**EAST WEST UNIVERSITY**

# Big Data Analysis

**Course Code:** CSE **488**

**Section**: **01**

**All Labs**

# **SUBMITTED TO:**

**Name:** Mohammad Rezwanul Huq

# **SUBMITTED BY:**

**Name:** MD. WAHIDUZZAMAN

**ID No:** **2019-2-60-048**

**Lab: 01**

All the quires from lab 1:

bombing = spark.read.json("Bombing\_Operations.json.gz")

aircraft = spark.read.json("Aircraft\_Glossary.json.gz")

Question 1:

bombing.createOrReplaceTempView("bombing")

get = """ select AirCraft,ContryFlyingMission,TargetCountry from bombing where TimeOnTarget < '1000' """

run = spark.sql(get)

run.show()

Question 2:

get = """ select TargetCountry,TimeOnTarget from bombing order by TimeOnTarget desc """

run = spark.sql(get)

run.show()

Question 3:

get = """ select TargetCountry from bombing where TimeOnTarget = ( select MAX(TimeOnTarget) from bombing ) """

run = spark.sql(get)

run.show()

Question 4:

get = """ select ContryFlyingMission from bombing where TimeOnTarget = ( select MAX(TimeOnTarget) from bombing ) """

run = spark.sql(get)

run.show()

Question 5:

get = """ select TakeoffLocation, COUNT(TakeoffLocation) from bombing group by TakeoffLocation having count(TakeoffLocation) """

run = spark.sql(get)

run.show()

Question 6:

aircraft.createOrReplaceTempView("aircraft")

get = """ select COUNT(aircraft) from aircraft """

run = spark.sql(get)

run.show()

Question 7:

aircraft.createOrReplaceTempView("aircraft")

get = """ select aircraft, COUNT(aircraft) from aircraft group by aircraft having count(aircraft) """

run = spark.sql(get)

run.show()

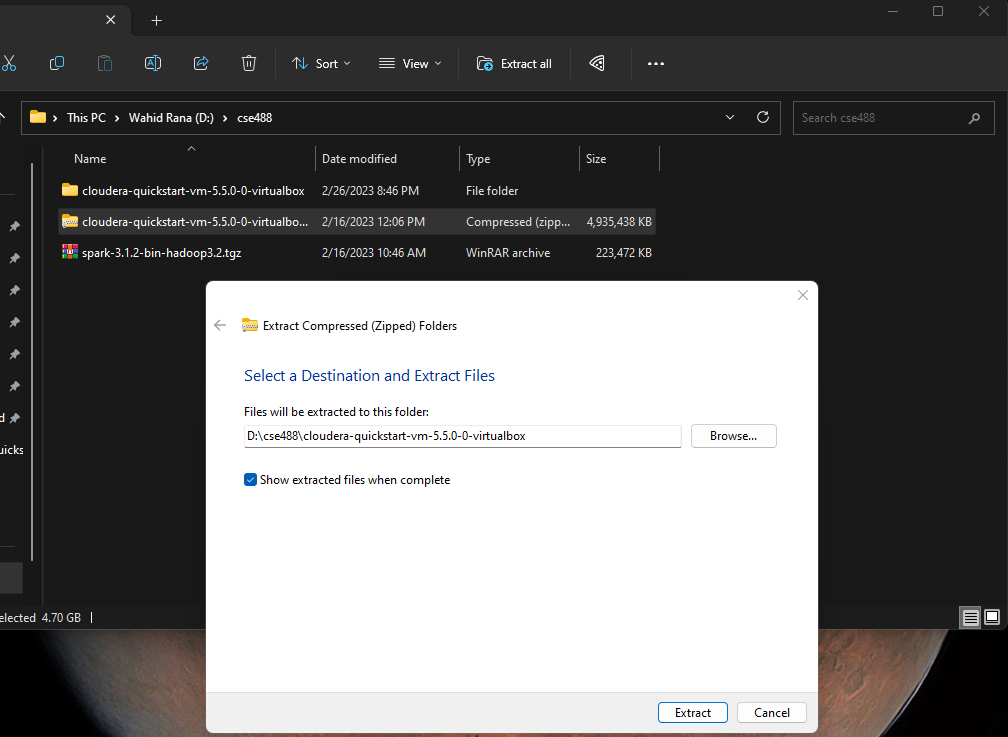
Question 8:

get = """ select MAX(aircraft) from aircraft """

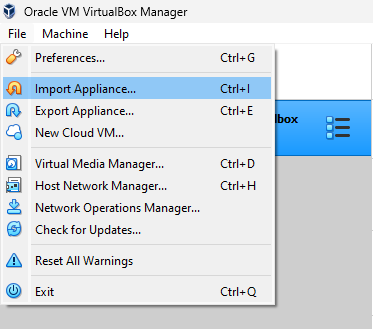
run = spark.sql(get)

run.show()

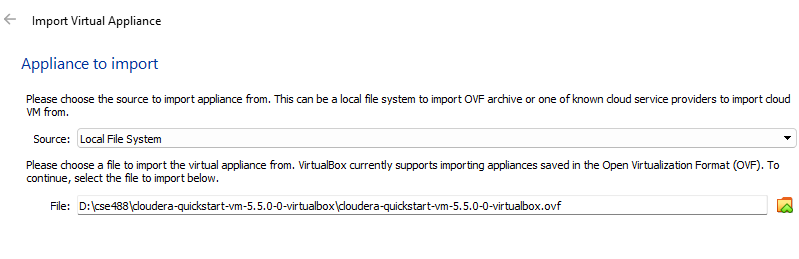
**Lab: 02**



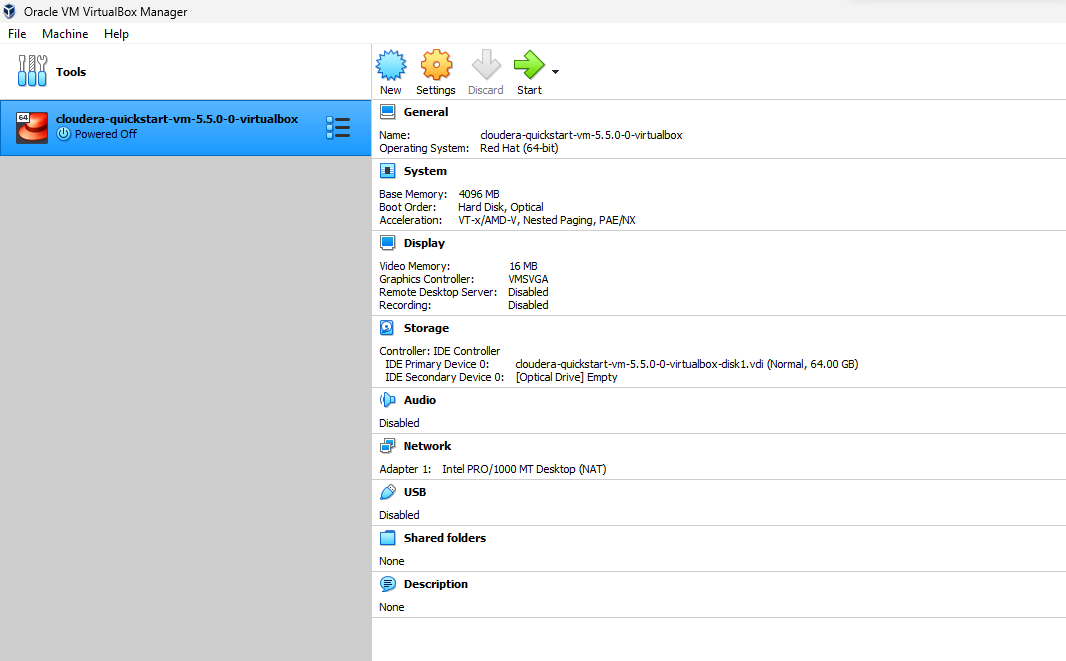
**Figure 1:** Extracting File



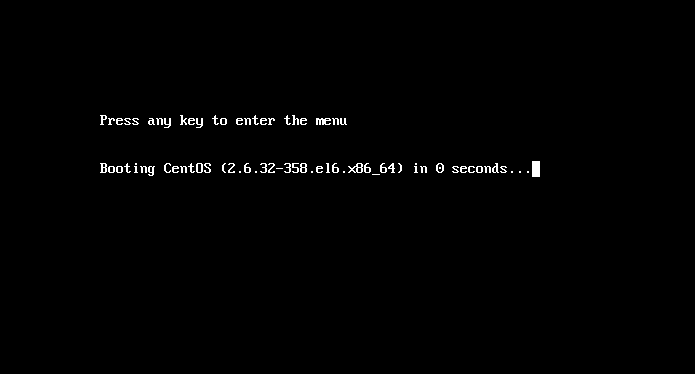
**Figure 2:** Importing OVF file



**Figure 3:** Locating the file

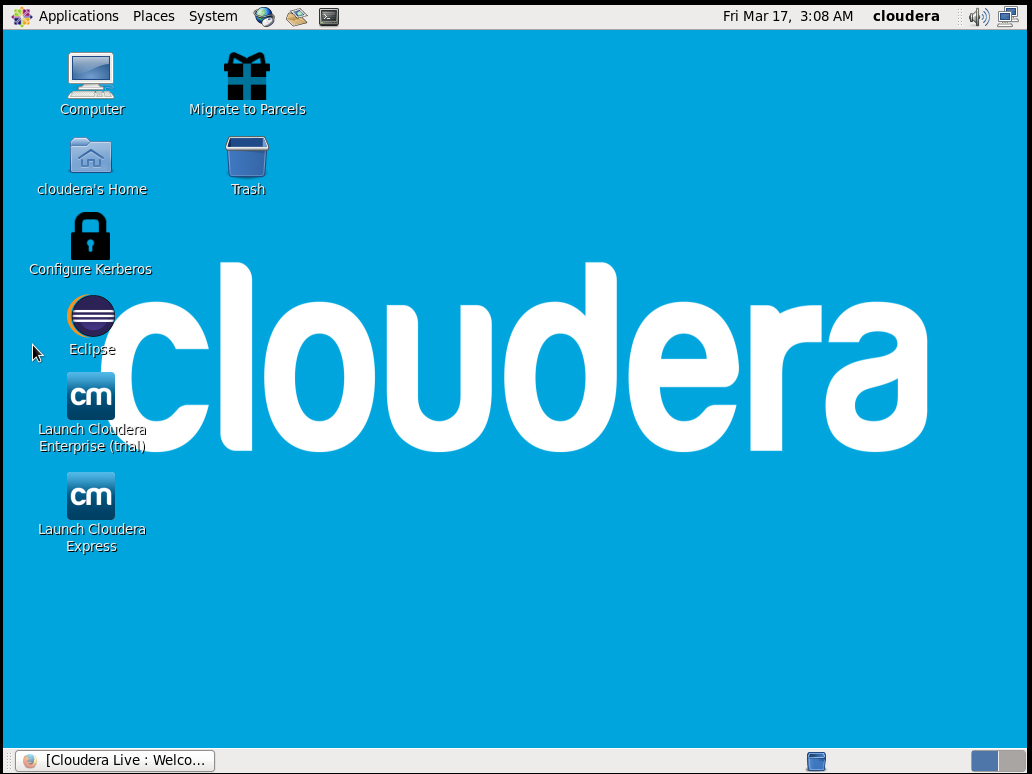


**Figure 4:** Successfully imported



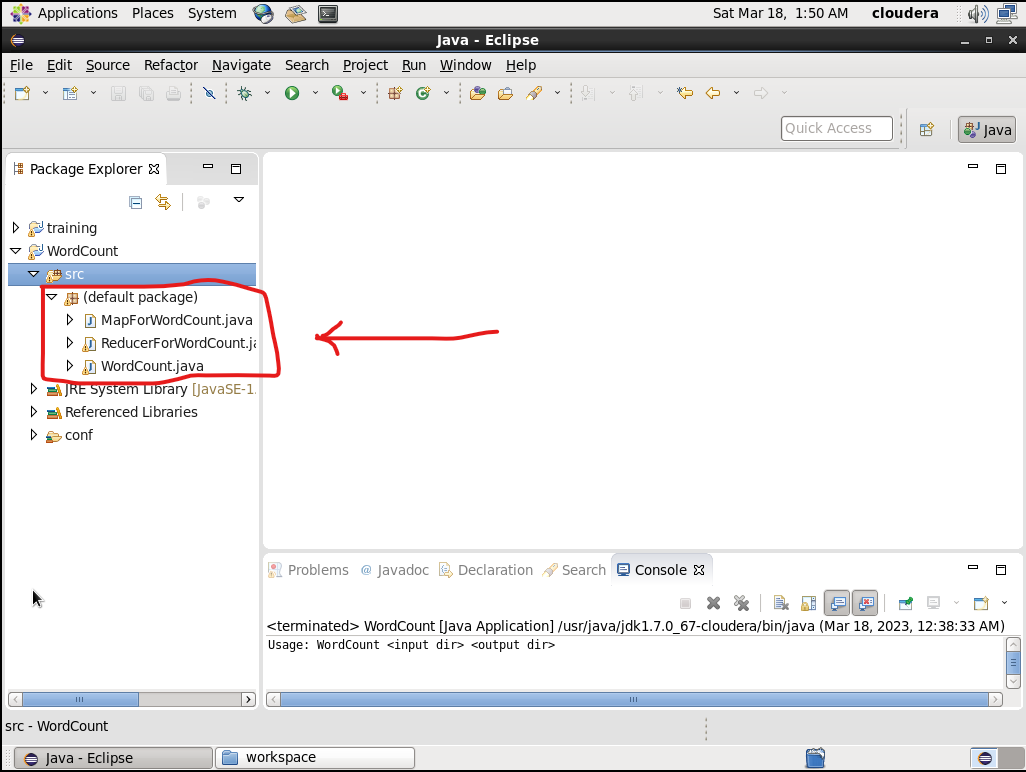


**Figure 5:** Starting the machine



**Figure 6:** Machine successfully started

**Creating Hadoop project in Eclipse:**

****

**Figure 7:** Creating project

**Implanting the given codes in the file**

**WordCount.java**

import org.apache.hadoop.mapreduce.Job;

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

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

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

public class 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);

}

Configuration config = new Configuration();

Path input = new Path(args[0]);

Path output = new Path(args[1]);

@SuppressWarnings("deprecation")

Job job = new Job();

job.setJarByClass(WordCount.class);

job.setJarByClass(WordCount.class);

job.setMapperClass(MapForWordCount.class);

job.setReducerClass(ReducerForWordCount.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, input);

FileOutputFormat.setOutputPath(job,output);

boolean success = job.waitForCompletion(true);

System.exit(success ? 0 : 1);

}

}

**MapForWordCount.java**

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable> {

@Override

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

throws IOException, InterruptedException {

String line = value.toString();

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

for (String word: words){

Text outputKey = new Text(word.toUpperCase().trim());

IntWritable outputValue = new IntWritable(1);

context.write(outputKey, outputValue);

}

}

}

**ReducerForWordCount.java**

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class ReducerForWordCount extends Reducer<Text, IntWritable, Text, IntWritable> {

@Override

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

throws IOException, InterruptedException {

int sum = 0;

for(IntWritable value: values){

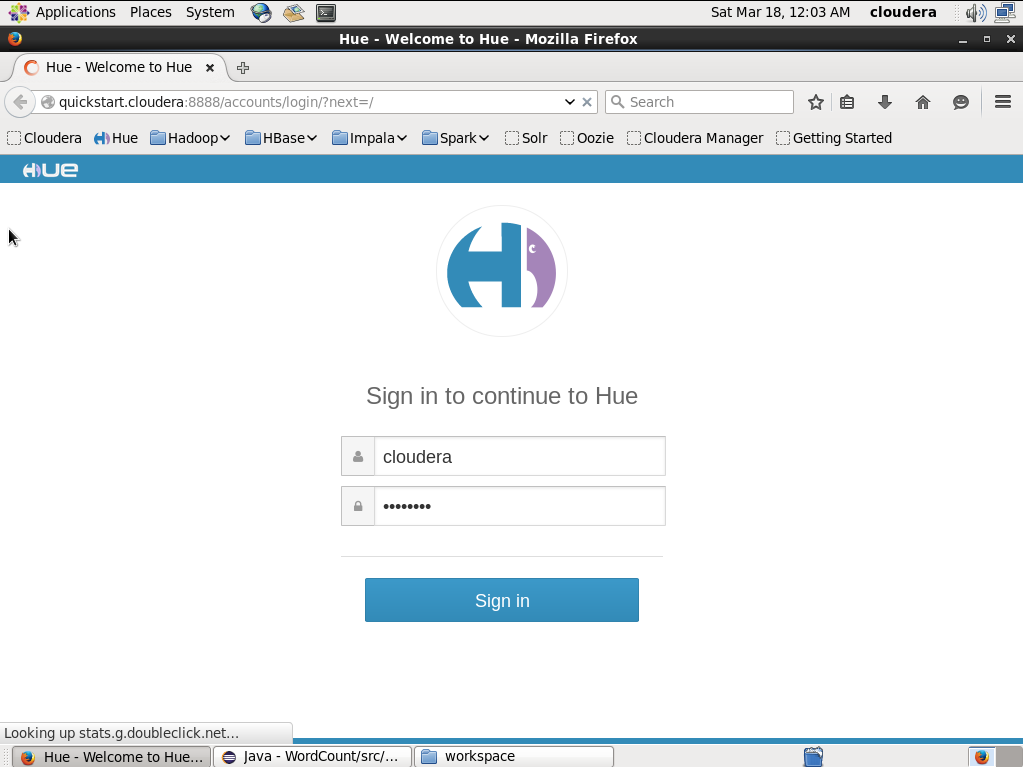
sum += value.get();

}

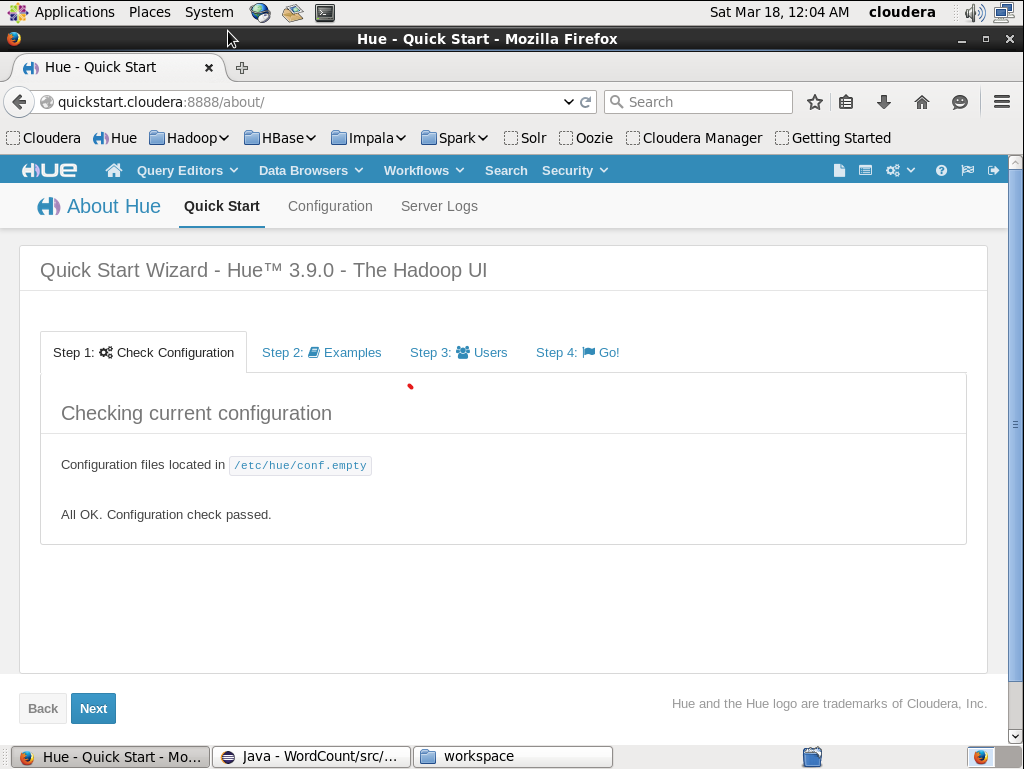
context.write(key, new IntWritable(sum));

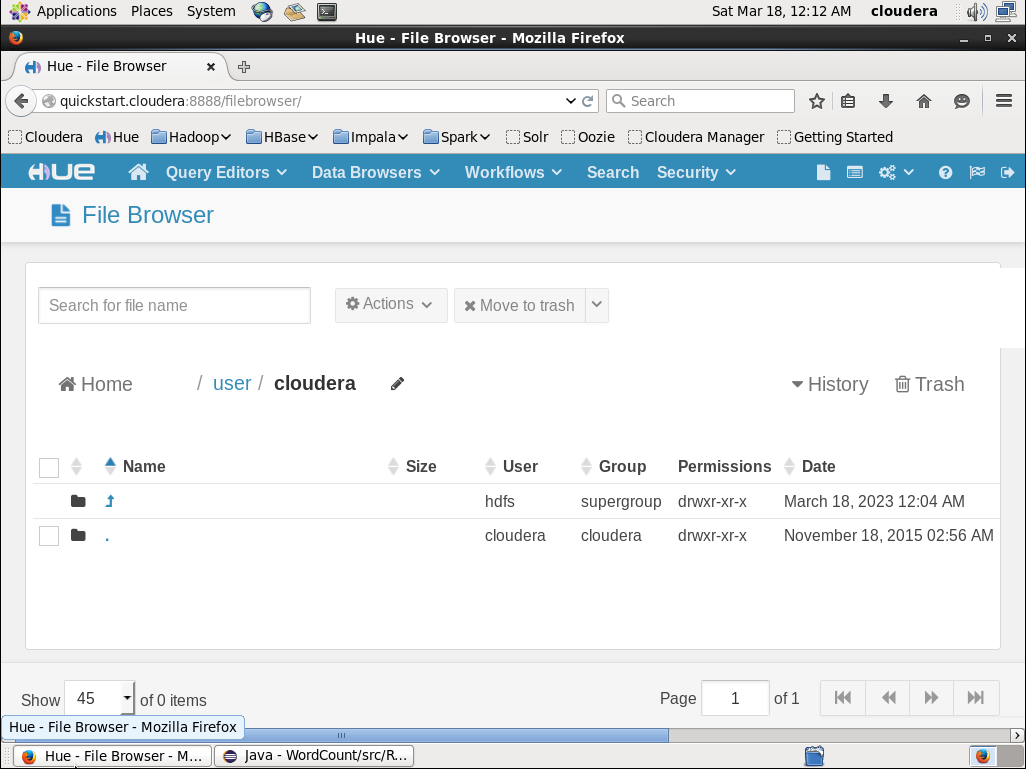
}

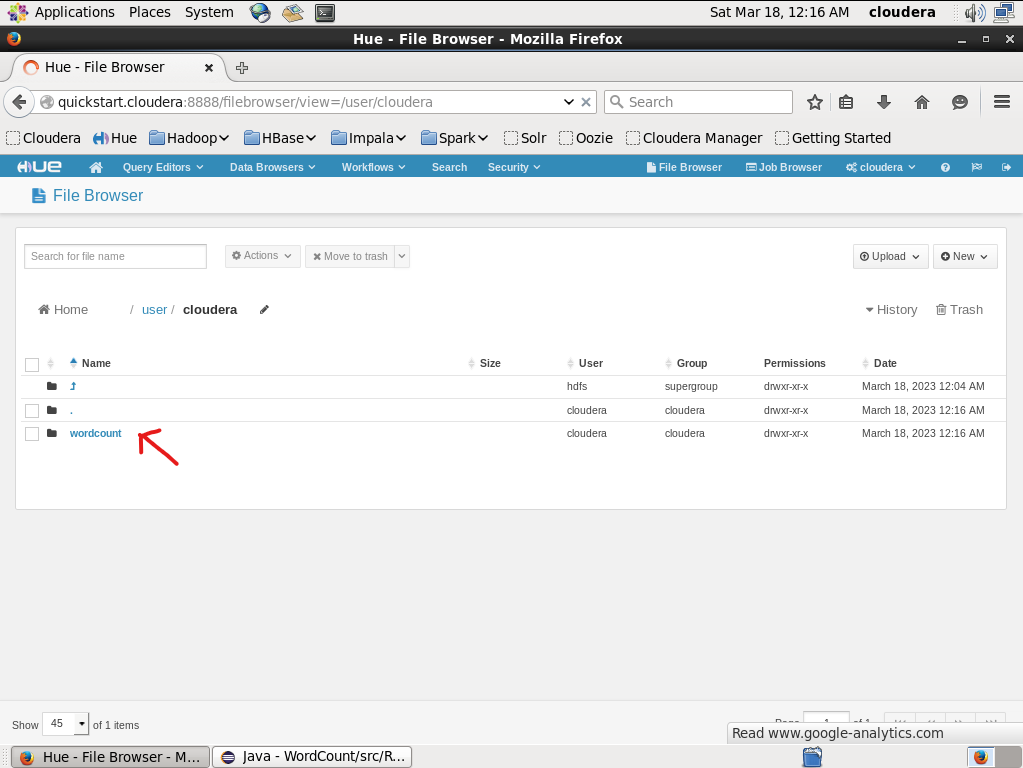
}

****

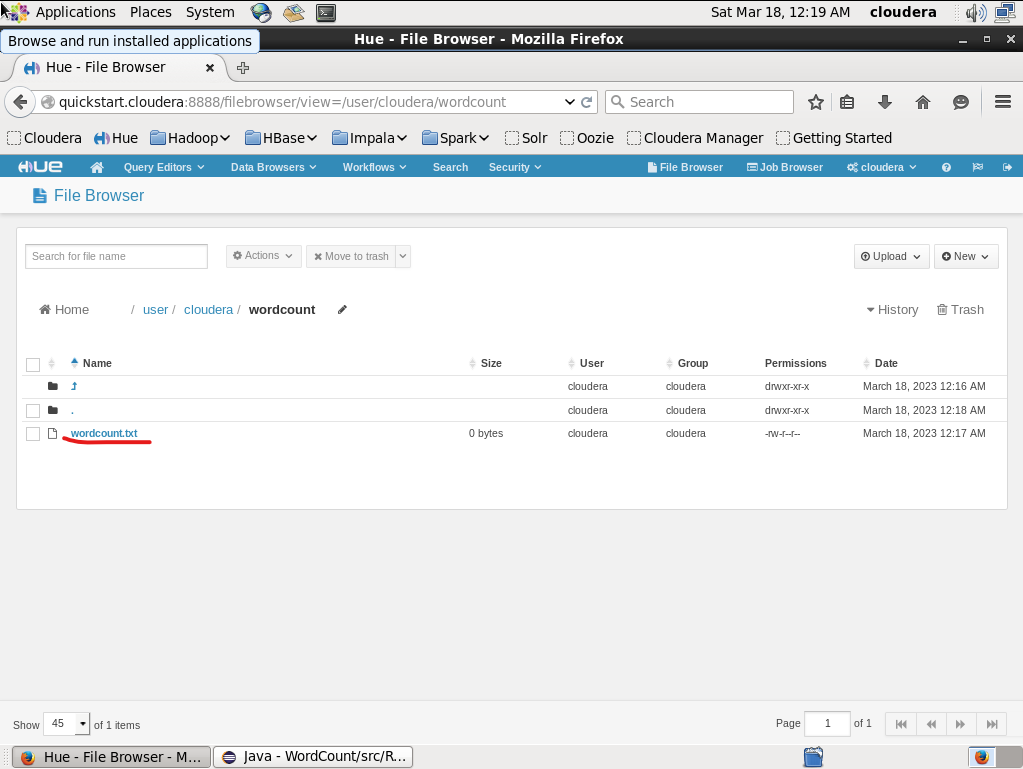
**Figure 8:** login to <http://quickstart.cloudera:8888>

****

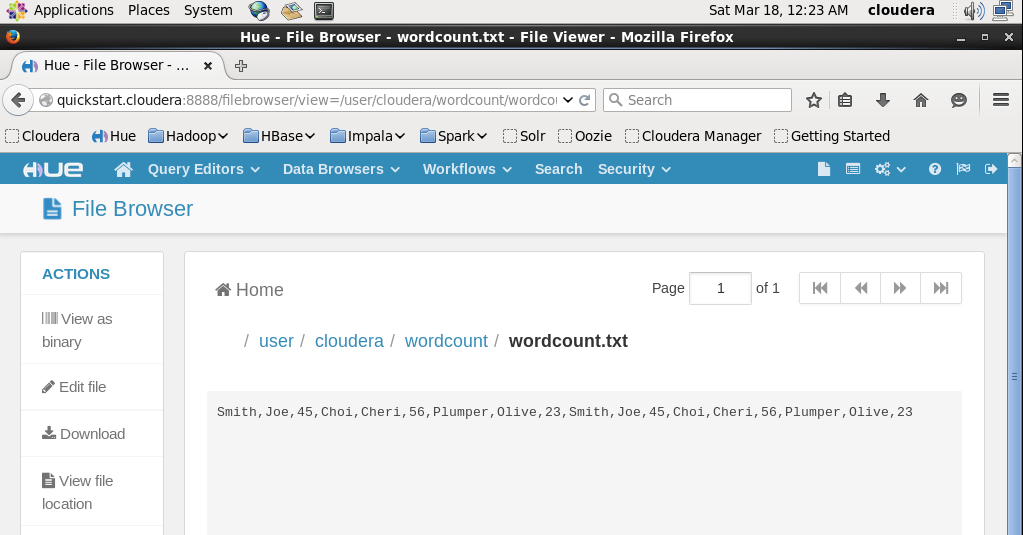
****



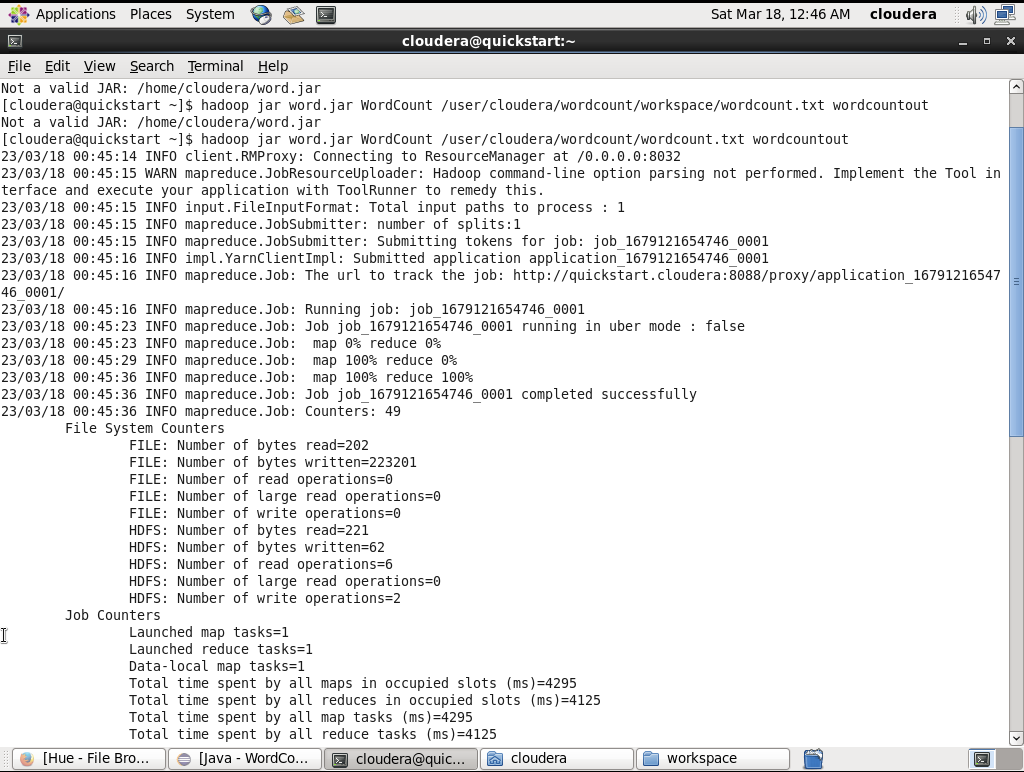
**Figure 9:** Creating a new directory



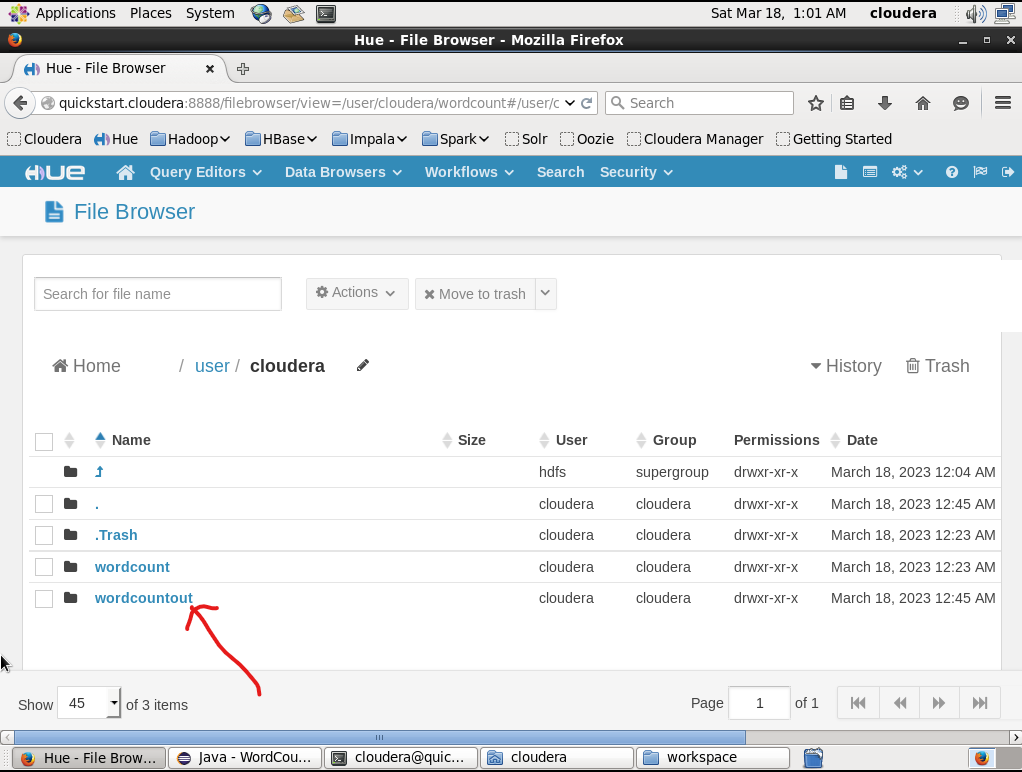
**Figure 10:** Creating new txt file for CSV values



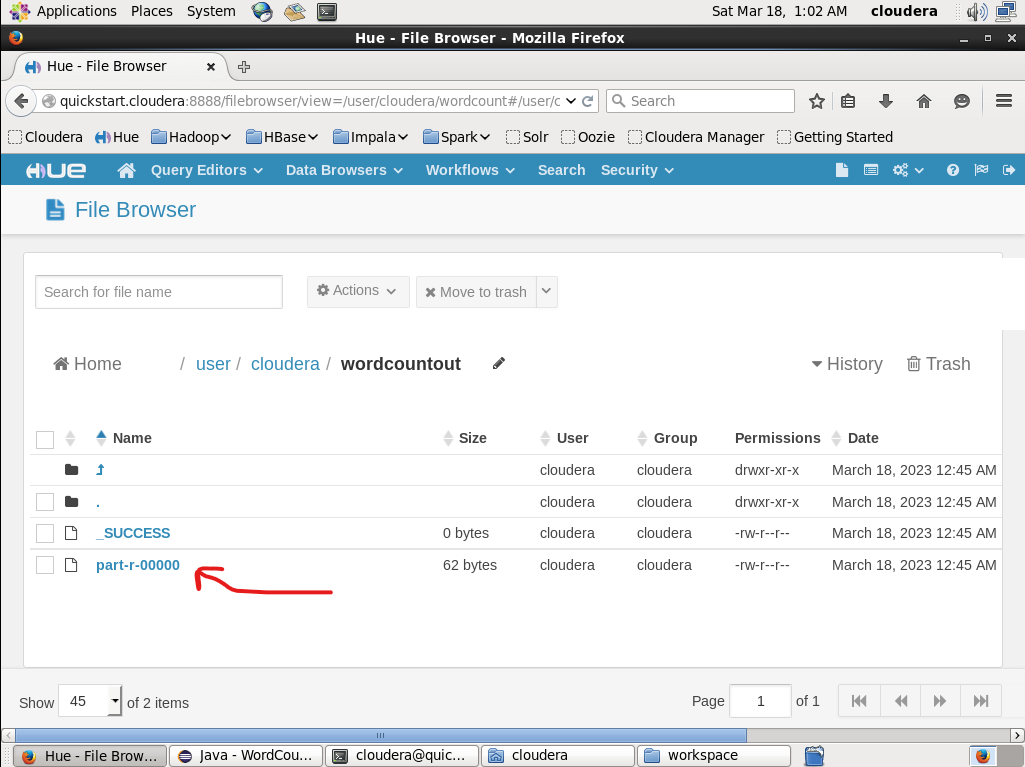
**Figure 11:** Editing the file



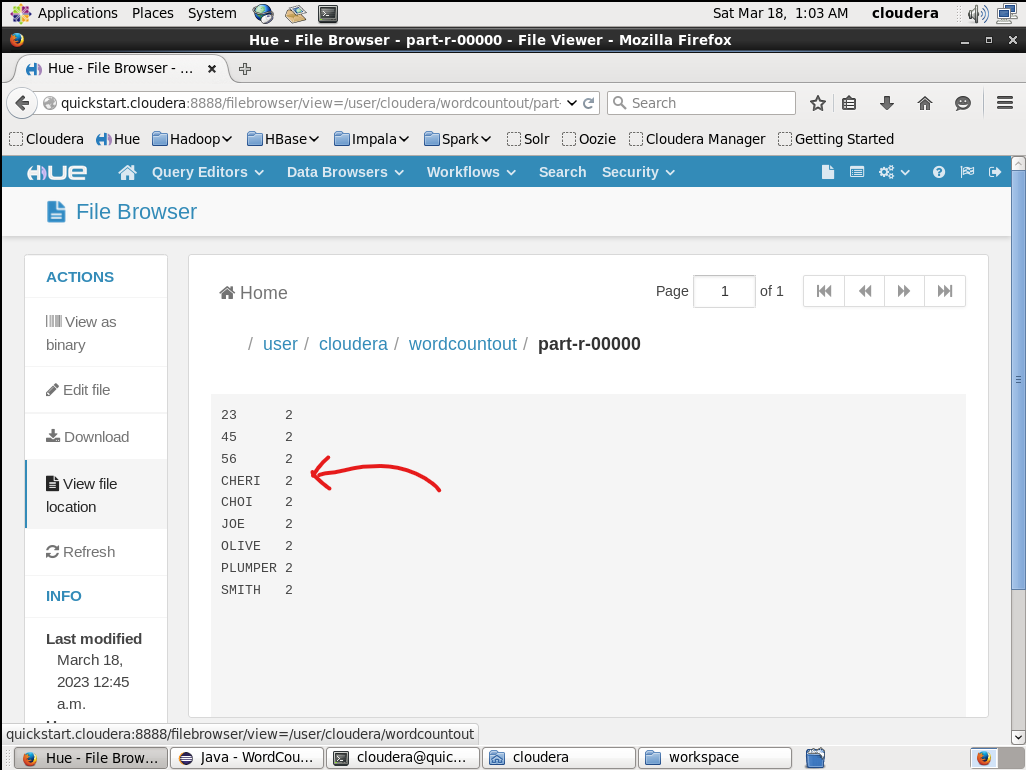
**Figure 12:** In terminal entering the command to run the project



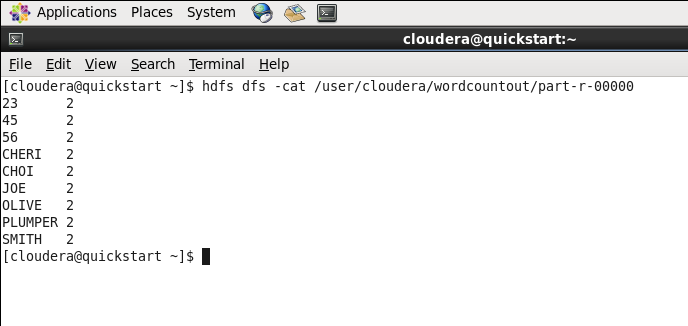
**Figure 13:** Output directory successfully generated



**Figure 14:** Output file



**Figure 15:** The output file

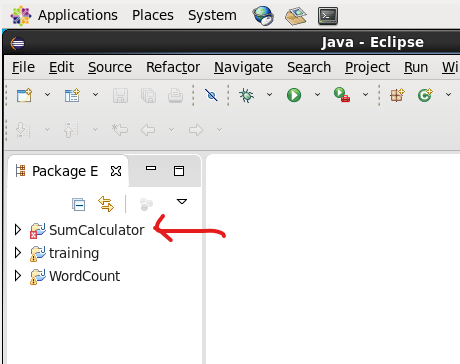


**Figure 16:** Reading the output file using terminal

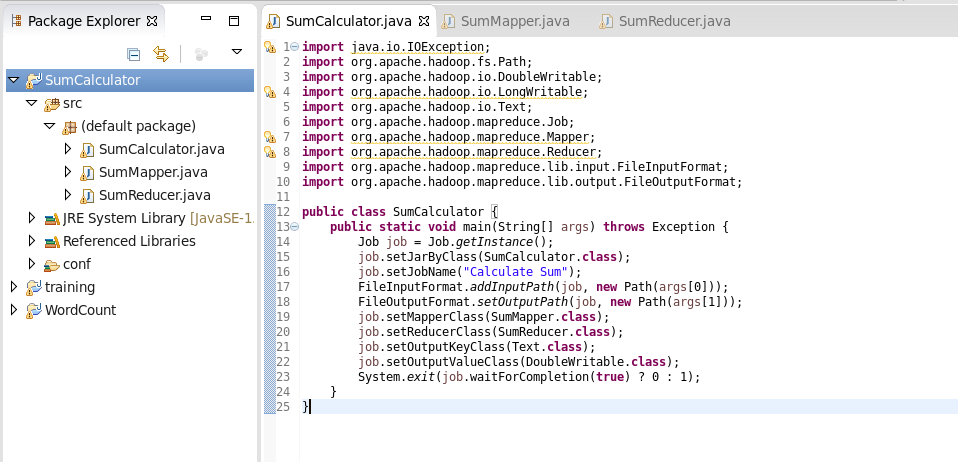
**Lab: 03**

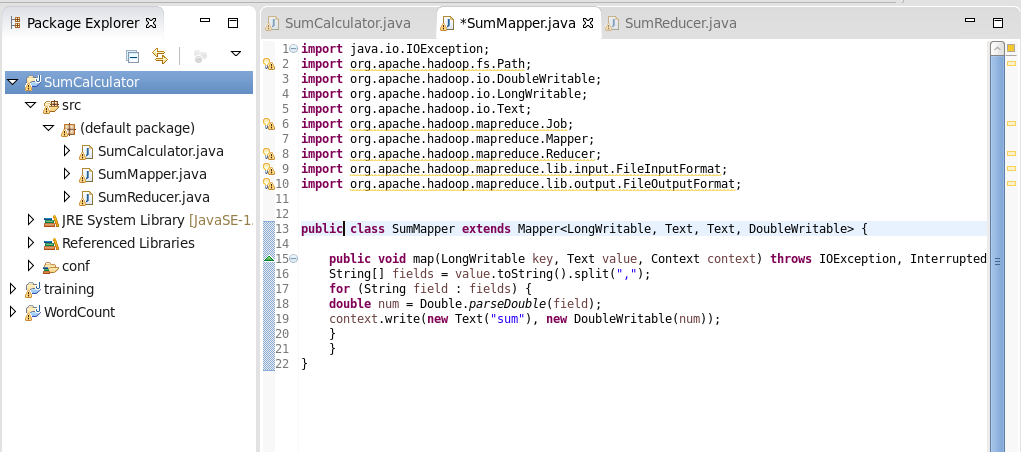
We will have to create a new project in Eclipse.

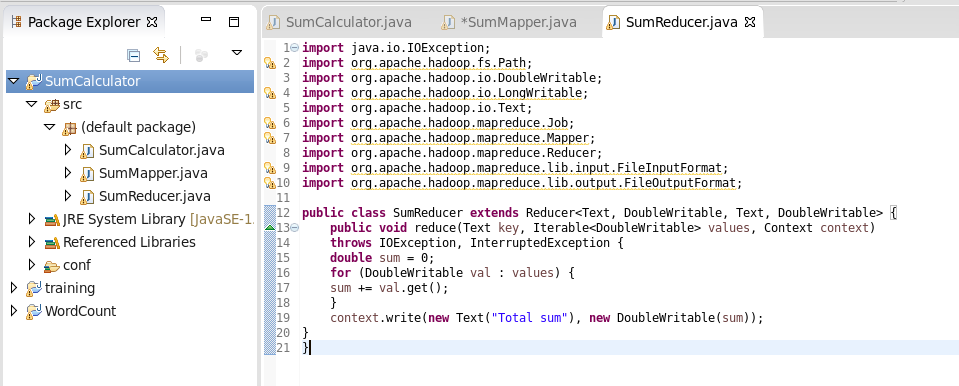
Firstly, we will copy the **training** project and then we will rename the project to **SumCalculator**.



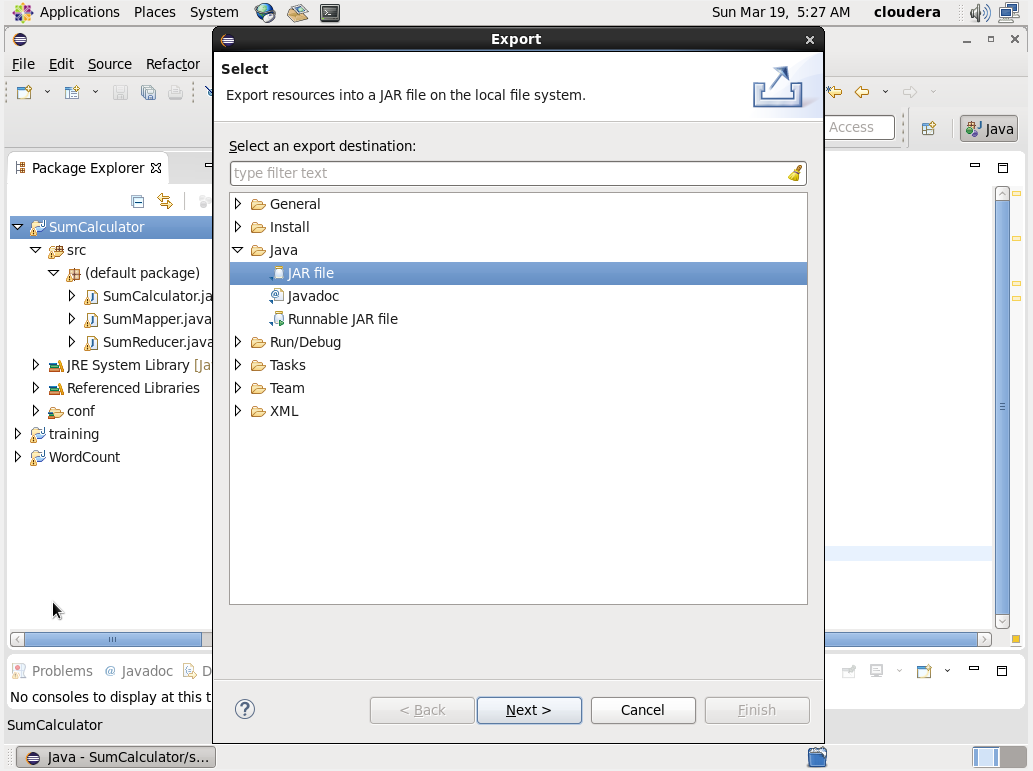
**Figure 1:** Creating a new project named “SumCalculator”

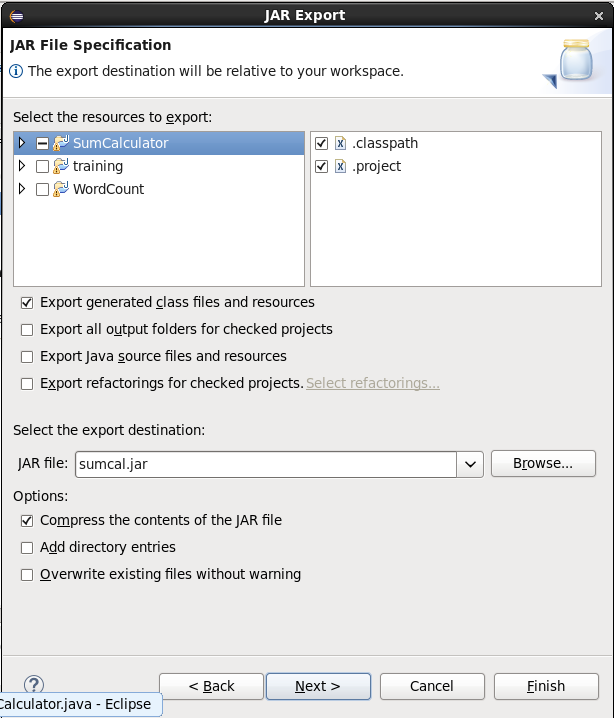
****

****

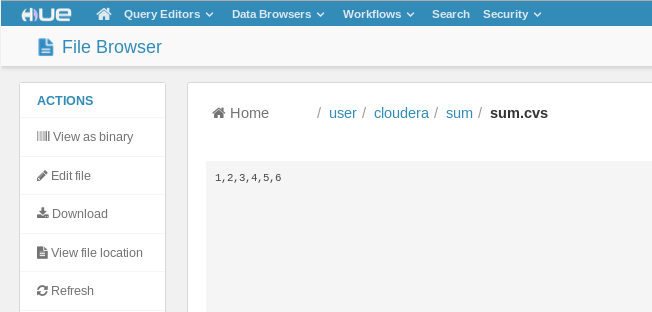
****

**Figure 2:** implanting code in right file

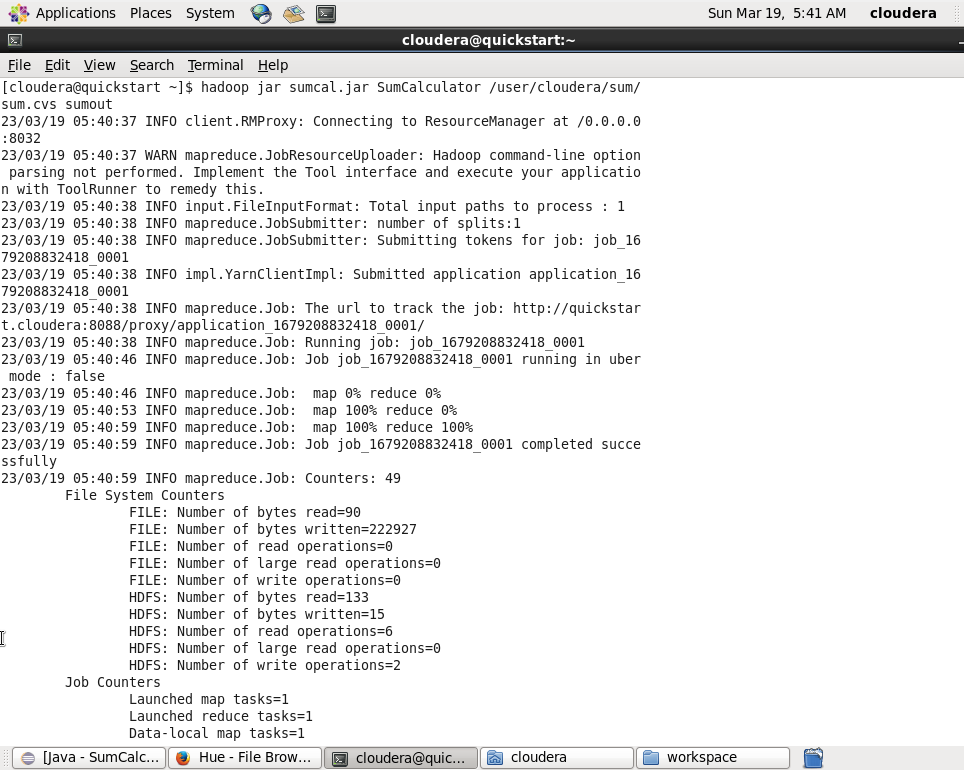
****

****

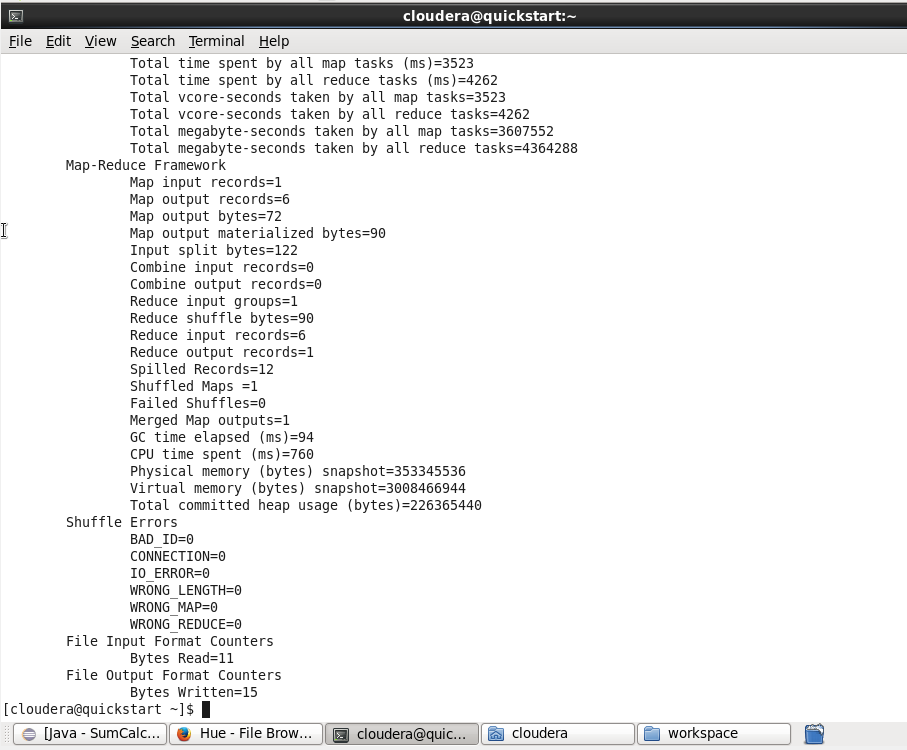
**Figure 3:** Generating jar file of the project

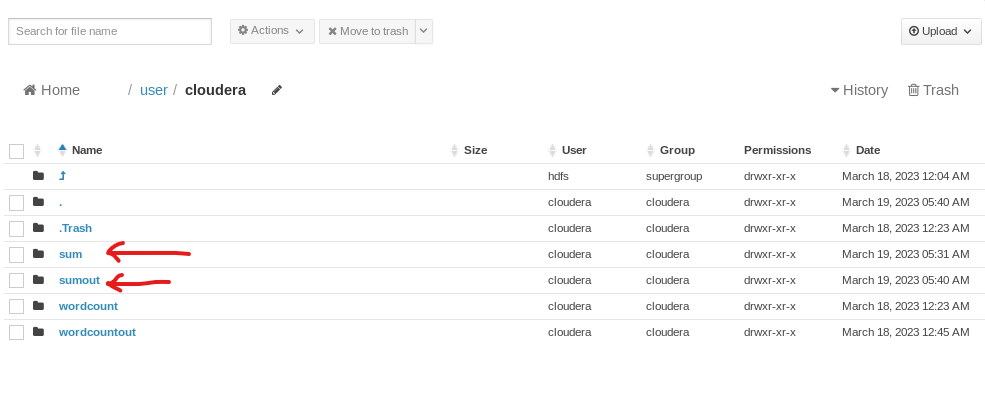
****

**Figure 4:** Creating new directory and new CVS file

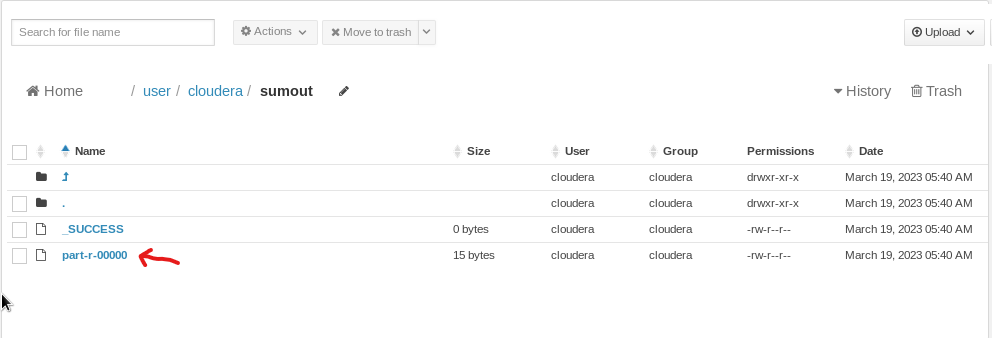
****

**Figure 5:** Entering the command “hadoop jar <jar file> <main class> <input path> <output path>”

****

****

**Figure 6:** Output file successfully generated

****

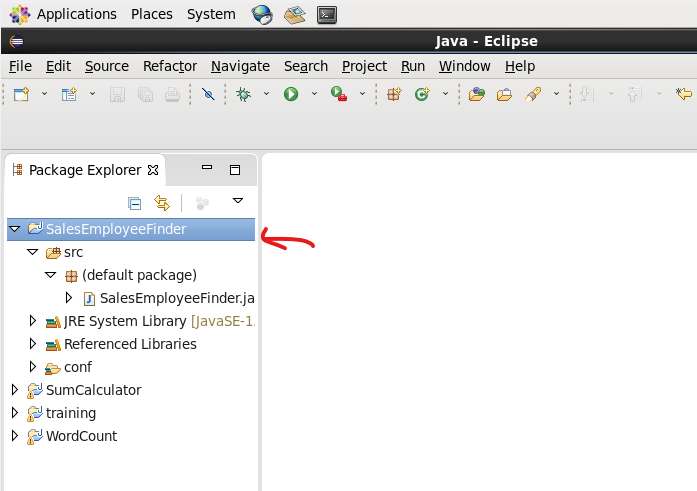
****

**Figure 7:** The result is showing successfully

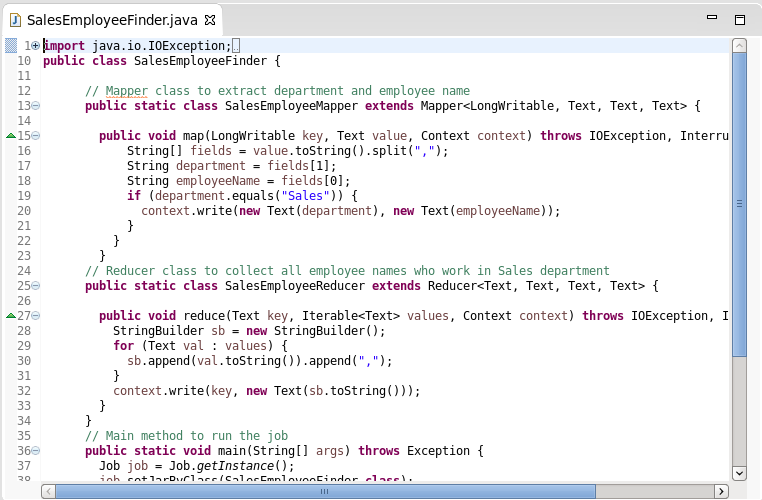
**Lab: 04**

We will have to create a new project in Eclipse.

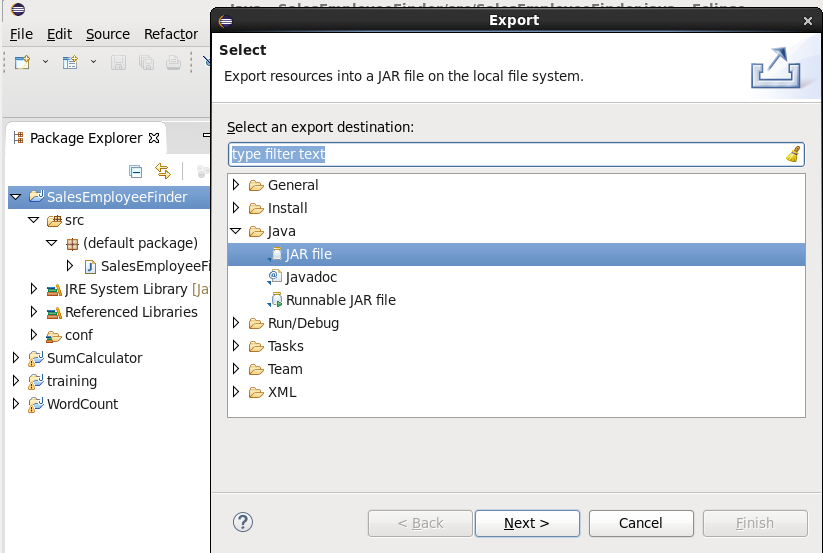
Firstly, we will copy the **training** project and then we will rename the project to **SalesEmployeeFinder** .

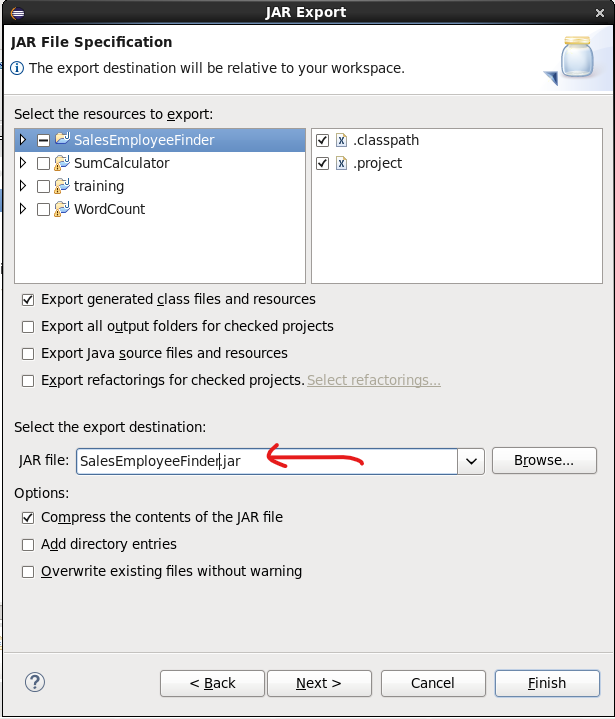


**Figure 1:** Creating a new project named “SalesEmployeeFinder”

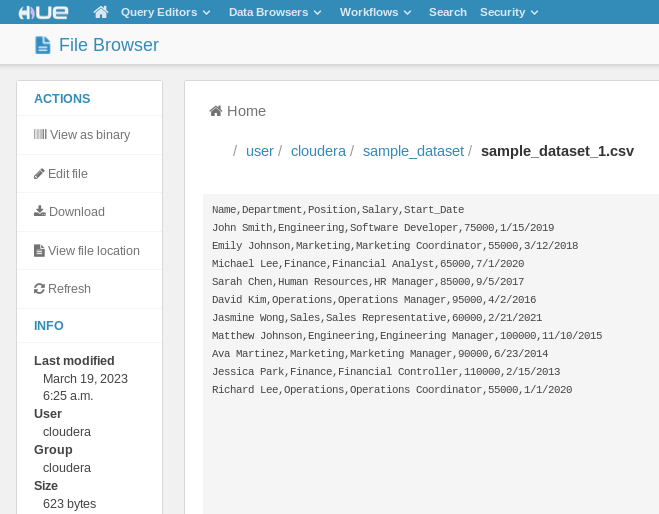
****

**Figure 2:** implanting code in right file

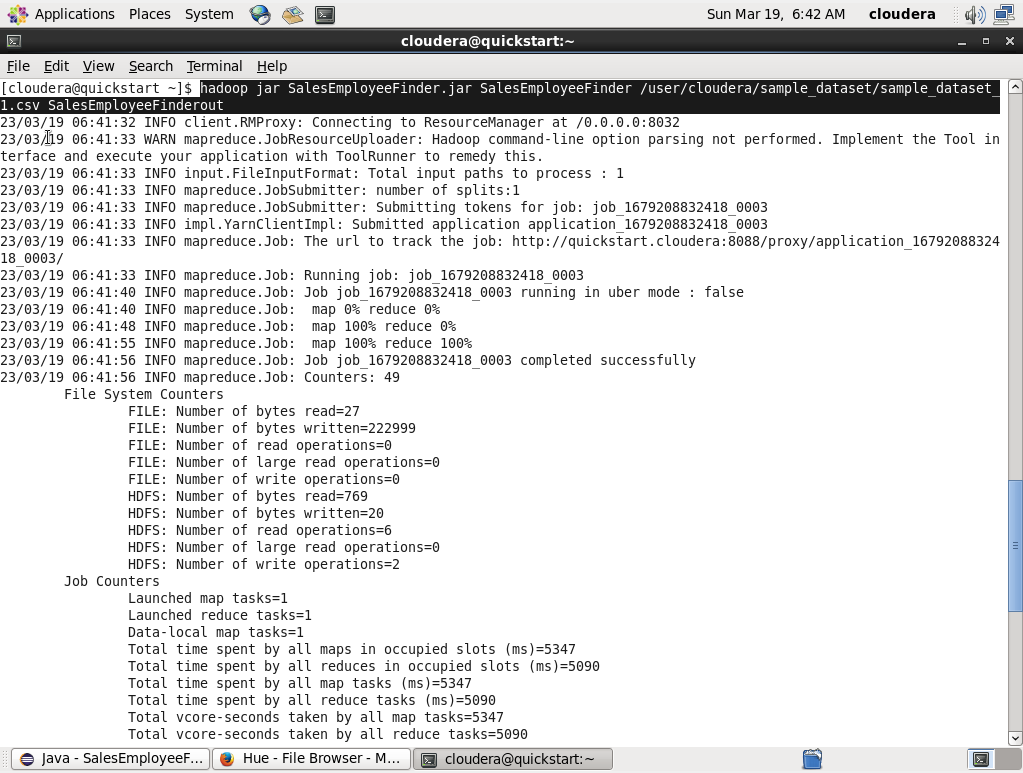
****

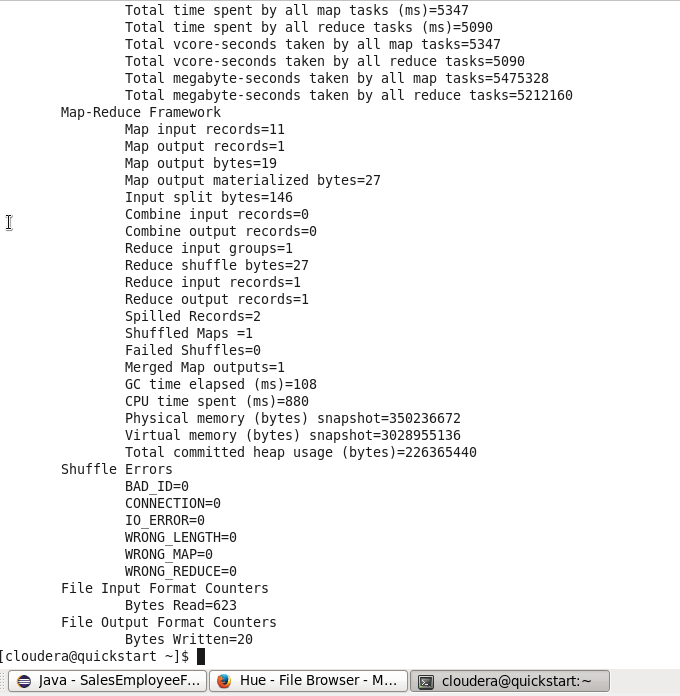
****

**Figure 3:** Generating jar file of the project

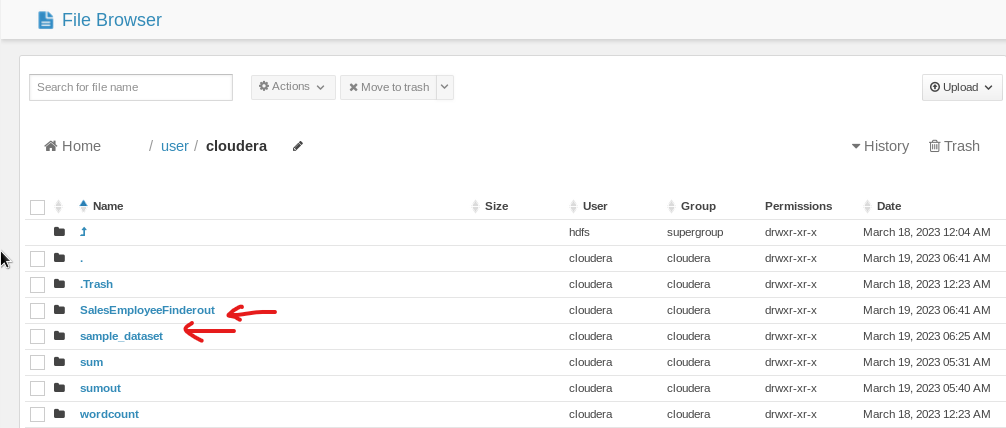
****

**Figure 4:** Creating new directory and new CVS file

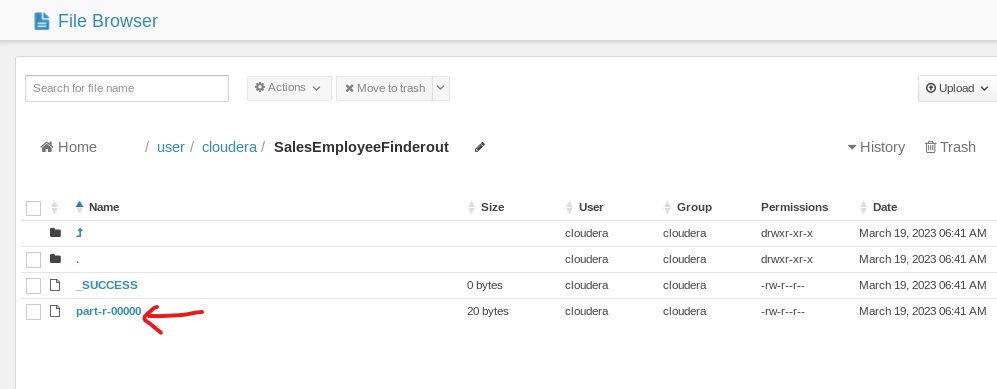
****

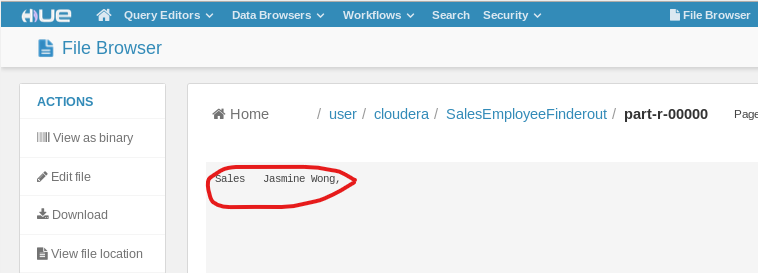
****

**Figure 5:** Entering the command “hadoop jar <jar file> <main class> <input path> <output path>”

****

**Figure 6:** Output file successfully generated

****

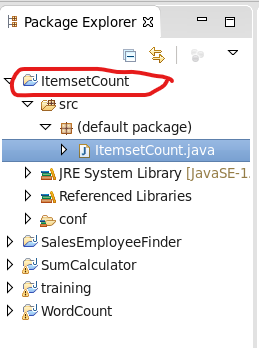
****

**Figure 7:** The result is showing successfully

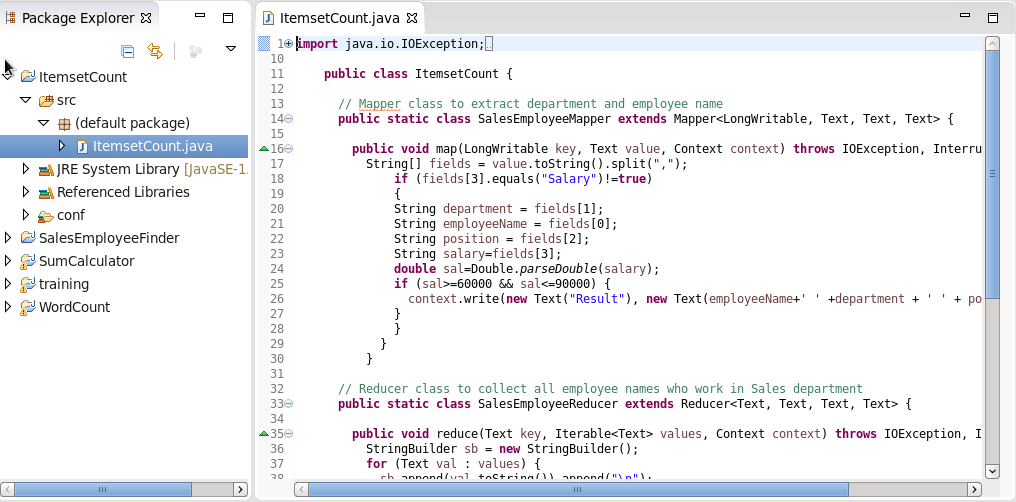
**Lab: 05**

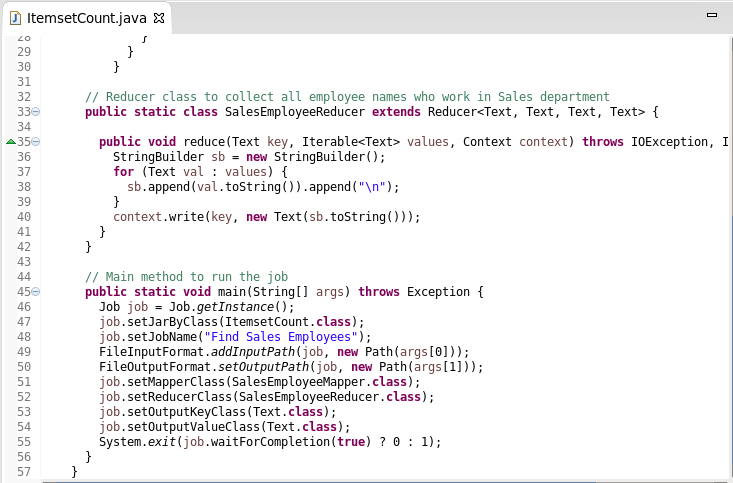
**We will have to create a new project in Eclipse.**

Firstly, we will copy the **training** project and then we will rename the project to **ItemsetCount** .

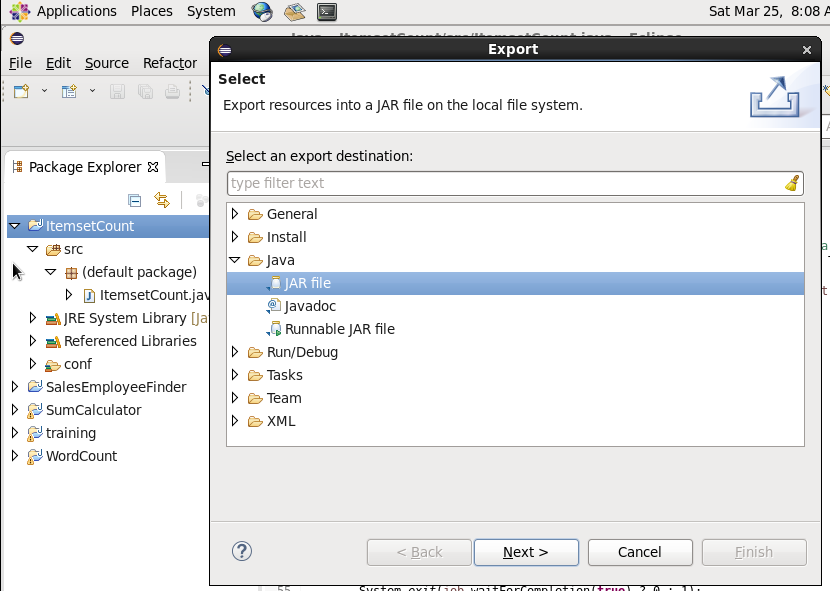


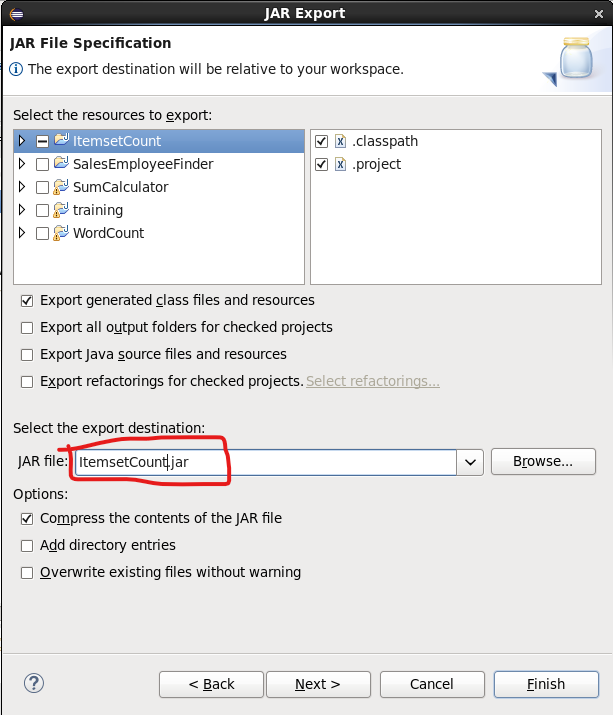
**Figure 1:** Creating a new project named “ItemsetCount”

****

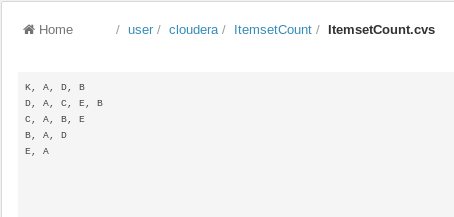
****

**Figure 2:** implanting code in right file

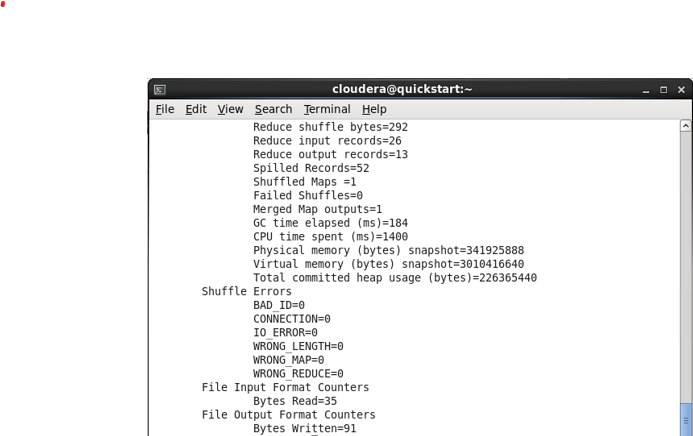
****

****

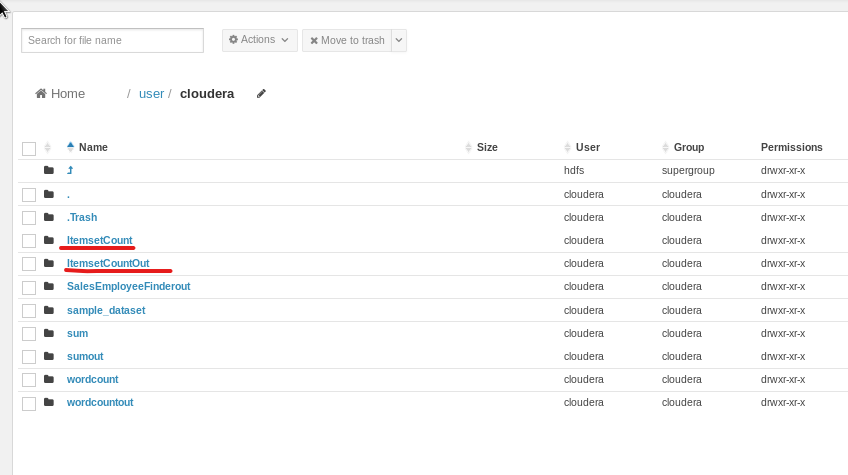
**Figure 3:** Generating jar file of the project

****

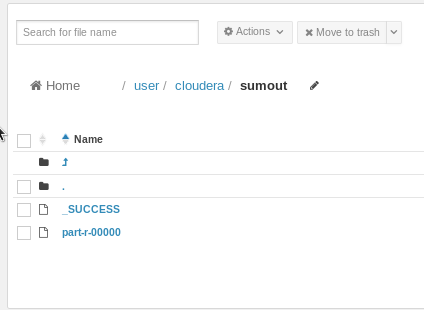
**Figure 4:** Creating new directory and new CVS file

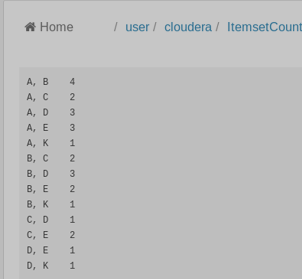
****

**Figure 5:** Entering the command “hadoop jar <jar file> <main class> <input path> <output path>”

****

**Figure 6:** Output file successfully generated

****

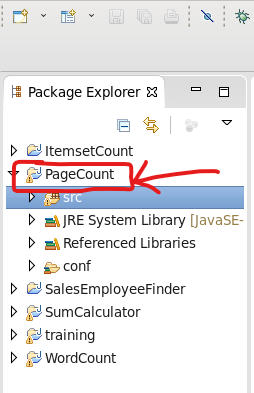
****

**Figure 7:** The result is showing successfully

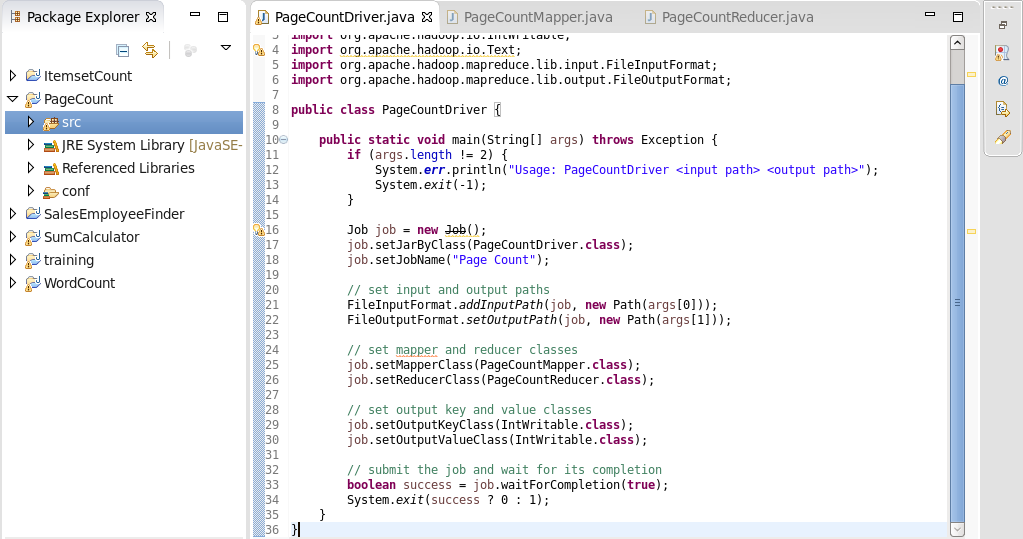
**Lab: 06**

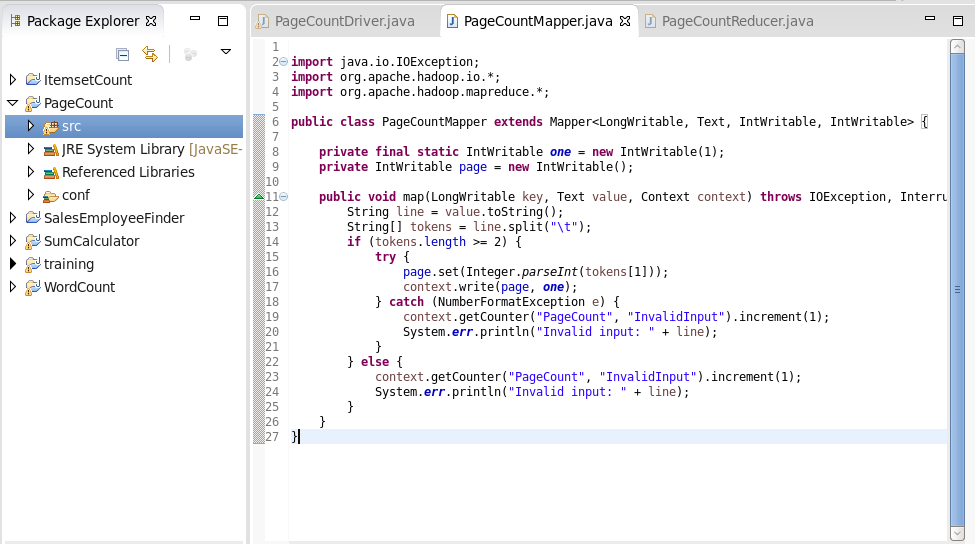
**We will have to create a new project in Eclipse.**

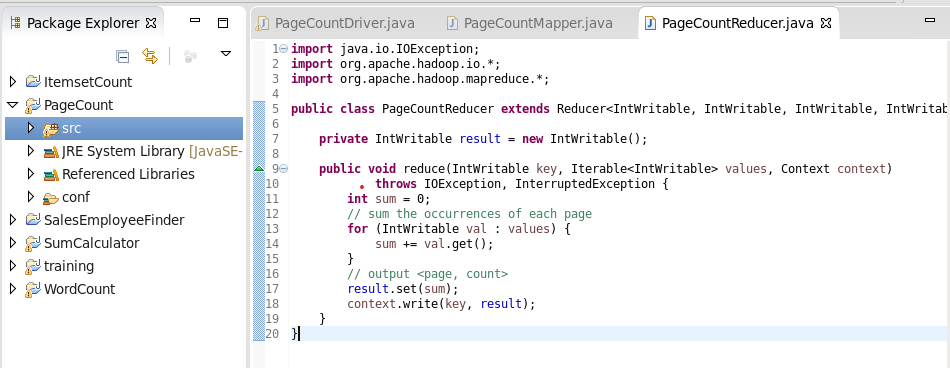
Firstly, we will copy the **training** project and then we will rename the project to PageCount .



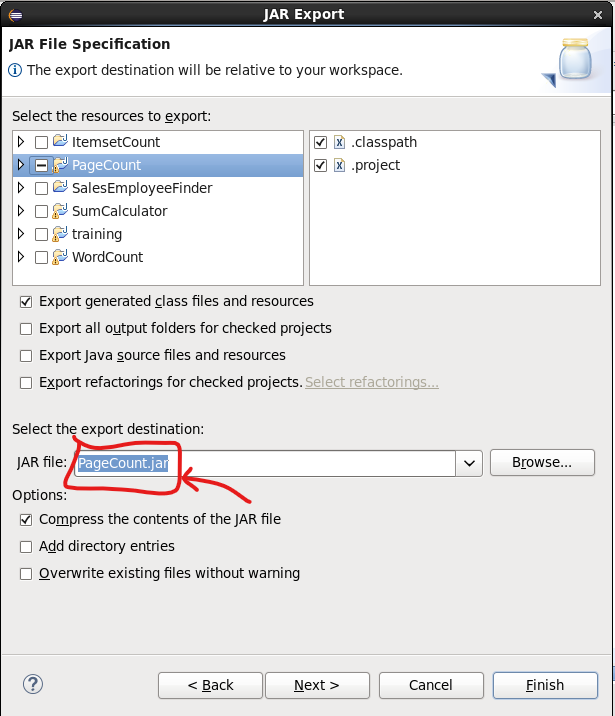
**Figure 1:** Creating a new project named “PageCount”

****

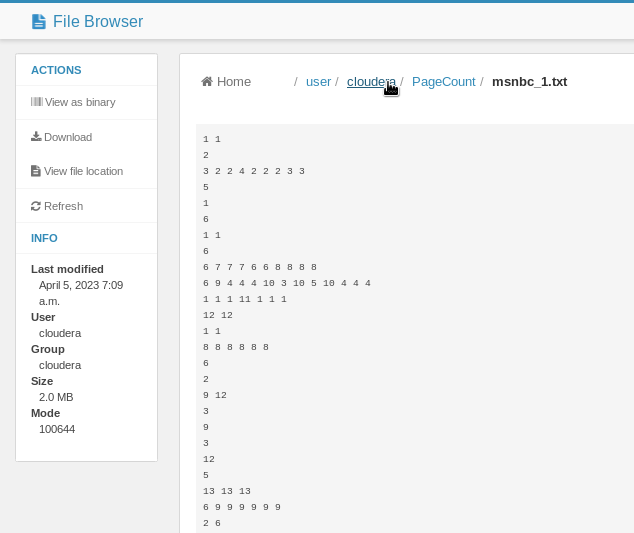
****

****

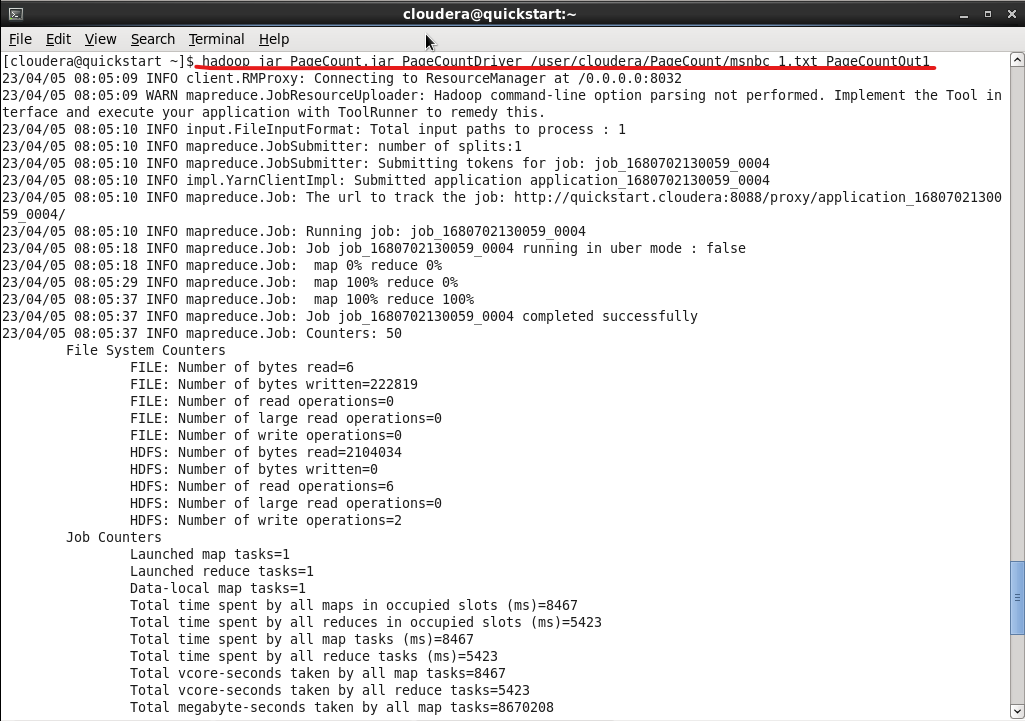
**Figure 2:** implanting code into specific file

****

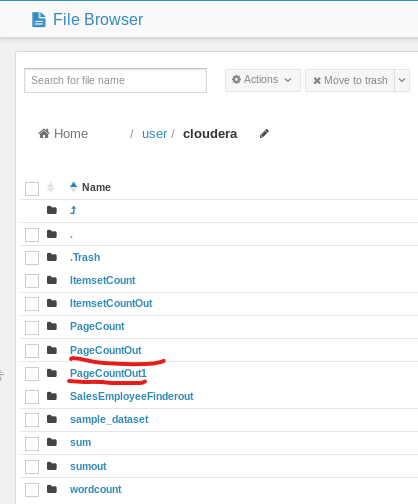
**Figure 3:** Generating jar file of the project

****

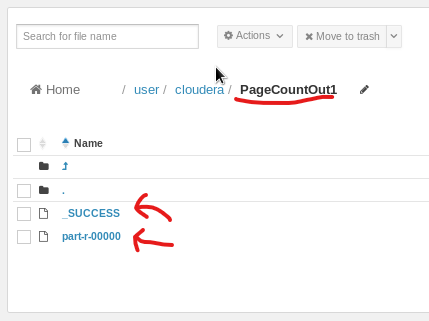
**Figure 4:** Creating new directory and new CVS file

****

**Figure 5:** Entering the command “hadoop jar <jar file> <main class> <input path> <output path>”

****

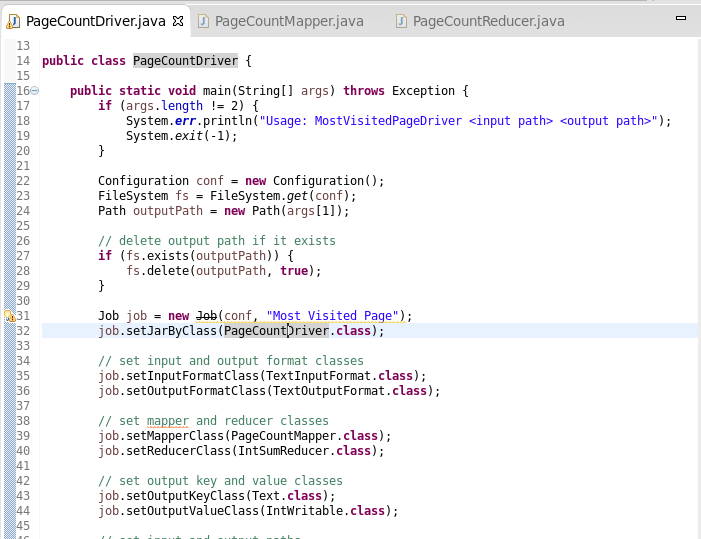
**Figure 6:** Output file successfully generated

****

**Figure 7:** The result is showing successfully

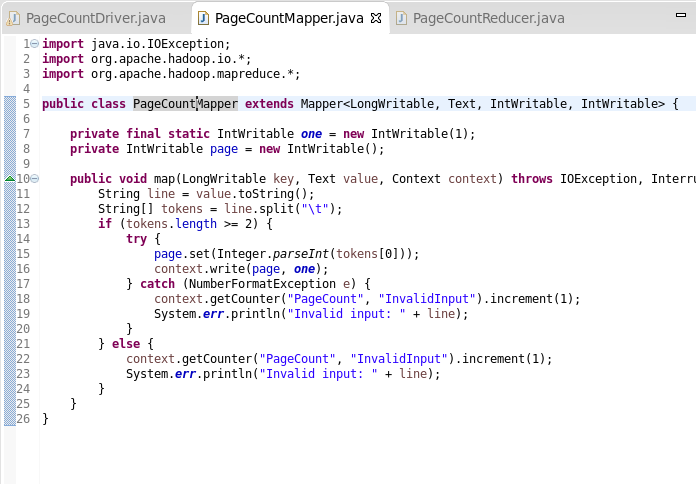
**Question 2:**

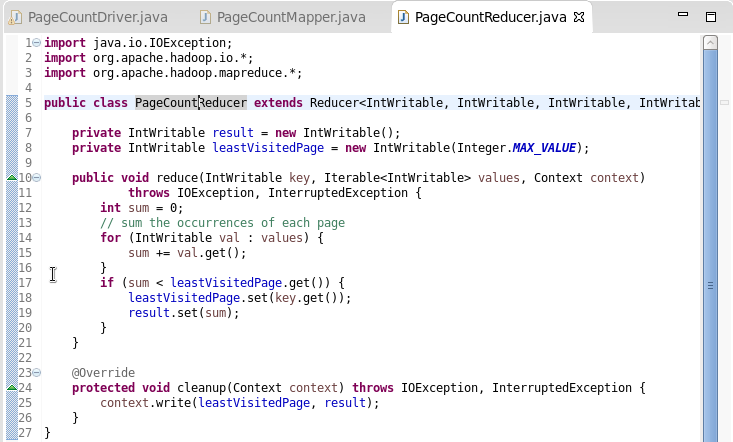
We will have to change the driver code like this.

****

**Question 3:**

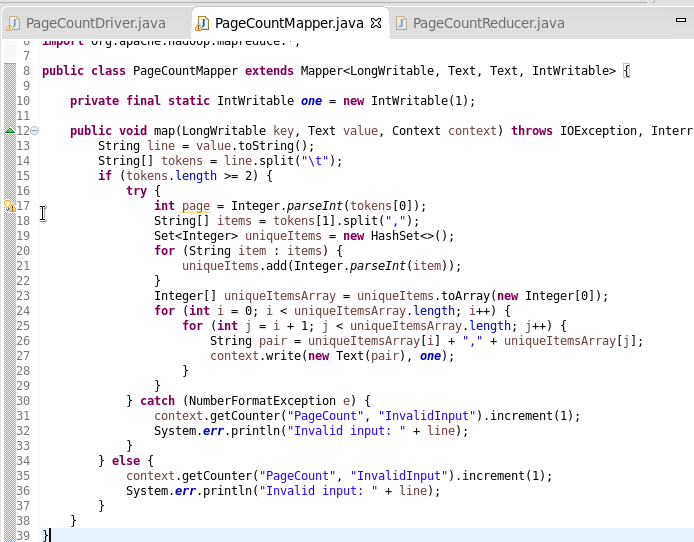
We will have to change the Mapper and Reducer code like this.

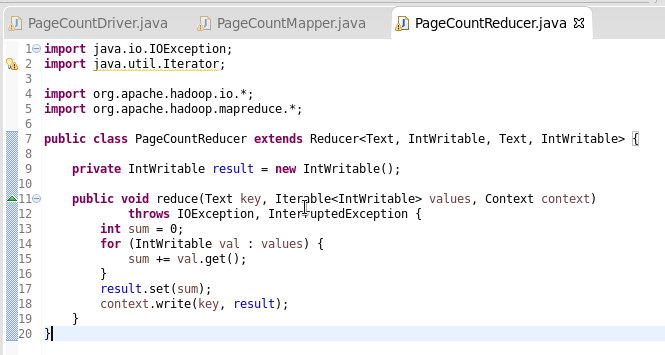


****

**Question 4:**

We will have to change the Mapper and Reducer code like this.

****

****