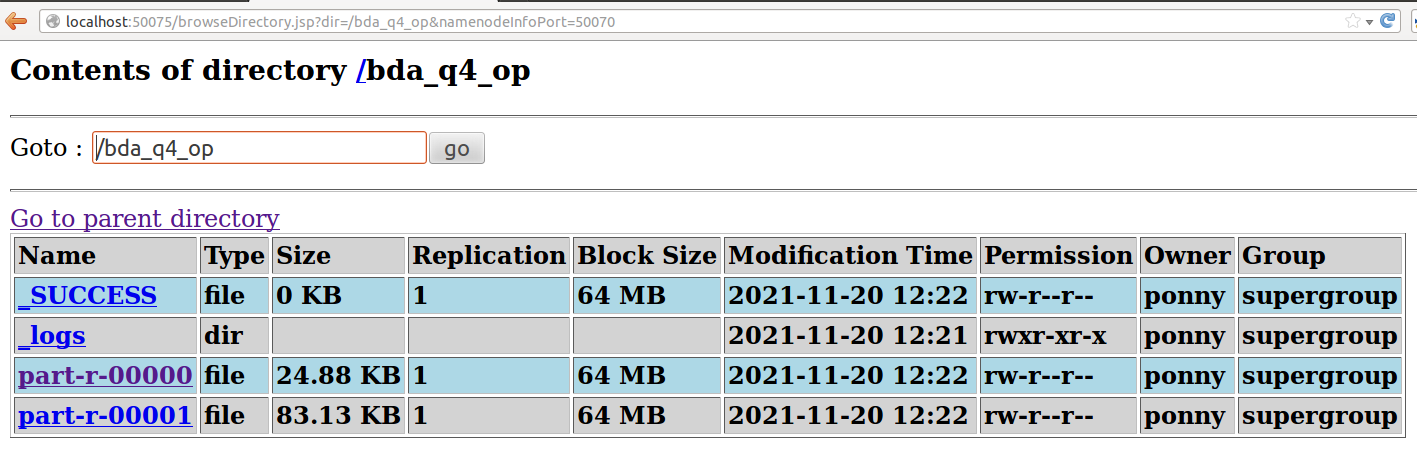
job.waitForCompletion(true);

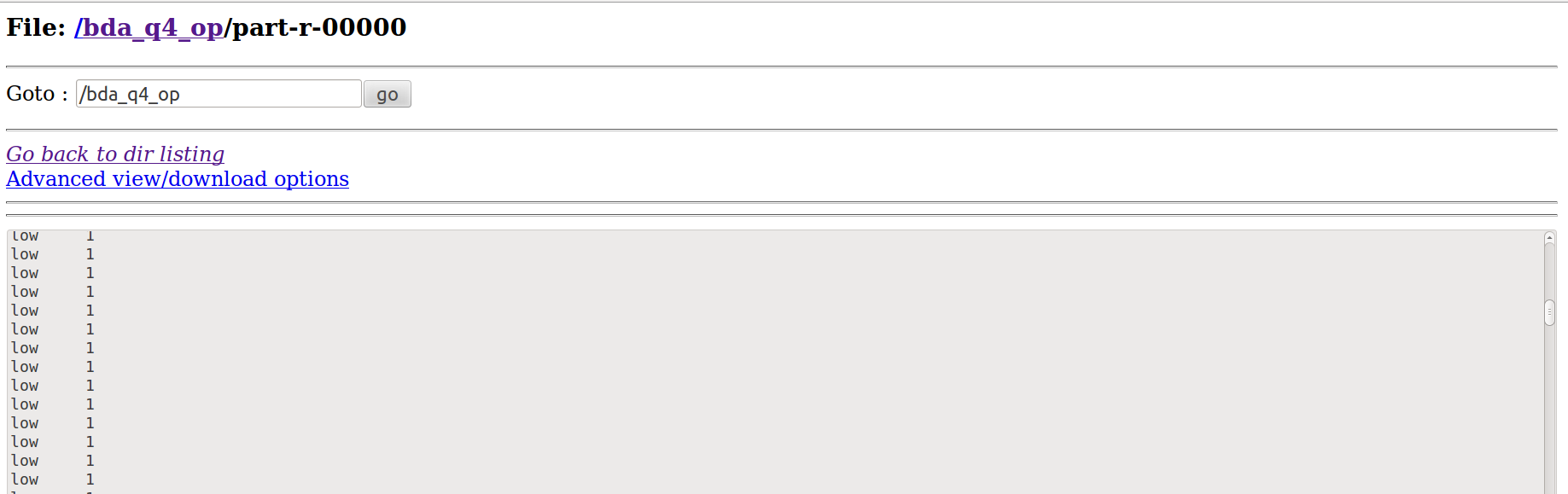
}

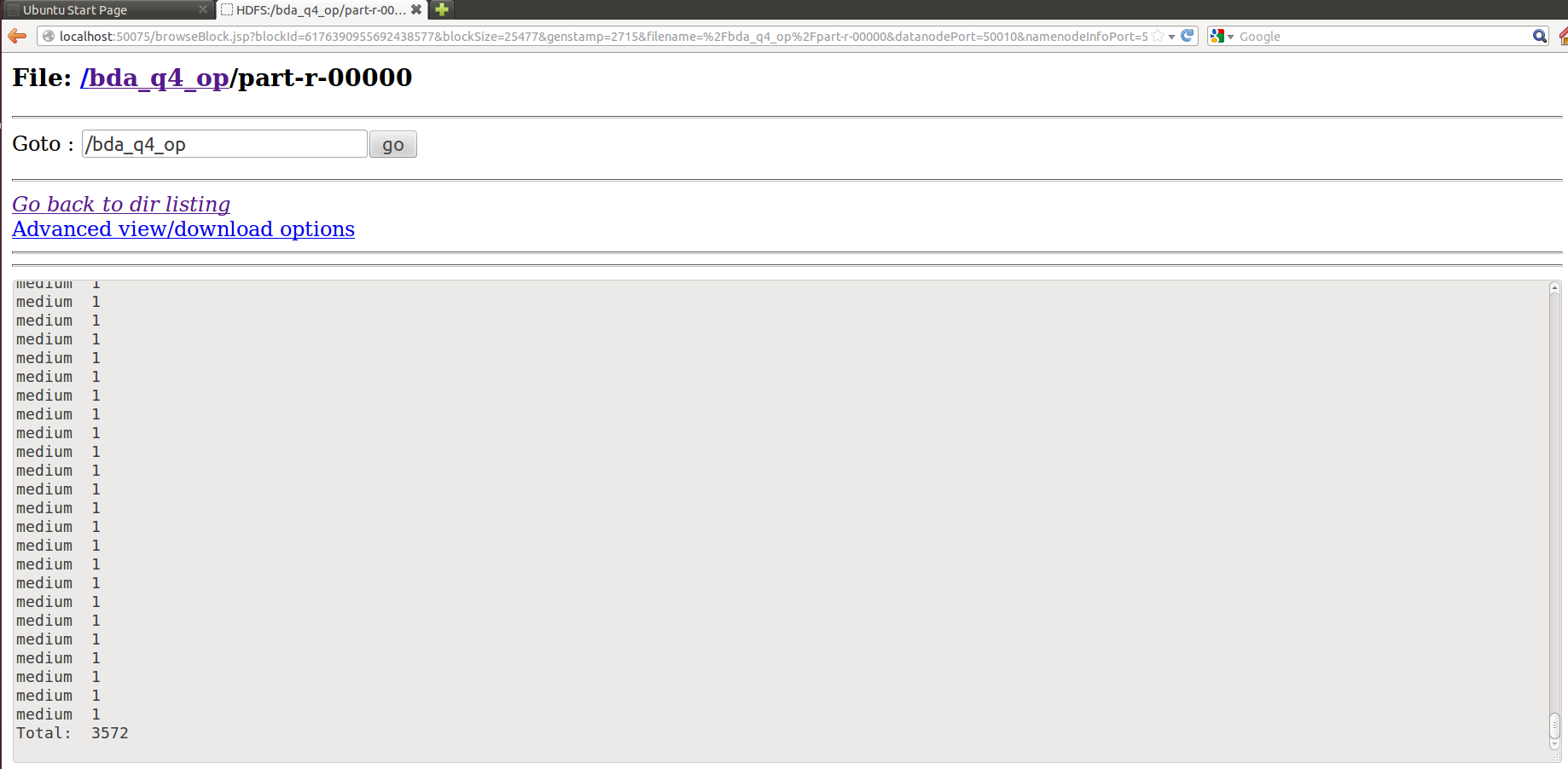
}

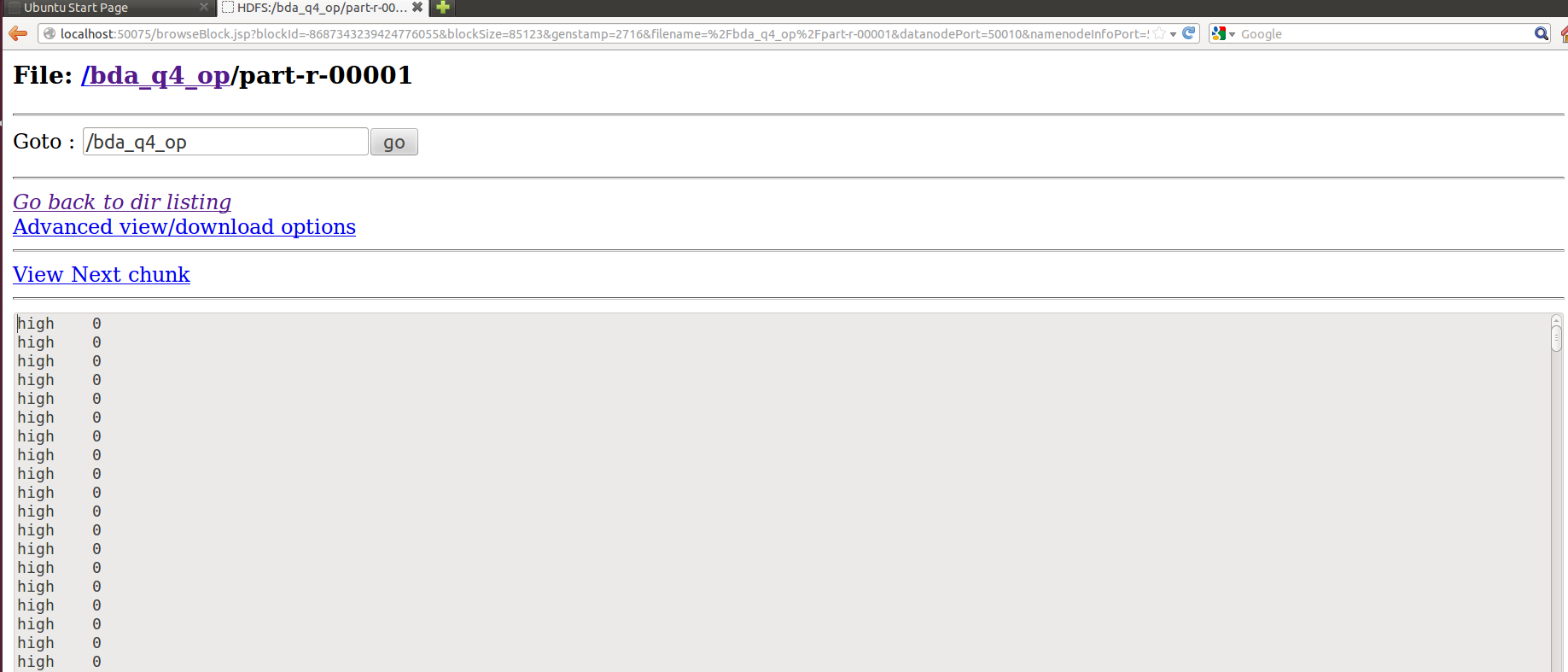
**Output:**











**Ques 5) Using the *user defined counters* function, we printed the details of the employees whose satisfaction level is greater than 0.5 and number of projects greater than 5 along with the left attribute.**

**Code:**

package bda;

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

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

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

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

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

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

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

public class ques5

{

public enum ct

{

cnt ;

};

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

{

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

{

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

if(Float.parseFloat(line[0])>0.5 && Integer.parseInt(line[2])>5)

{

context.getCounter(ct.cnt).increment(1);

context.write(new Text(line[2]),new LongWritable(Long.parseLong(line[6])));

}

}

}

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

{

public void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOException, InterruptedException

{

for (LongWritable val : values)

{

context.write(key, val);

}

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(ques5.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

job.waitForCompletion(true);

Counters cn=job.getCounters();

cn.findCounter(ct.cnt).getValue();

}

}

**Output:**

