Name : Shubham Lad

Subject: Business Intelligence & Big Data Analytics

Class: MSC-I (Computer Science) Semester 2 Exam Seat no: 512

Academic Year: 2022-2023

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **INDEX** |  |  |
| **No** | **Date** | **TITLE** | **Page**  **No** | **SIGN** |
| 1 | 06/01/23 | Installing and setting environment variables for Working with Apache Hadoop. |  |  |
| 2 | 06/01/23 | Implementing Map-Reduce Program for Word Count problem. |  |  |
| 3 | 19/01/23 | Write a Pig Script for solving counting problems. |  |  |
| 4 | 20/01/23 | Install HBase and use the HBase Data model Store and retrieve data. |  |  |
| 5 | 20/01/23 | Install Hive and use Hive Create and store structured databases. |  |  |
| 6 | 25/01/23 | Write a program to construct different types of k-shingles for a given document. |  |  |
| 7 | 25/01/23 | Write a program for measuring similarity among documents and detecting passages which have been reused. |  |  |
| 8 | 02/02/23 | Write a program to compute the n- moment for a given stream where n is given. |  |  |
| 9 | 02/02/23 | Write a program to demonstrate the Alon-Matias-Szegedy Algorithm for second moments. |  |  |

Practical 01

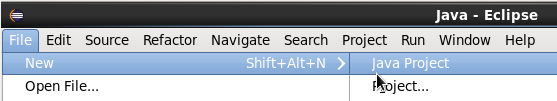
Aim : Installing and setting environment variables for Working with Apache Hadoop

Practical 02

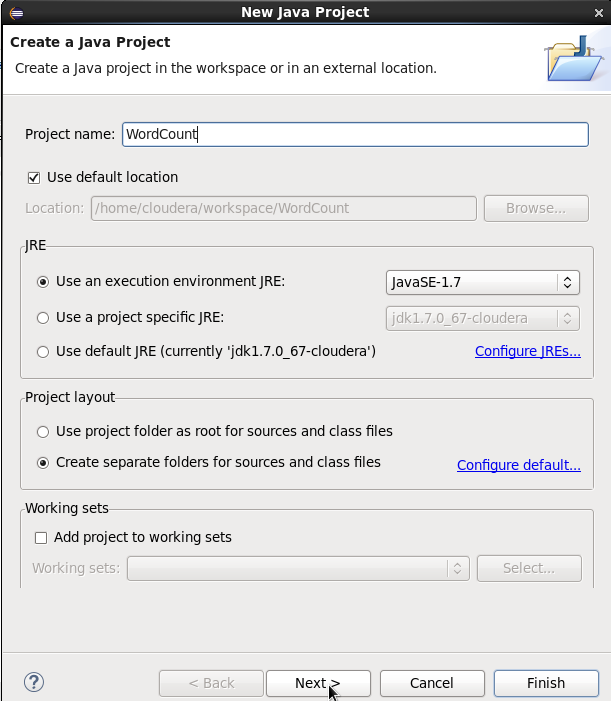
Aim: Implementing Map-Reduce Program for Word Count problem.

Steps:

1. Goto Eclipse in Cloudera and creating a new Java project.



1. Now under project name enter “WordCount” and click on next.

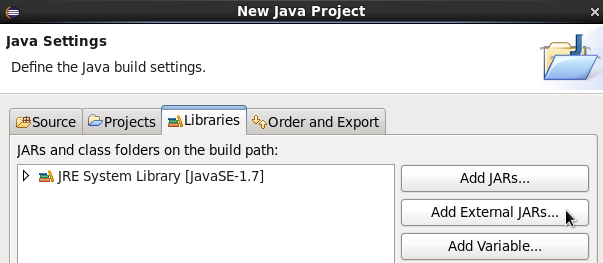


1. Now under “Add External JARs” add all the jar files present under the following locations:

File System/usr/lib/hadoop

File System/usr/lib/hadoop/client

File System/usr/lib/hadoop/client-0.20



After adding all the JAR files click on Finish.

1. Now create a new Java class “WordCount.java” and add the following lines of code in it.

import java.io.IOException;

import java.util.StringTokenizer;

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.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

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

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

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(Object key, Text value, Context context

                ) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {

word.set(itr.nextToken());

context.write(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new IntWritable();

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

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);

job.setMapperClass(TokenizerMapper.class);

job.setCombinerClass(IntSumReducer.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

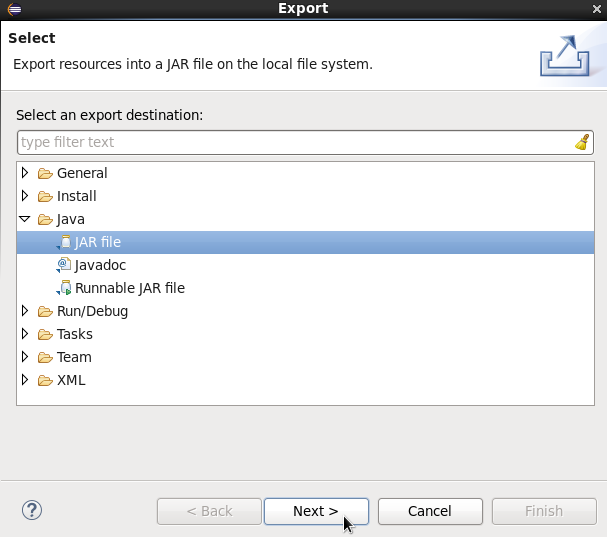
FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

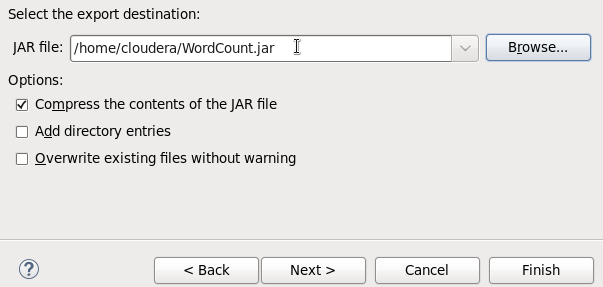
}

}

1. Now right-click on the project and under Export select Jar file and click on next



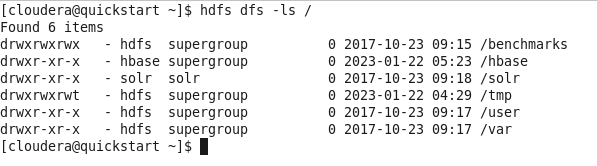
1. Now give the same file destination and file name as shown below and click on Finish



1. Perform the following commands by opening a new terminal.

Commands:

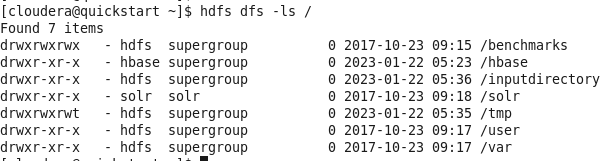
$ hdfs dfs -ls /



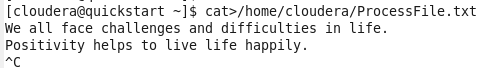
$ sudo -u hdfs hadoop fs -mkdir /inputdirectory



$ hdfs dfs -ls /



$ cat>/home/cloudera/ProcessFile.txt



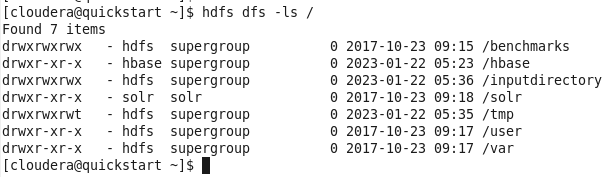
$ cat /home/cloudera/ProcessFile.txt



$ sudo -u hdfs hadoop fs -chmod -R 777 /inputdirectory



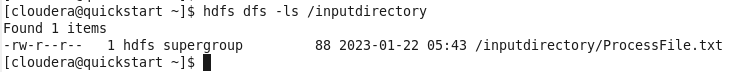
$ hdfs dfs -ls /



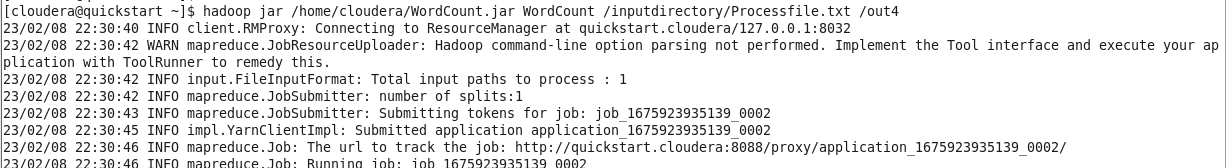
$ sudo -u hdfs hadoop fs -put /home/cloudera/ProcessFile.txt /inputdirectory

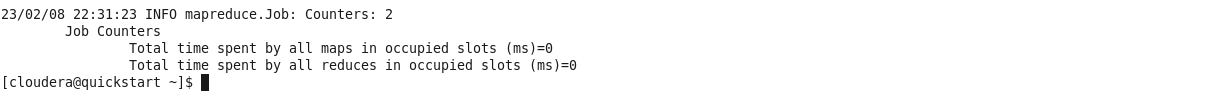


$ hdfs dfs -ls /inputdirectory

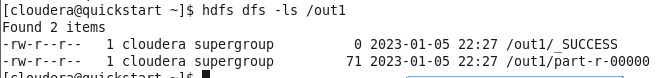


$ hadoop jar /home/cloudera/WordCount.jar WordCount /inputdirectory/ProcessFile.txt /out1





$ hdfs dfs -ls /out1



$ hdfs dfs -cat /out1/part-r-00000



Practical 03

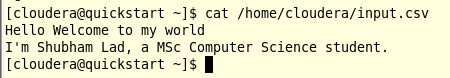
**Aim**: Write a Pig Script for solving counting problems.

**Commands**:

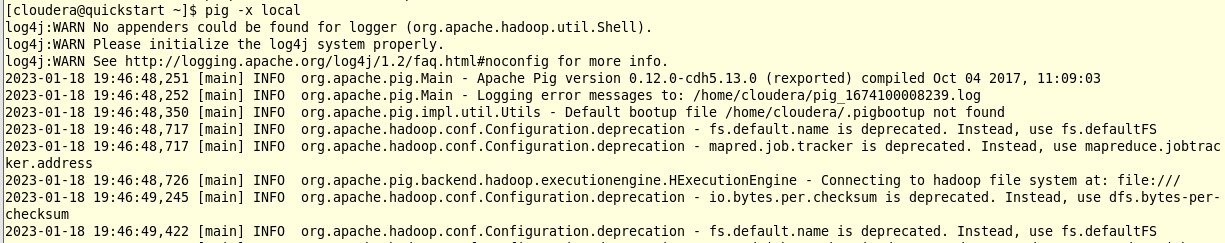
$ cat>/home/cloudera/input.csv

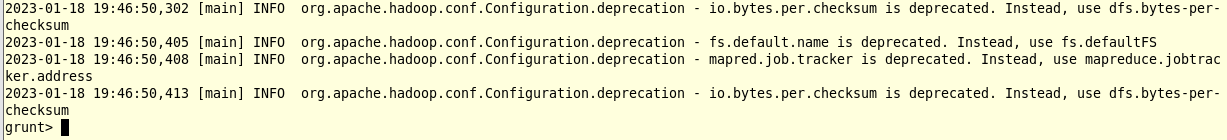


$ cat /home/cloudera/input.csv



$ pig –x local





Grunt shell is Pig’s interactive shell which is used to execute all Pig’s scripts.

Scripting Code:

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



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



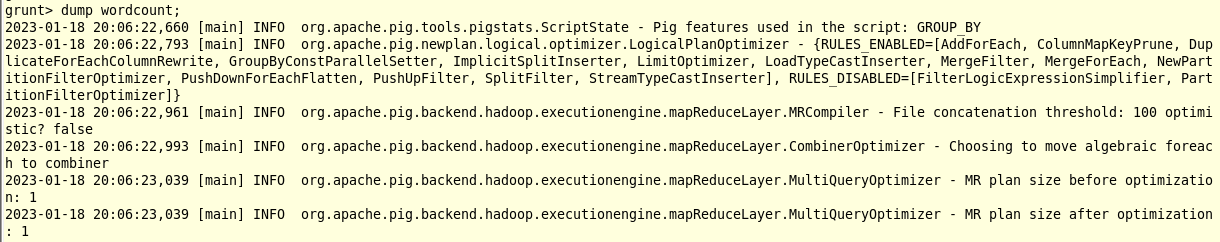
$ grouped = GROUP words by woed;



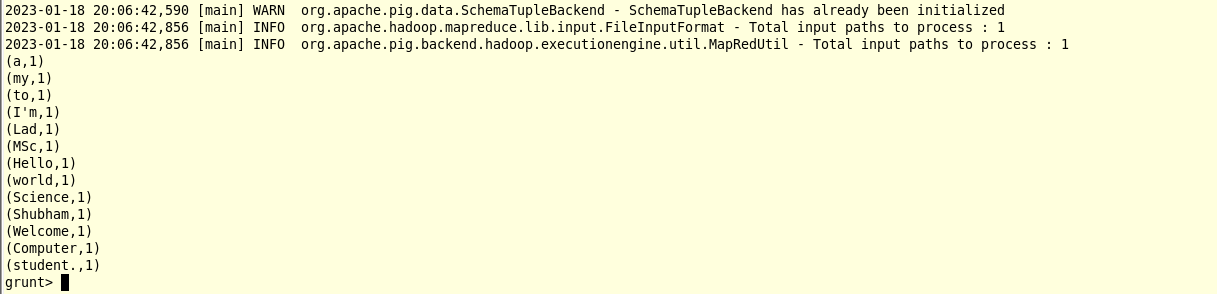
$ wordcount = foreach grouped GENERATE group, COUNT(words);



$ dump wordcount;





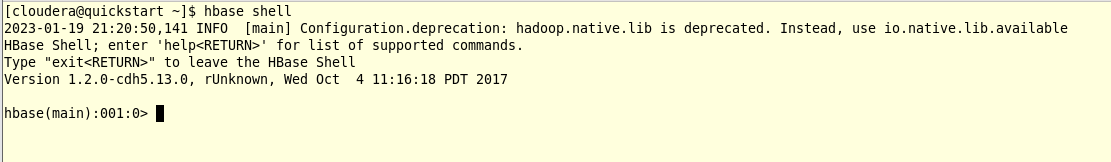


Practical 04

**Aim**: Install HBase and use the HBase Data Model store and retrieve data.

# Start HBase

$ hbase shell



//HBase Commands

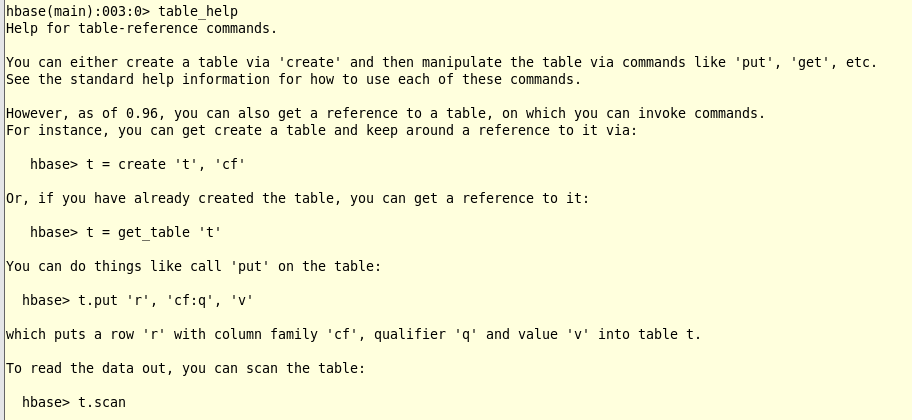
$ status



$ version



$ table\_help

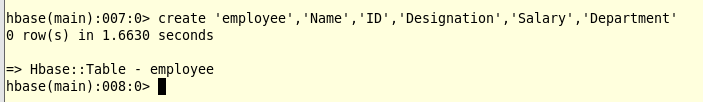


$ whoami



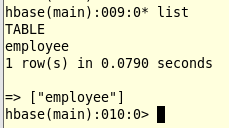
//Data definition Language

$ create 'employee','Name','ID','Designation','Salary','Department'



// Verify created table

$ list

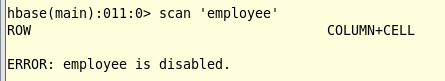


// Disable single table

$ disable ‘employee’



$ scan ‘employee’

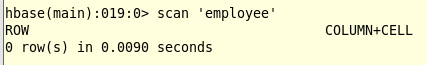


// Enabling table

$ enable ‘employee’

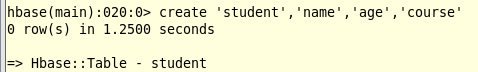


$ scan ‘employee’



//To insert record

$ create 'student','name','age','course'



$ put 'student','shubham','name:fullname','shubham lad'



$ put 'student','shubham','age:presentage','21'



$ put 'student','shubham','course:pursuing','Hadoop'



$ put 'student','mayuresh','name:fullname','mayuresh mhatre'



$ put 'student','mayuresh','age:presentage','25'

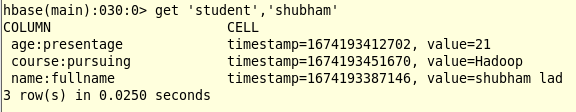


$ put 'student','mayuresh','course:pursuing','Hadoop'

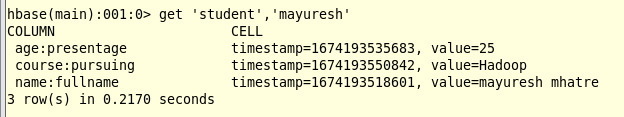


// Get Information

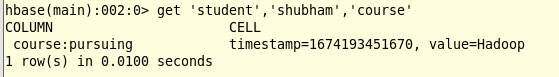
$ get 'student','shubham'



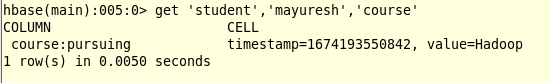
$ get 'student','mayuresh'



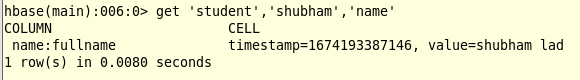
$ get ‘student’,‘shubham’,‘course’



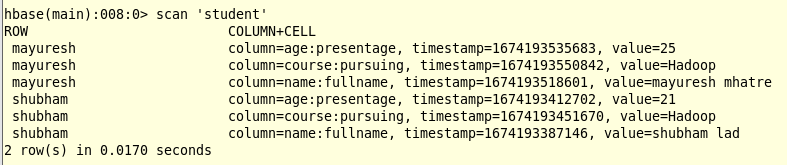
$ get ‘student’,‘mayuresh’,‘course’



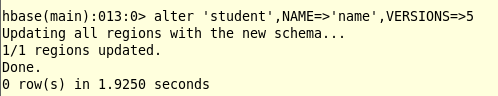
$ get ‘student’,‘shubham’,‘name’



$ scan ‘student’



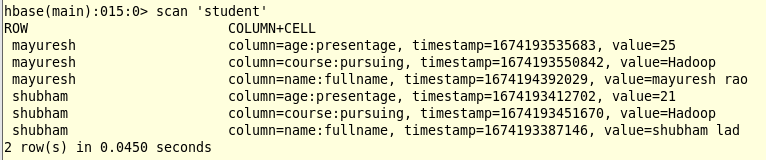
$ alter ‘student’,NAME=>‘name’,VERSIONS=>5



$ put ‘student’,’mayuresh’,‘name:fullname’,‘mayuresh rao’



$ scan ‘student’

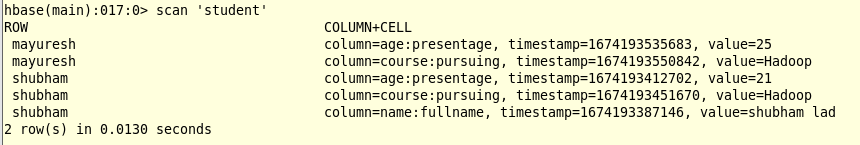


# Deletion of record

$ delete 'student','mayuresh','name:fullname'



$ scan ‘student’



Practical 05

**Aim**: Install Hive and use Hive Create and store structured databases

**Steps**:

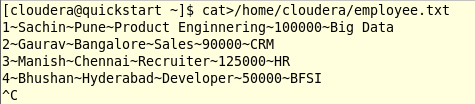
$ cat>/home/cloudera/employee.txt

1~Sachin~Pune~Product Enginnering~100000~Big Data

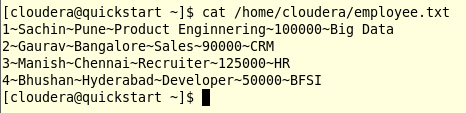
2~Gaurav~Bangalore~Sales~90000~CRM

3~Manish~Chennai~Recruiter~125000~HR

4~Bhushan~Hyderabad~Developer~50000~BFSI



$ cat /home/cloudera/employee.txt



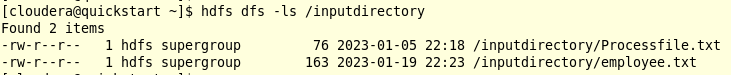
$ sudo -u hdfs hadoop fs -put /home/cloudera/employee.txt /inputdirectory



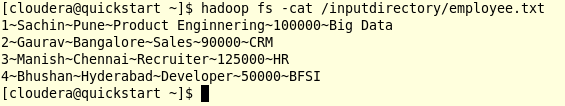
$ hdfs dfs –ls



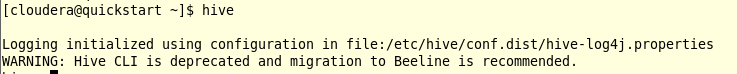
$ hdfs dfs -ls /inputdirectory



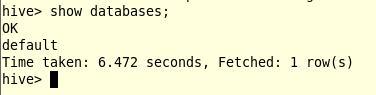
$ hadoop fs -cat /inputdirectory/employee.txt



$ hive



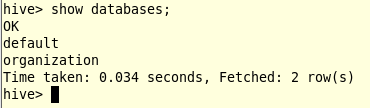
$ show databases;



$ create database organization;



$ show databases;



$ use organization;



$ show tables;



$ create table employee(

> id int,

> name string,

> city string,

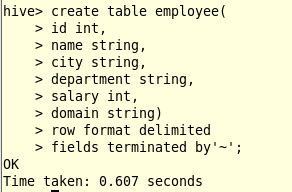
> department string,

> salary int,

> domain string)

> row format delimited

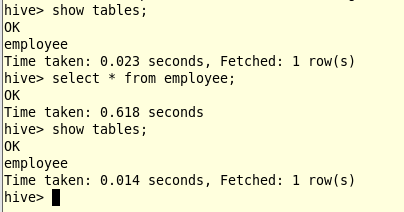
> fields terminated by'~';



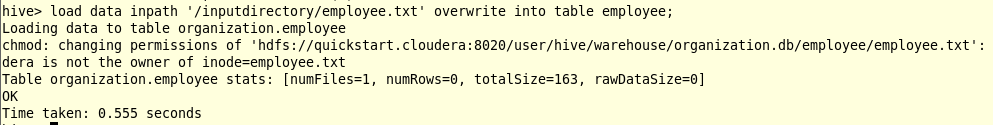
$ show tables;

$ select \* from employee;

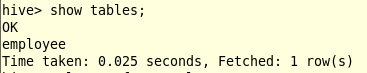
$ show tables;



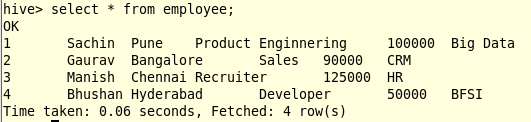
$ load data inpath '/inputdirectory/employee.txt' overwrite into table employee;



$ show tables;



$ select \* from employee;



Practical 06

**Aim**: Write a program to construct different types of shingles for given document.

Commands for installation of required packages before executing program:

> install.packages("tm")

> require("tm")

> install.packages("devtools")

Code:

readinteger <- function()

{

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

k<-as.integer(n)

u1 <- readLines("File1.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()

File1.txt:

Life is not a problem to be solved, but a reality to be experienced.

Output:

Enter value of k-1: 5

character(0)

[1] "Life i"

[1] "ife is"

[1] "fe is "

[1] "e is n"

[1] " is no"

[1] "is not"

[1] "s not "

[1] " not a"

[1] "not a "

[1] "ot a p"

[1] "t a pr"

[1] " a pro"

[1] "a prob"

[1] " probl"

[1] "proble"

[1] "roblem"

[1] "oblem "

[1] "blem t"

[1] "lem to"

[1] "em to "

[1] "m to b"

[1] " to be"

[1] "to be "

[1] "o be s"

[1] " be so"

[1] "be sol"

[1] "e solv"

[1] " solve"

[1] "solved"

[1] "olved,"

[1] "lved, "

[1] "ved, b"

[1] "ed, bu"

[1] "d, but"

[1] ", but "

[1] " but a"

[1] "but a "

[1] "ut a r"

[1] "t a re"

[1] " a rea"

[1] "a real"

[1] " reali"

[1] "realit"

[1] "eality"

[1] "ality "

[1] "lity t"

[1] "ity to"

[1] "ty to "

[1] "y to b"

[1] " to be"

[1] "to be "

[1] "o be e"

[1] " be ex"

[1] "be exp"

[1] "e expe"

[1] " exper"

[1] "experi"

[1] "xperie"

[1] "perien"

[1] "erienc"

[1] "rience"

[1] "ienced"

[1] "enced."

Practical 07

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

Commands for installation of required packages before executing program:

> install.packages("tm")

> require("tm")

> install.packages("devtools")

> install.packages(“ggplot2”)

> install.packages(“textreuse”)

**Code**:

> my.corpus <- Corpus(DirSource("C:/Users/shubh/OneDrive/Documents/RStudio"))

> 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))

<<TermDocumentMatrix (terms: 21, documents: 3)>>

Non-/sparse entries: 22/41

Sparsity : 65%

Maximal term length: 12

Weighting : term frequency (tf)

Sample :

Docs

Terms File1.txt File2.txt File3.txt

experienced. 1 0 0

goal, 0 1 0

happy 0 1 0

life 1 0 1

life, 0 1 0

live 0 1 0

people 0 1 0

problem 1 0 0

reality 1 0 0

solved, 1 0 0

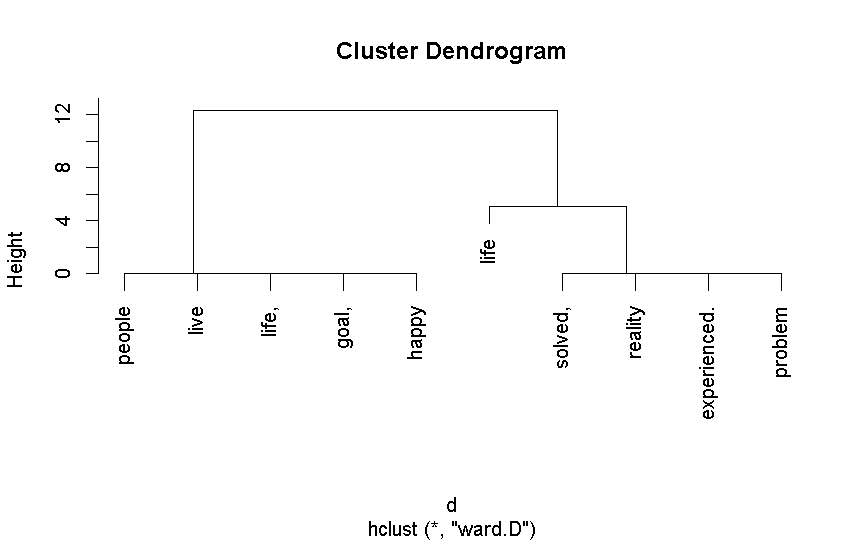
> my.df.scale <- scale(my.df)

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

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

> plot(fit)

**Output**:



Practical 08

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

**Code:**

package moment;

import java.util.ArrayList;

import java.util.Arrays;

public class Moment {

public static void main(String[] args) {

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

int n=15;

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 {

arrlist.add(count);

count=1;

}

}

arrlist.add(count);

zero\_moment=arrlist.size();

System.out.println("\nZeroth moment for the given stream is: "+zero\_moment);

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

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

}

System.out.println("\nFirst moment for the given stream is: "+first\_moment);

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

int j=arrlist.get(i);

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

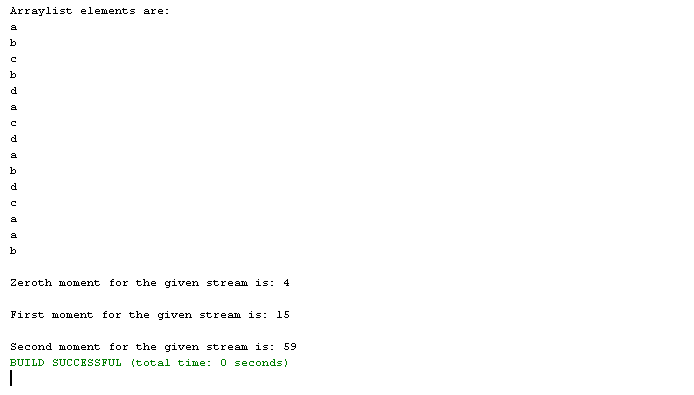
}

System.out.println("\nSecond moment for the given stream is: "+second\_moment);

}

}

**Output**:



Practical 09

**Aim**: Write a program to demonstrate the Alon-Matias-Szegedy Algorithm for second moments.

**Code**:

package amsa;

/\*\*

\*

\* @author shubham

\*/

public class AMSA {

/\*\*

\* @param stream

\* @param XE

\* @param random

\* @param n

\* @return

\*/

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) {

System.out.println("Alon-Matias-Szegedy Algorithm for second moments");

String stream="abcbdacdabdcaab";

int n = stream.length();

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+"\n"+XE2+"="+XV2+"\n"+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-Matia-Szegedy is :: "+apprSecondMomentValue);

}

}

**Output:**

