**Name:-Arya More.**

**Roll no:527**

**Subject:- Business Intelligence and Big Data**

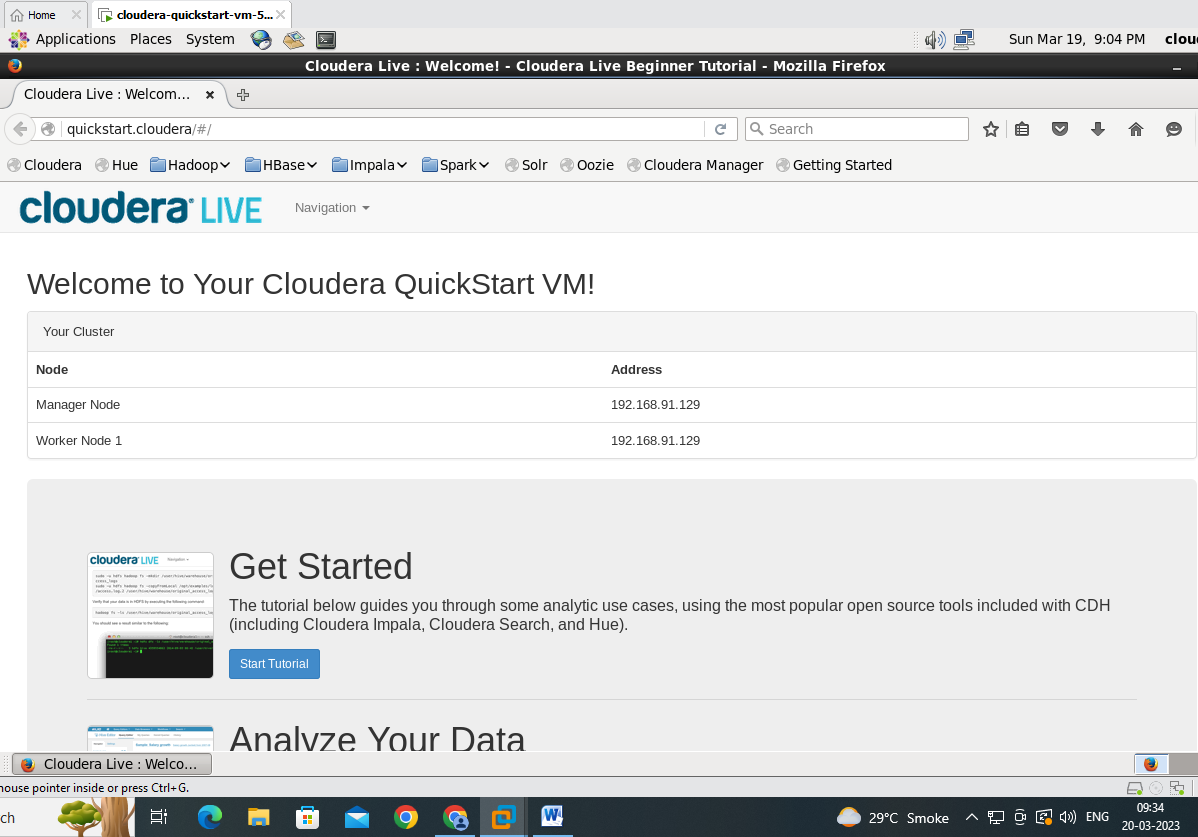
**MSc-CS Part1**

**Practical 2-** Implementation Map-Reduce Program for Word Count Problem.

**Step1**



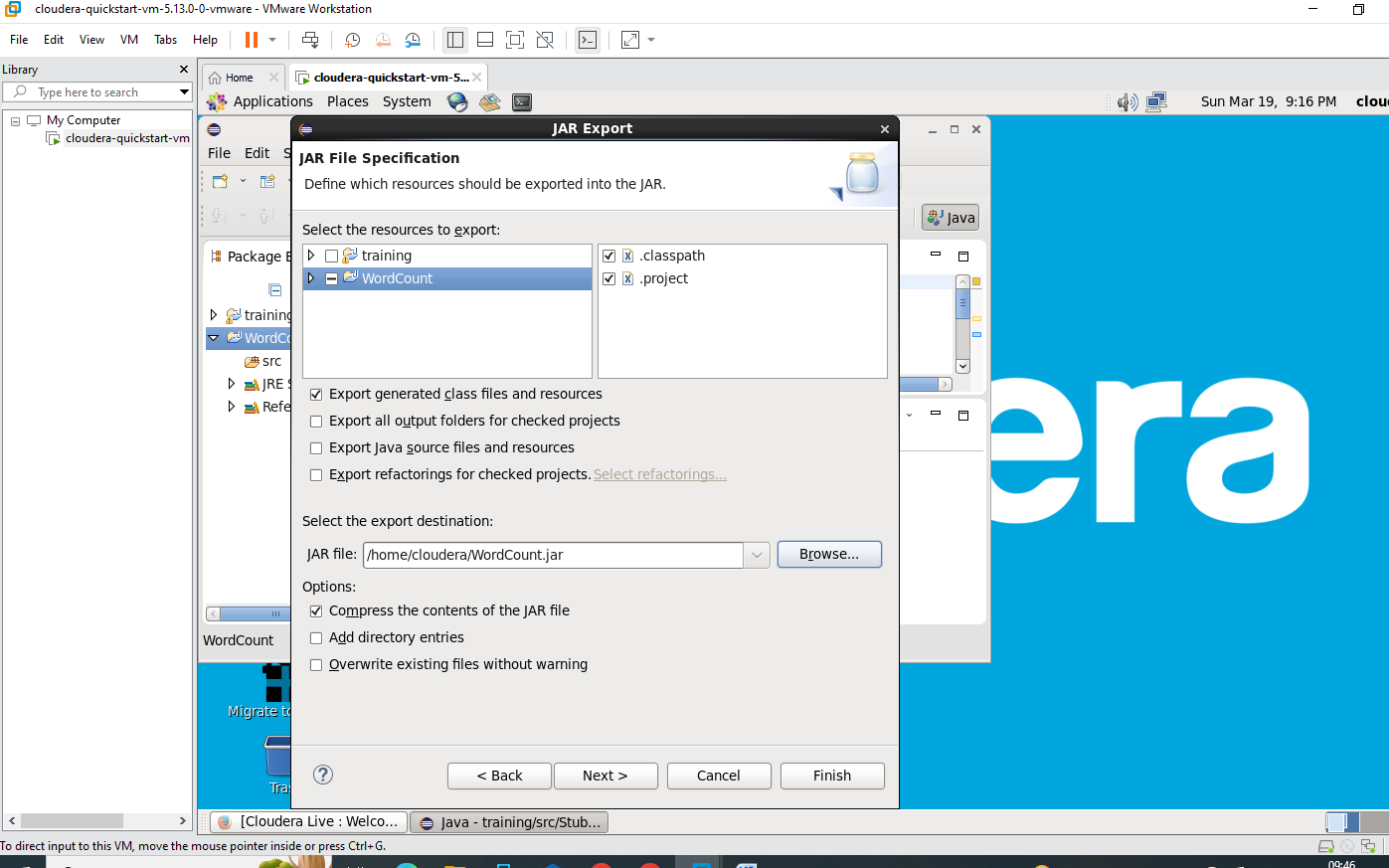
Step 2



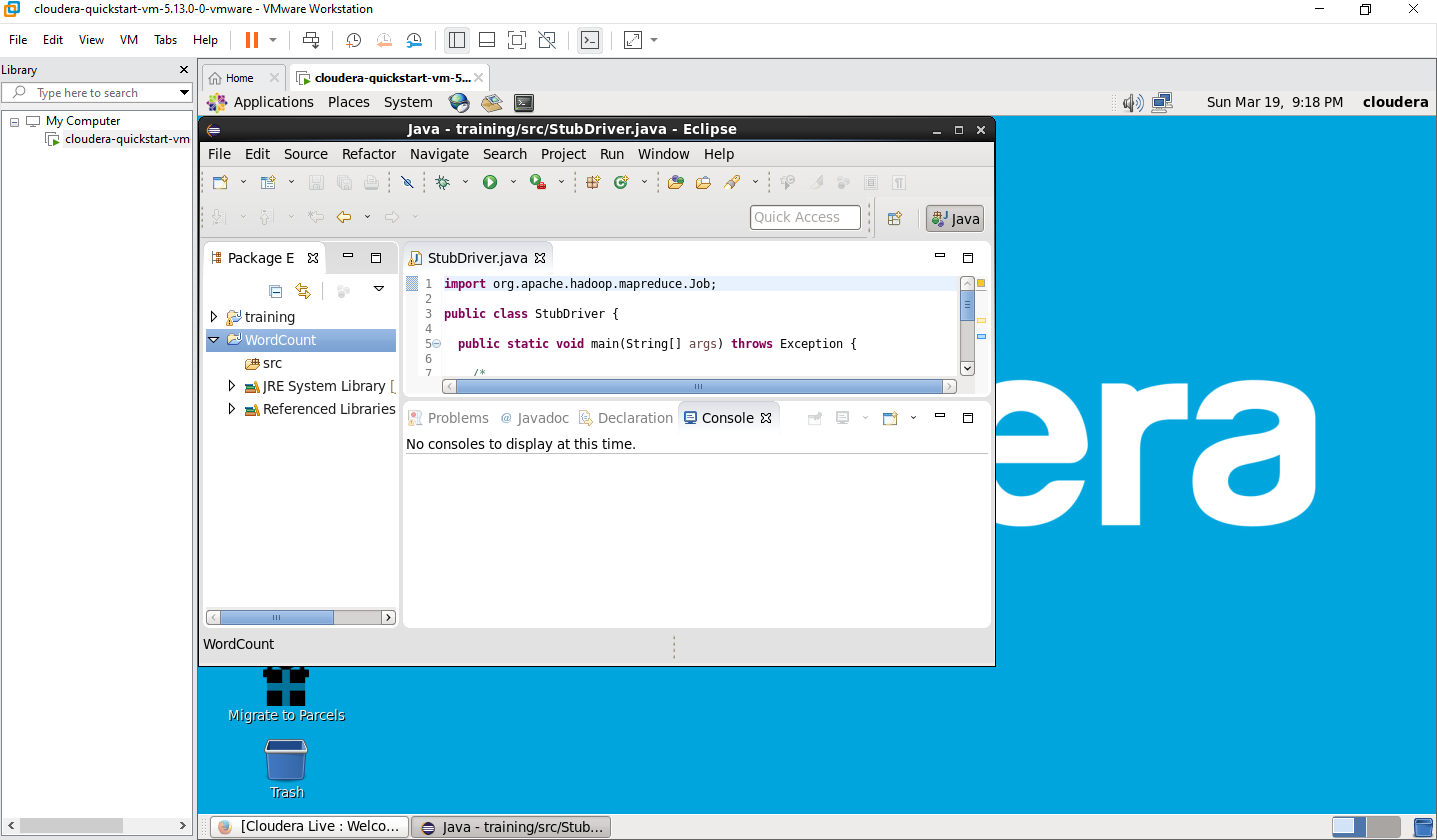
Step 3



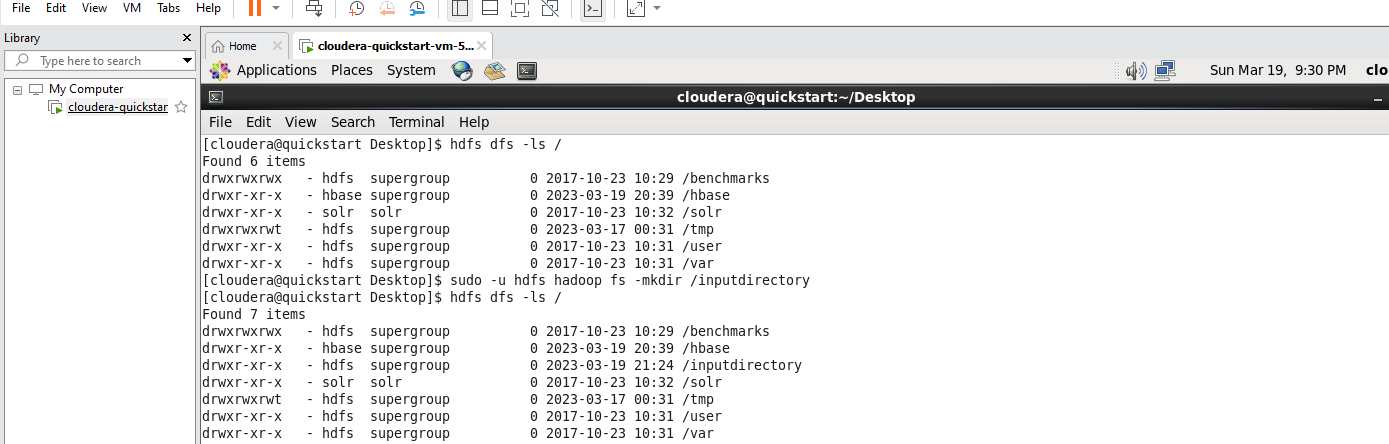
Step 4



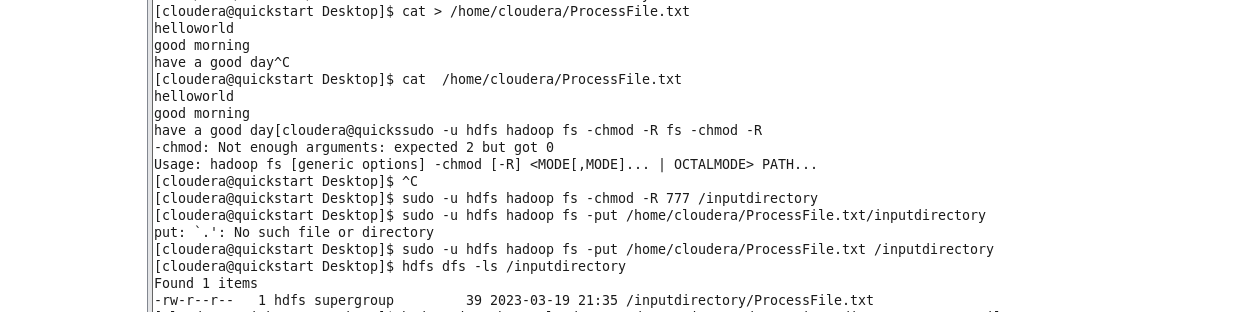
Step 5



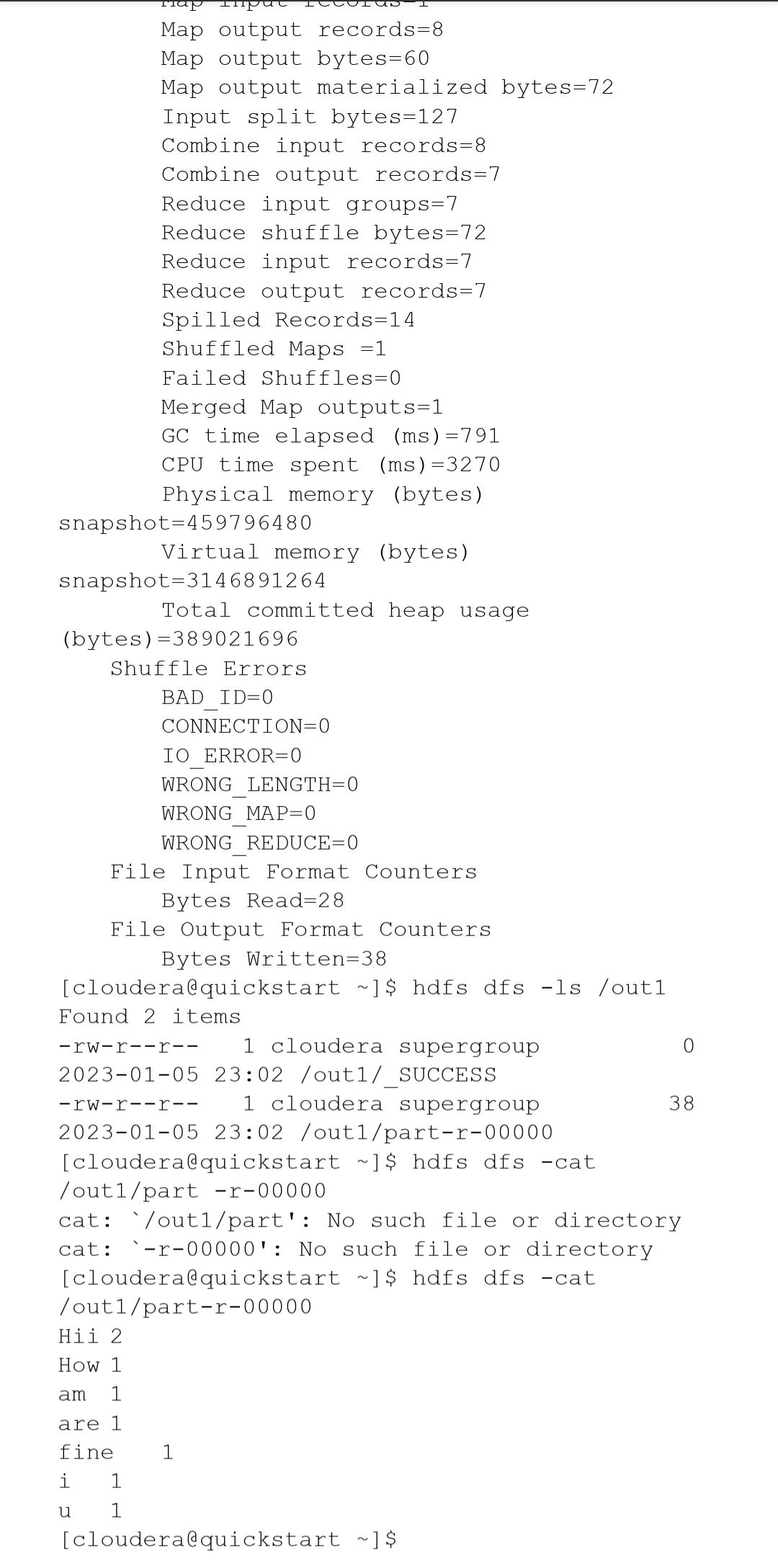
Step 6



Step 7



Step 8



**Practical-3:-** Write a Pig Script For Solving counting Problems.

Steps :

cat> /home/cloudera/input.csv

cat /home/cloudera/input.csv

pig -x local

lines = load '/home/cloudera/input.csv' as (line:chararray);

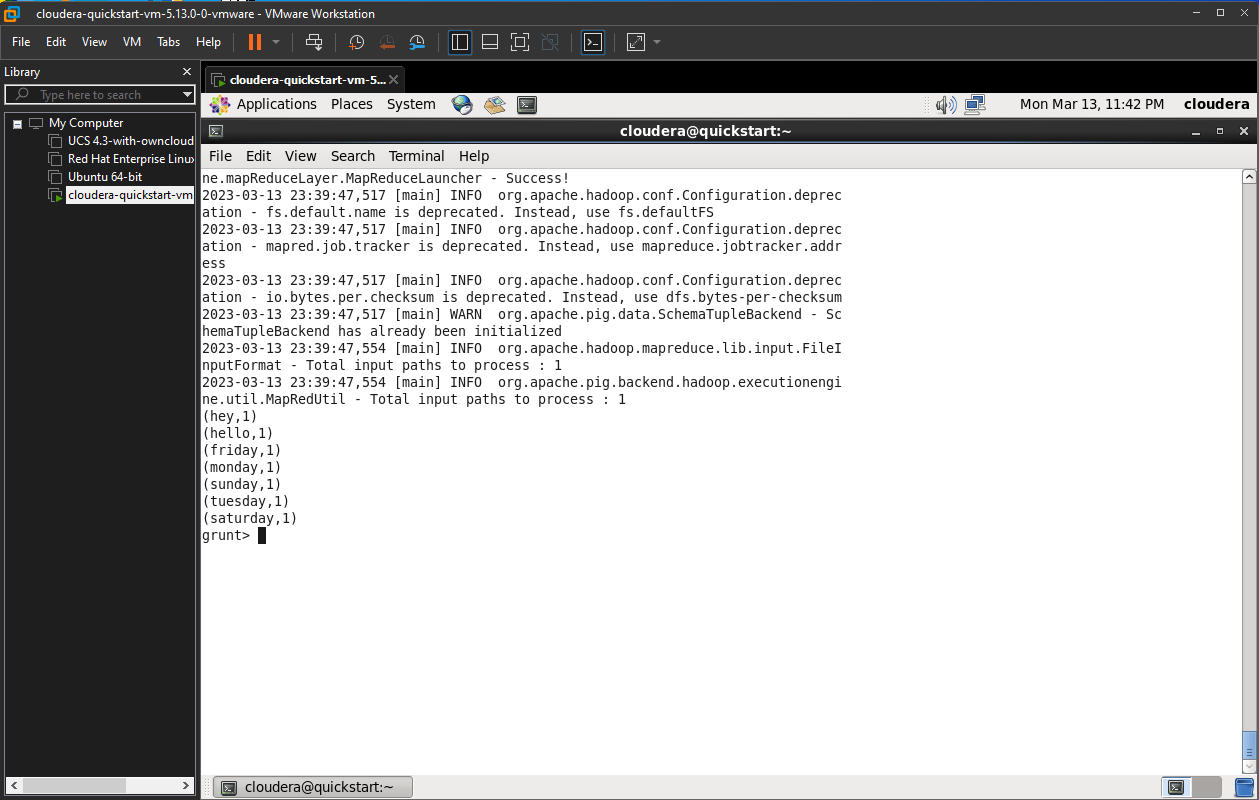
words = foreach lines GENERATE FLATTEN(TOKENIZE(line)) as woed;

grouped = GROUP words by woed;

wordcount = foreach grouped GENERATE group, COUNT(words);

dump wordcount;

output:



**Practical-4:-**Install HBase and use the HBase Data model Store and retrieve Data

Steps :

//Start HBase

hbase shell

//HBase Commands

status

version,

table\_help

whoami

//Data Definition Language

create ‘employee’, ’Name’, ’ID’, ’Designation’, ’Salary’, ’Department’

//Verify created table

list

//Disable single table

disable ‘employee’

scan ‘employee’

//or

is\_disable ‘employee’

//Disable multiple tables

disable\_all ‘e.\*’

// Enabling table

enable‘employee’

//Or

is\_enabled'employee'

//create new table

create‘student’, ‘name’, ‘age’, ‘course’

put ‘student’, ‘sharath’, ‘name:fullname’, ‘sharathkumar’

put ‘student’, ‘sharath’, ‘age:presentage’, ‘24’

put ‘student’, ‘sharath’, ‘course:pursuing’, ‘Hadoop’

put ‘student’, ‘shashank’, ‘name:fullname’, ‘shashank R

put ‘student’, ‘shashank’, ‘age:presentage’, ‘23’

put ‘student’, ‘shashank’, ‘course:pursuing’, ‘Java’

//Get Information

get ‘student’, ‘shashank’

get ‘student’, ‘sharath’

get ‘student’, ‘sharath’, ‘course’

get ‘student’, ‘shashank’, ‘course’

get ‘student’, ‘sharath’, ‘name’

//Scan

scan ‘student’

//Count

Count ‘student’

//Alter

alter ‘student’, NAME=>’name’, VERSIONS=>5

put ‘student’, ‘shashank’, ‘name:fullname’, ‘shashank Rao’

scan ‘student’

//Delete

delete ‘student’, ‘shashank’, ‘name:fullname’

**Practical-5:-**Install Hive and use Hive Create Store Structured databases.

Steps :

cat > /home/cloudera/employee.txt

1~Sachine~Pune~Product Engineering~100000~Big Data

2~Gaurav~Banglore~Sales~90000~CRM

3~Manish~Chennai~Recruiter~125000~HR

4~Bhushan~Hyderabad~Developer~50000~BFSI

cat /home/cloudera/employee.txt

sudo -u hdfs hadoop fs -put /home/cloudera/employee.txt /inputdirectroy

hdfs dfs -ls /

hdfs dfs -ls /inputdirectory

hadoop fs -cat /inputdirectory/employee.txt

hive

show databases;

create database organization;

show databases;

use organization;

show tables;

hive> create table employee(

> id int,

> name string,

> city string,

> department string,

> salary int,

> domain string)

> row format delimited

> fields terminated by '~';

show tables;

select \* from employee;

show tables;

load data inpath '/inputdirectory/employee.txt' overwrite into table employee;

show tables;

select \* from employee;

**Practical-6**:- Write a program to construct different types of k-shingles for a given document.

install.packages("tm")

require("tm")

install.packages("devtools")

readinteger <-function()

{

n<-readline(prompt="Enter value of k-1:")

k<- as.integer(n)

u1<- readLines("C:/MSC Notes/file.txt")

Shingle <-0

i<-0

while(i<nchar(u1)-k+1){

Shingle[i] <- substr(u1,start=i,stop=i+k)

print(Shingle[i])

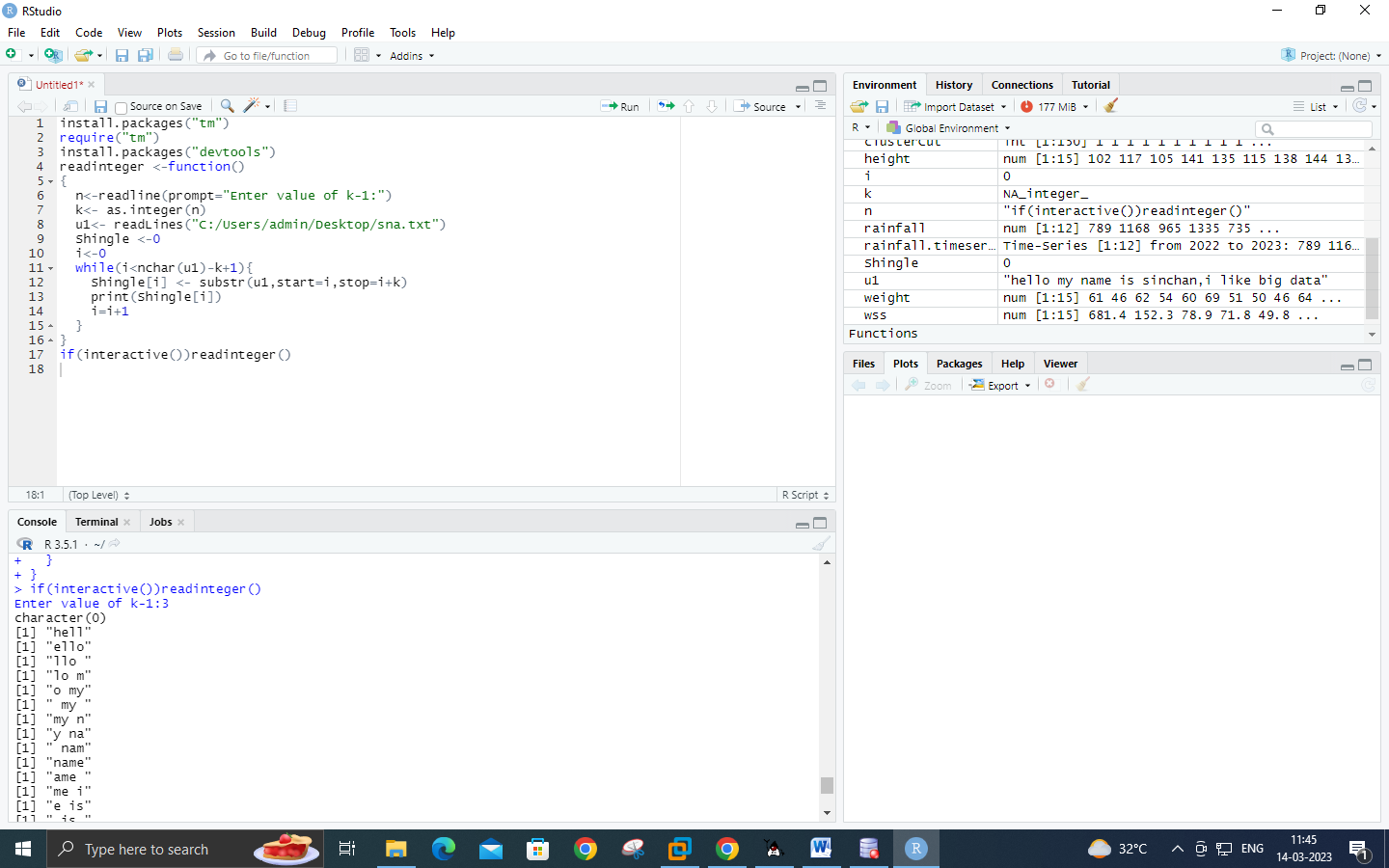
i=i+1

}

}

if(interactive())readinteger()

output:



**Practical-7**:- Write a program for measuring similarity among documents and detecting passages which have been reused.

install.packages("tm")

require("tm")

install.packages("devtools")

my.corpus <- Corpus(DirSource("C:/MSC Notes/r-corpus"))

my.corpus<- tm\_map(my.corpus, removeWords ,stopwords("english"))

my.tdm<- TermDocumentMatrix(my.corpus)

my.dtm<- DocumentTermMatrix(my.corpus,control=list(weighting= weightTfIdf ,stopwords=TRUE))

my.df<- as.data.frame(inspect(my.tdm))

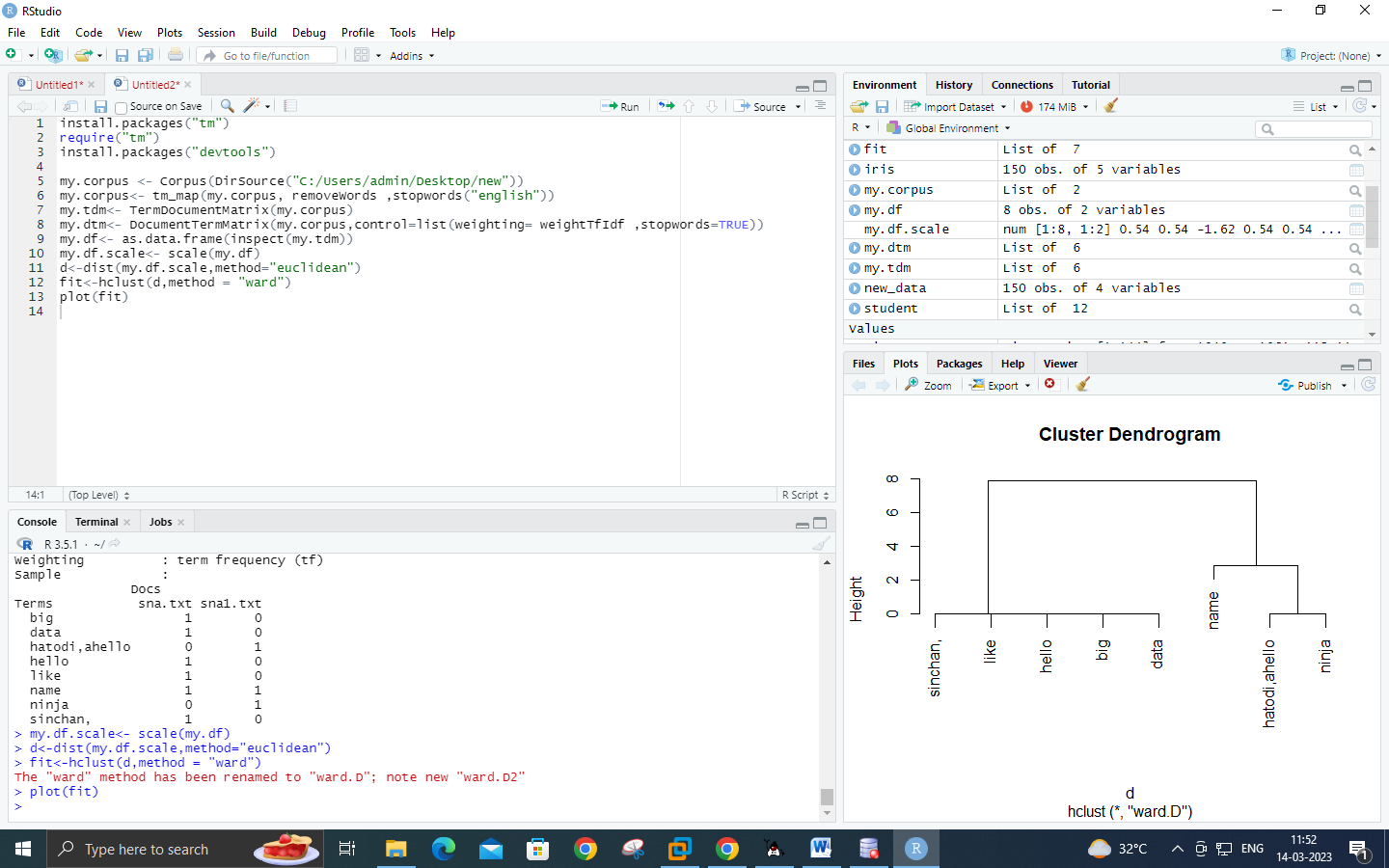
my.df.scale<- scale(my.df)

d<-dist(my.df.scale,method="euclidean")

fit<-hclust(d,method = "ward")

plot(fit)

output:

****

**Practical-8:-** Write a program to compute the **n-moment** for a given stream where n is given.

import java.io.\*;

import java.util.\*;

public class n\_moment

{

public static void main(String args[]) {

int n=15;

String stream[]= {"a","b","c","b","d","a","c","d","a","b","d","c","a","a","b"};

int zero\_moment=0,first\_moment=0,second\_moment=0,count=1,flag=0;

ArrayList<Integer> arrlist=new ArrayList();

System.out.println("Arraylist elements are::");

for (int i=0;i<15;i++)

{

System.out.println(stream[i]+" ");

}

Arrays.sort(stream);

for(int i=1;i<n;i++)

{

if(stream[i]==stream[i-1])

{

count++;

}

else

{

//System.out.println("Hello"+i);

arrlist.add(count);

count=1;

}

}

arrlist.add(count);

zero\_moment=arrlist.size();

System.out.println("\n\n\nValue of Zeroth moment for given stream::"+zero\_moment);

for(int i=0;i<arrlist.size();i++)

{

first\_moment+=arrlist.get(i);

}

System.out.println("\n\nValue of First moment for given stream::"+first\_moment);

for (int i=0;i<arrlist.size();i++)

{

int j=arrlist.get(i);

second\_moment+=(j\*j);

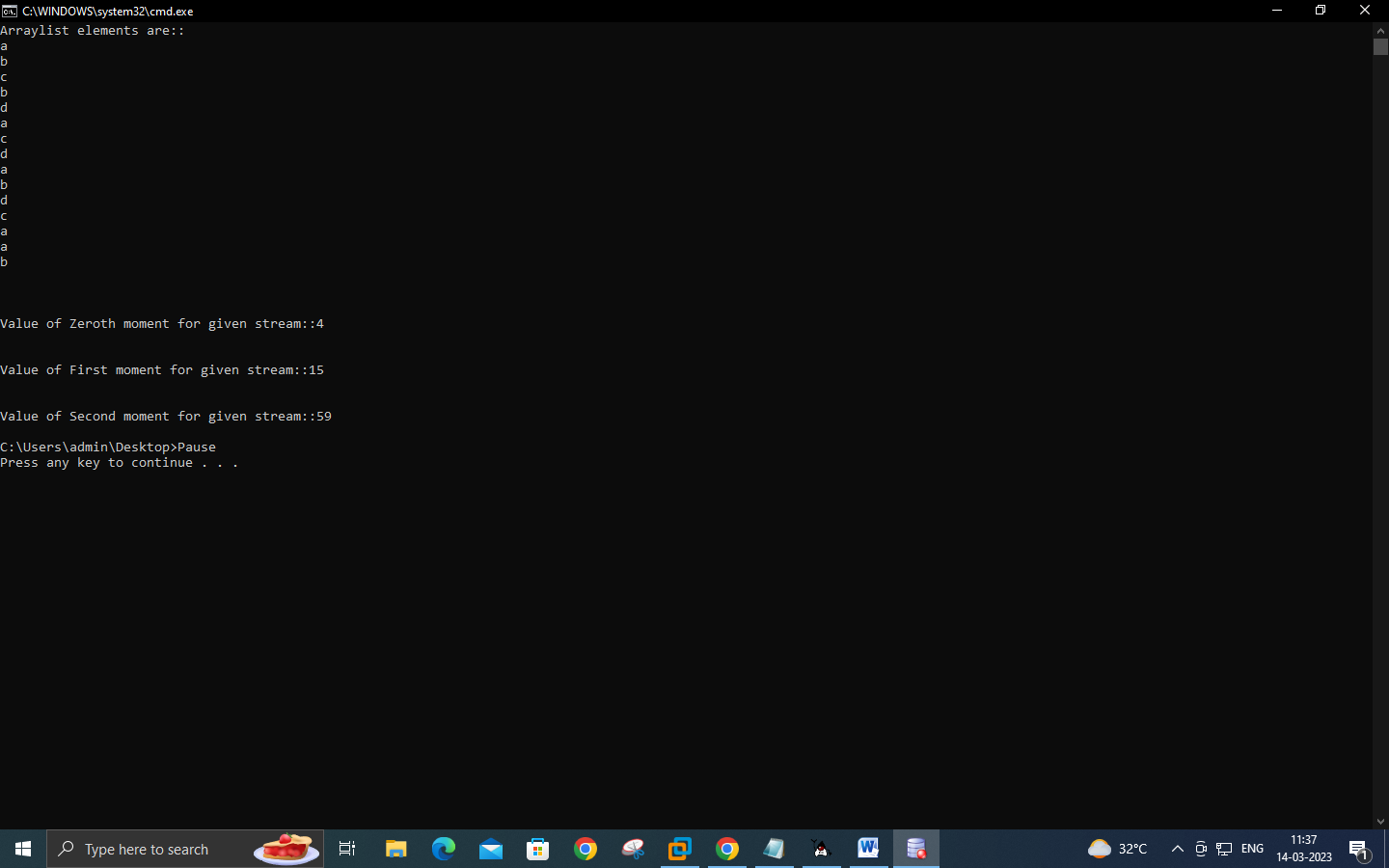
}

System.out.println("\n\nValue of Second moment for given stream::"+second\_moment);

}

}

Output:



**Practical-9:-** Write a program to demonstrate the ALON-Matias-Szegedy Algoithm for second moments.

import java.io.\*;

import java.util.\*;

class AMSA

{

public static int findCharCount(String stream,char XE,int random,int n)

{

int countoccurance=0;

for(int i=random;i<n;i++)

{

if(stream.charAt(i)==XE)

{

countoccurance++;

}

}

return countoccurance;

}

public static int estimateValue(int XV1,int n)

{

int ExpValue;

ExpValue=n\*(2\*XV1-1);

return ExpValue;

}

public static void main(String args[])

{

int n=15;

String stream="abcbdacdabdcaab";

int random1=3,random2=8,random3=13;

char XE1,XE2,XE3;

int XV1,XV2,XV3;

int ExpValuXE1,ExpValuXE2,ExpValuXE3;

int apprSecondMomentValue;

XE1=stream.charAt(random1-1);

XE2=stream.charAt(random2-1);

XE3=stream.charAt(random3-1);

XV1=findCharCount(stream,XE1,random1-1,n);

XV2=findCharCount(stream,XE2,random2-1,n);

XV3=findCharCount(stream,XE3,random3-1,n);

System.out.println(XE1+"="+XV1+" "+XE2+"="+XV2+" "+XE3+"="+XV3);

ExpValuXE1=estimateValue(XV1,n);

ExpValuXE2=estimateValue(XV2,n);

ExpValuXE3=estimateValue(XV3,n);

System.out.println("Expected value for"+XE1+" is::"+ExpValuXE1);

System.out.println("Expected value for"+XE2+" is::"+ExpValuXE2);

System.out.println("Expected value for"+XE3+" is::"+ExpValuXE3);

apprSecondMomentValue=(ExpValuXE1+ExpValuXE2+ExpValuXE3)/3;

System.out.println("approximate second moment value using alon-matis-szegedy is::"+apprSecondMomentValue);

}

}

Output:

