**APPLIED BIG DATA ANALYTICS THEORY**

SUBMITTED BY

 **SHUBHAM PRAVIN KALE**

**2023-2024**

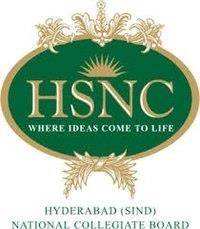
**HSNC UNIVERSITY**

MASTERS OF SCIENCE IN INFORMATION TECHNOLOGY KISHINCHAND CHELLARAM COLLEGE

D.W.ROAD, CHURCHGATE, MUMBAI–400 020.

**SUBJECT CODE - MS-SIT-3P8**

APPLIED BIG DATA ANALYTICS THEORY

**KISHINCHAND CHELLARAM COLLEGE**

CHURCHGATE, MUMBAI – 400 020.

**DEPARTMENT OF INFORMATION TECHNOLOGY M.SC. PART- II**

**CERTIFICATE**

This is to certify that the practical done at **K.C. College** by

# MR/MS. Shubham . Pravin. Kale

## **(**Seat No: KFMSCIT015 ) in partial fulfillment for M.SC. (I.T.) Degree Examination has been found satisfactory. This Practical journal had not been submitted for any other examination and does not form part of any other course undergone by the candidate.

|  |  |  |
| --- | --- | --- |
| **Signature** | **Signature** | **Signature** |
| **Lecturer-In-Charge** | **External Examiner** | **Course Coordination** |
| **Guided By** | **Examined By** | **Certified By** |

**College Stamp**

# INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Practical Names** | | **Sign** |
| 01. | Apriori Algorithm | |  |
| 02. | Write a python program to pick the content for billboards from the big data. | |  |
| A) | Using Python with CSV |
| B) | Using Python with Static Data |
| C) | Using R with CSV |
| D) | Using R with Static Data |
| 03. | Implement an application that stores big data in Mongo DB and manipulate  using R and Python. | |  |
| A) | Manipulate using R |
|  | B) | Manipulate using Python |  |
| 04. | A) | Analyzing Instagram App Reviews |  |
| B) | Data visualization of social media post with the help of word cloud |  |
| 05. | Data visualization using pygal | |  |
| 06. | Installation of Apache Cassandra | |  |
| 07. | A) | Sentimental Analysis in R |  |
| B) | Sentimental Analysis in R to analyze the data and term document  matrix |
| 08. | Database - Create two tables and make primary and foreign key and use  select and where clause | |  |
| 09. | Calculate the sentiment analysis score and visualize the result | |  |
| 10 | A) | Creating data model using Cassandra |  |
|  | B) | Create, Insert, update and display the data from Cassandra using  python |  |

## Practical 01

Aim: Apriori Algorithm Code:

install.packages("arules") #analysis package used in transactional dataset install.packages("arulesViz") #Vizualization install.packages("RColorBrewer") #Coloring the graph

#Loading Libraries library(arules) library(arulesViz) library(RColorBrewer)

#import data("Groceries")

rules<-apriori(Groceries,parameter = list(supp=0.01,conf=0.2))

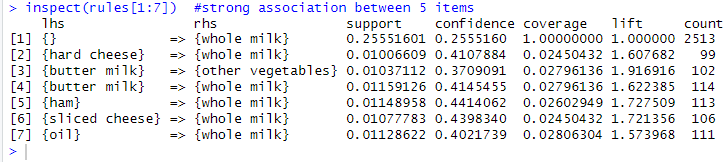
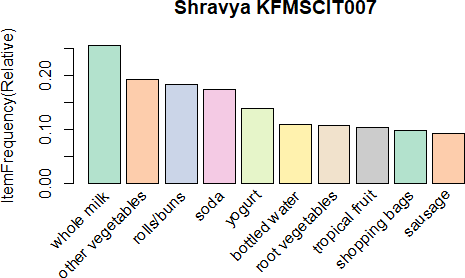
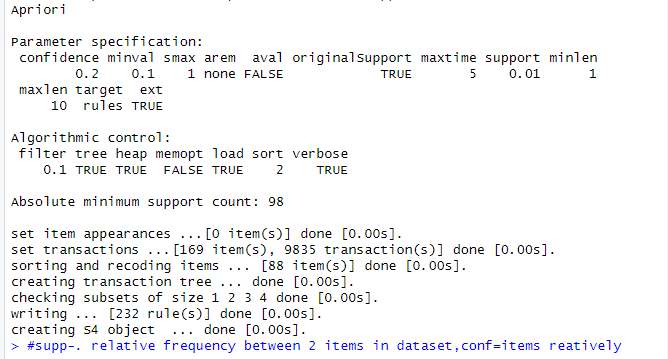
#supp-. relative frequency between 2 items in dataset,conf=items reatively

#using inspect function

inspect(rules[1:7]) #strong association between 7 items #Using itemFrequencyPlot() function arules::itemFrequencyPlot(Groceries,topN=10,

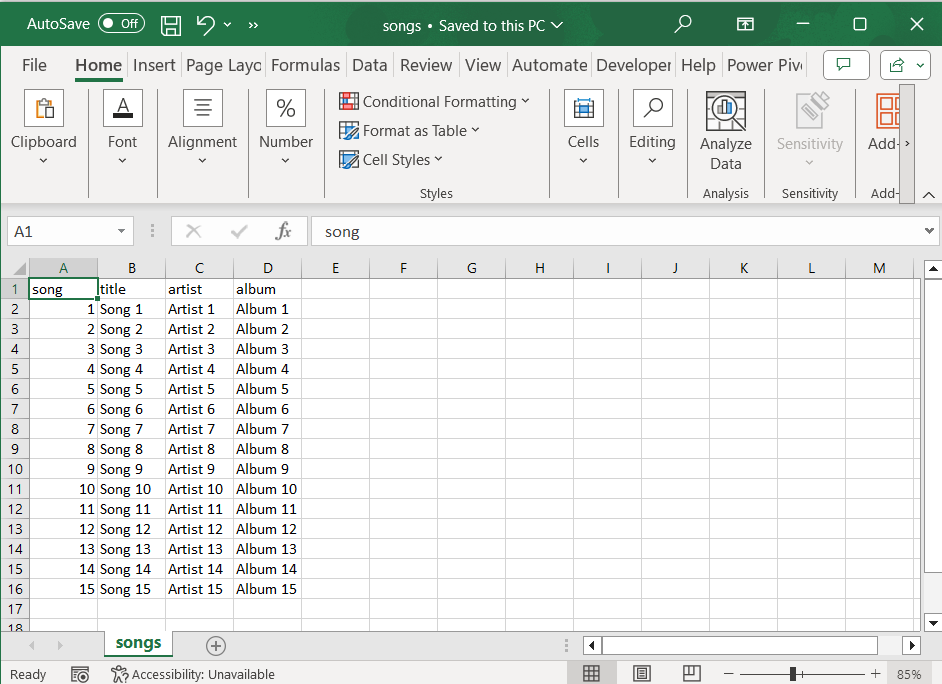
col=brewer.pal(8,'Pastel2'), main="Shravya KFMSCIT007", type="relative", ylab="ItemFrequency(Relative)")

## Output:



Practical 02

## Aim**:** Write a python program to pick the content for billboards from the big data

Songs.csv

## Using Python with CSV Code:

print("Shravya Erabathini,07") import csv

import random

def get\_random\_songs(csv\_file, num\_songs): with open(csv\_file, 'r') as file:

reader = csv.DictReader(file) all\_songs = list(reader)

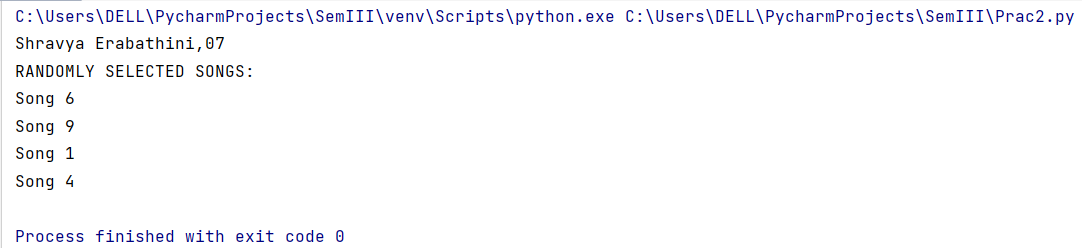
random.shuffle(all\_songs) random\_songs = all\_songs[:num\_songs] return random\_songs

csv\_file = 'songs.csv' num\_songs = 4

random\_songs = get\_random\_songs(csv\_file, num\_songs) print("RANDOMLY SELECTED SONGS:")

for song in random\_songs: print(f"{song['title']}")

Output:



## Using Python with Static Data

Code:

print("Shravya Erabathini,07") import random

def billboard\_songs(songs,num\_songs): random.shuffle(songs) bill\_songs=songs[:num\_songs] return bill\_songs

all\_songs=["song1","song2","song3","song4","song5"] num\_bill\_song=3 bill\_songs=billboard\_songs(all\_songs,num\_bill\_song) print("Billboard songs are: ")

for song in bill\_songs: print(song)

## Output:



1. Using R with Static Data

## Code:

pick\_bill\_songs<-function(songs,num\_songs)

{

shuffle\_songs<-sample(songs)

bill\_songs<-head(shuffle\_songs,num\_songs) return (bill\_songs)

}

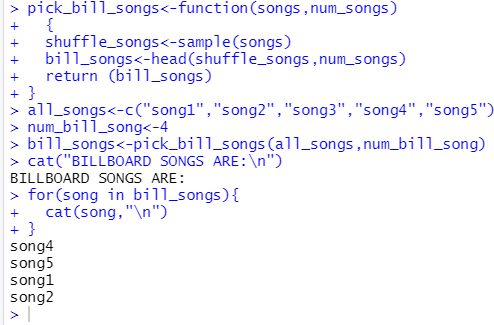
all\_songs<-c("song1","song2","song3","song4","song5") num\_bill\_song<-4

bill\_songs<-pick\_bill\_songs(all\_songs,num\_bill\_song) cat("BILLBOARD SONGS ARE:\n")

for(song in bill\_songs){ cat(song,"\n")

}

## Output:



1. Using R with CSV

## Code:

pick\_bill\_songs<-function(songs,num\_songs)

{

shuffle\_songs<-sample(songs)

bill\_songs<-head(shuffle\_songs,num\_songs) return (bill\_songs)

}

all\_songs <- read.csv("D:\MSC-IT\Sem 3\Big Data Analytics\songs.csv", header = TRUE) print(all\_songs)

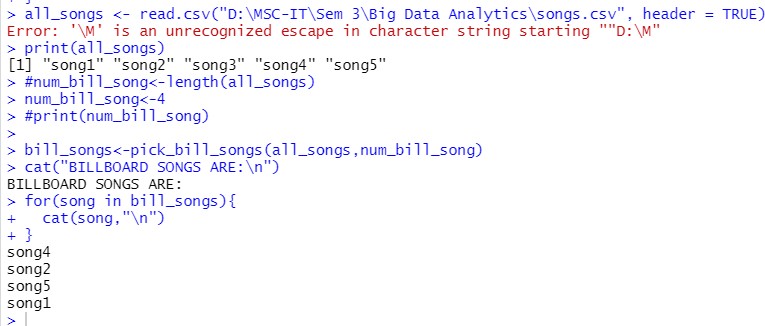
#num\_bill\_song<-length(all\_songs) num\_bill\_song<-4 #print(num\_bill\_song)

bill\_songs<-pick\_bill\_songs(all\_songs,num\_bill\_song) cat("BILLBOARD SONGS ARE:\n")

for(song in bill\_songs){ cat(song,"\n")

}

## Output:



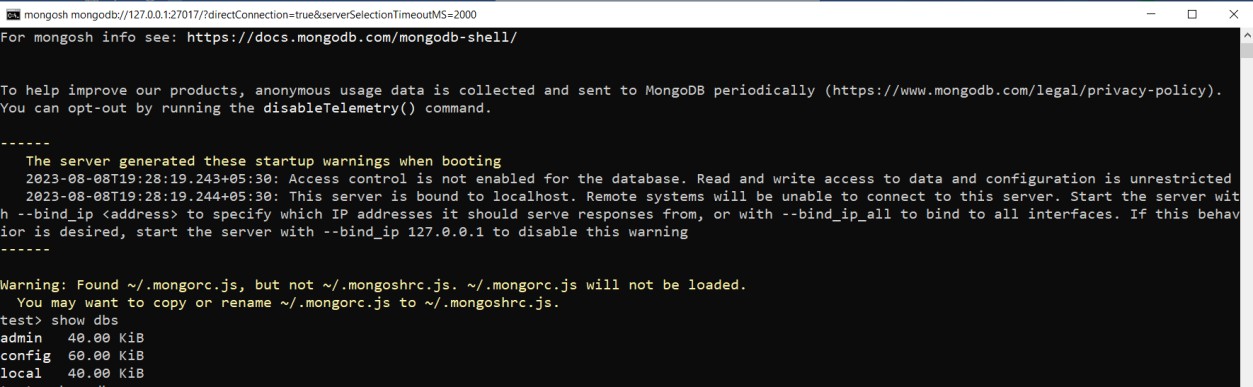
Practical 03

## Aim: Implement an application that stores Big Data in Mongo DB and manipulate using R and Python

Step 1: Check whether server is running using mongod command in the cmd at location till bin of mongodb



Step 2: On another shell enter command **mongosh**

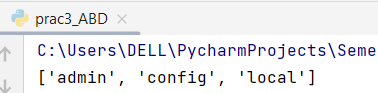


## Using Python

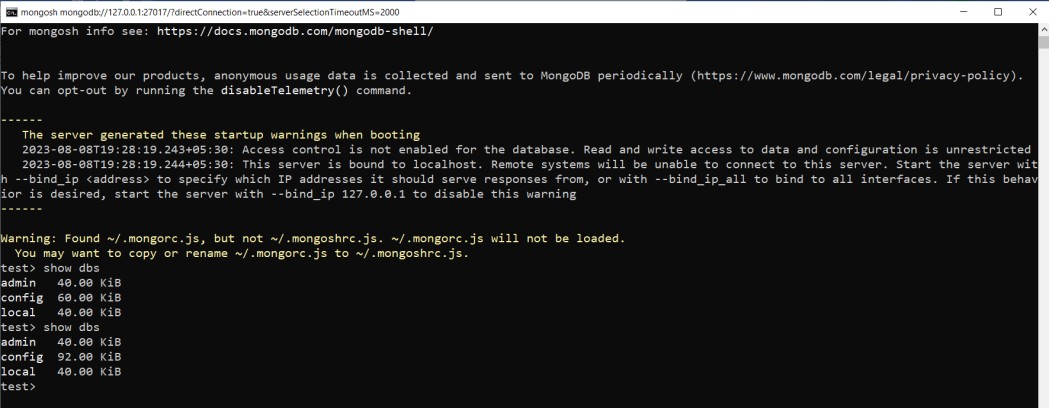
Step 1: Establish a connection with Mongodb Create a database “mydatabase” Code:

import pymongo myclient=pymongo.MongoClient("mongodb://127.0.0.1:27017/") mydb=myclient["mydatabase"]

Output:



Step 2: Verify in the shell whether database is displayed or not.



Step 3: List the database and verify if it exists Code:

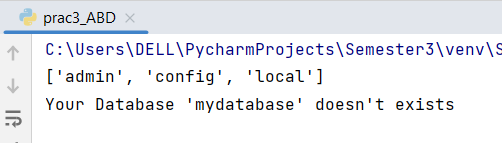
import pymongo myclient=pymongo.MongoClient("mongodb://127.0.0.1:27017/") mydb=myclient["mydatabase"] print(myclient.list\_database\_names()) dblist=myclient.list\_database\_names()

if "mydatabase" in dblist:

print("Your Database 'mydatabase' exist") else:

print("Your Database 'mydatabase' doesn't exists")

Output:



Step 4: Enter One collection at a time Code:

import pymongo

myclient=pymongo.MongoClient("mongodb://127.0.0.1:27017/") mydb=myclient["mydatabase"] print(myclient.list\_database\_names()) dblist=myclient.list\_database\_names()

mycol=mydb["STUDENTS"] *#Collection name*

print("Collection names: ",mydb.list\_collection\_names())

mydict = {"name":"Shravya","Roll-No":"07"} x=mycol.insert\_one(mydict)

print('Id of the inserted record',x.inserted\_id) y=mycol.find\_one()

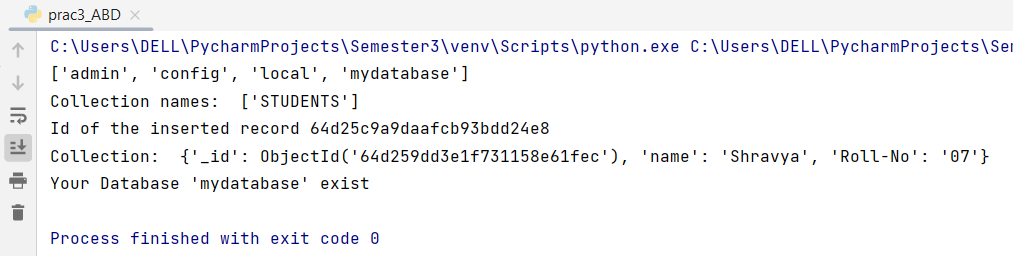
print("Collection: ",y)

if "mydatabase" in dblist:

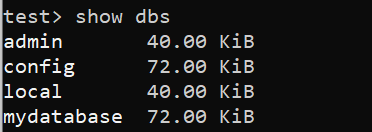
print("Your Database 'mydatabase' exist") else:

print("Your Database 'mydatabase' doesn't exists")

Output:



Note: Once collections created in the database, It starts reflecting the list of database.



Step 5: Enter Multiple collections at a time. Code:

import pymongo myclient=pymongo.MongoClient("mongodb://127.0.0.1:27017/") mydb=myclient["mydatabase"] print(myclient.list\_database\_names()) dblist=myclient.list\_database\_names()

mycol=mydb['STUDENTS'] *#Collection name*

print("Collection names: ",mydb.list\_collection\_names())

mydict = {"name":"Nabila","Roll-No":"16"},{"name":"Sayali","Roll-No":"20"} x=mycol.insert\_many(mydict)

print('Id of the inserted record',x.inserted\_ids)

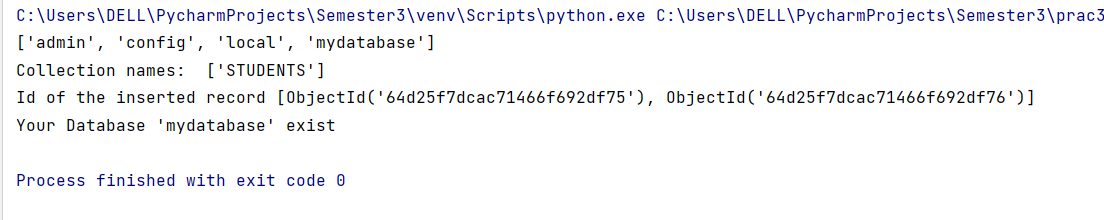
*#for i in mycol.find(): #print("Collections: ",i)*

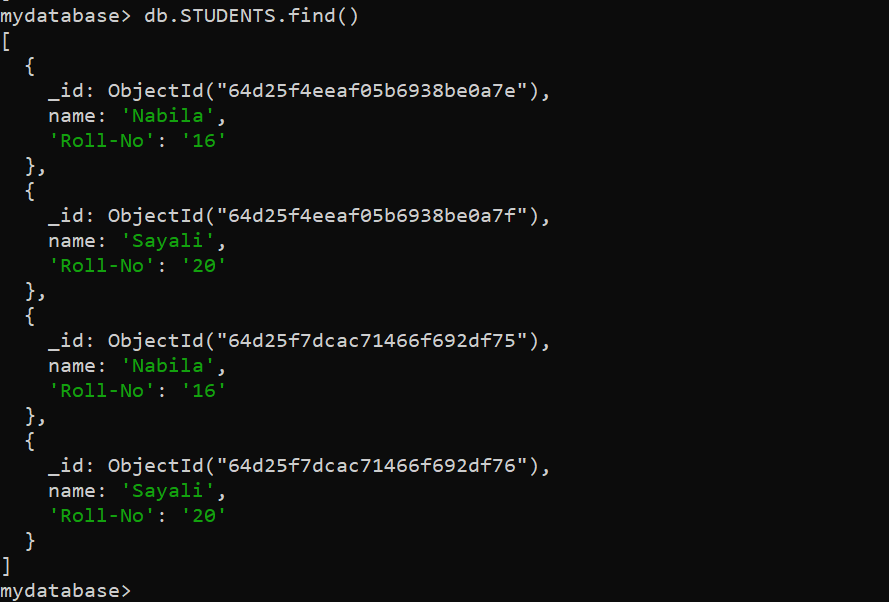
if "mydatabase" in dblist:

print("Your Database 'mydatabase' exist") else:

print("Your Database 'mydatabase' doesn't exists")

Output:



1. Displaying the collections of database in mongo shell
2. Displaying the collections of database in python shell Code:

import pymongo myclient=pymongo.MongoClient(“mongodb://127.0.0.1:27017/”) mydb=myclient[“mydatabase”] print(myclient.list\_database\_names()) dblist=myclient.list\_database\_names()

mycol=mydb[“STUDENTS”] *#Collection name*

print(“Collection names: “,mydb.list\_collection\_names())

mydict = {“name”:”Nabila”,”Roll-No”:”16”},{“name”:”Sayali”,”Roll-No”:”20”} x=mycol.insert\_many(mydict)

print(‘Id of the inserted record’,x.inserted\_ids) for I in mycol.find():

print(“Collections: “,i)

if “mydatabase” in dblist:

print(“Your Database ‘mydatabase’ exist”) else:

print(“Your Database ‘mydatabase’ doesn’t exists”)

Output:



## Using R

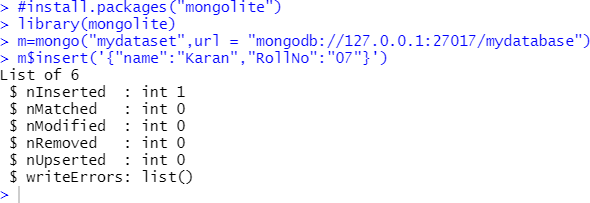
Step 1: Connecting R with Mongodb



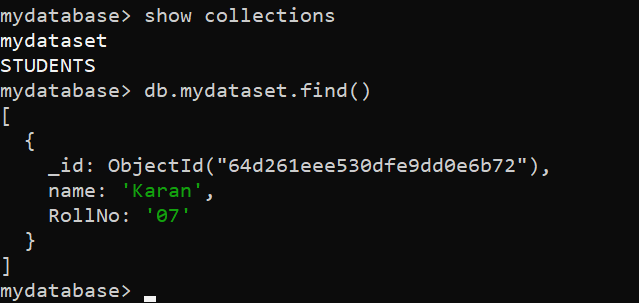
Step 2: Inserting values #install.packages("mongolite") library(mongolite)

m=mongo("mydataset",url = "mongodb://127.0.0.1:27017/mydatabase") m$insert('{"name":"Karan","RollNo":"07"}')

Output:



Step 3: Verifying in console



## Practical 04 4A) Aim: Analyzing instagram app reviews Code:

import pandas as pd import numpy as np

import matplotlib.pyplot as plt import warnings

import seaborn as sns warnings.filterwarnings('ignore')

df=pd.read\_csv("threads\_reviews.csv") print("First 5 rows\n",df.head())

print("last 5 rows\n",df.tail()) print("Shape function:",df.shape) print("Columns:\n",df.columns)

print("Duplicate Sum: ",df.duplicated().sum()) print("Drop Duplicates\n",df.drop\_duplicates()) print("IsNull sum\n",df.isnull().sum()) print("Information:\n",df.info) print("Describe\n",df.describe())

print("Unique values\n",df.nunique())

print("Unique with respect to Source",df['source'].unique()) plt.figure(figsize=(15,7))

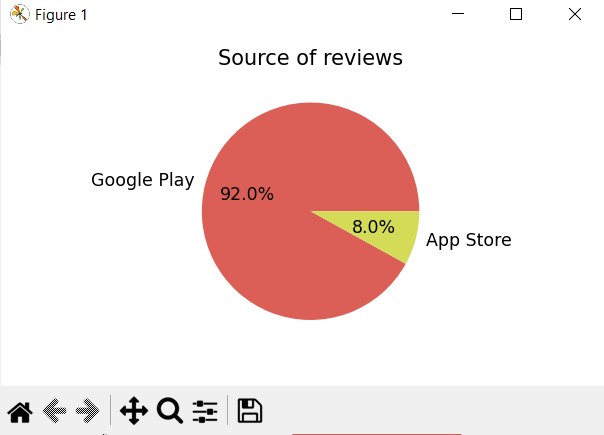
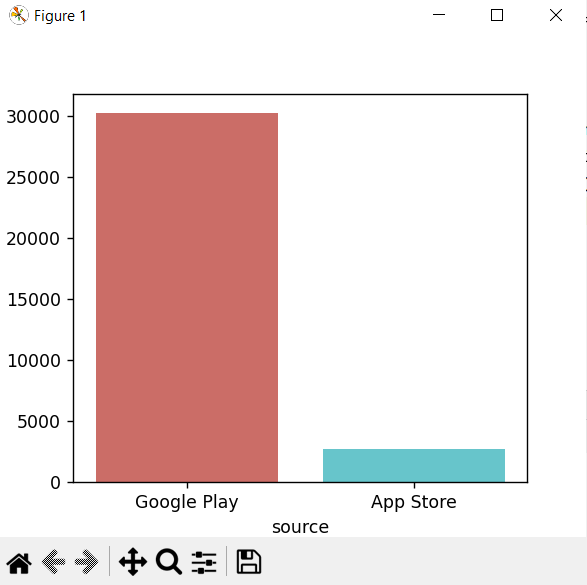
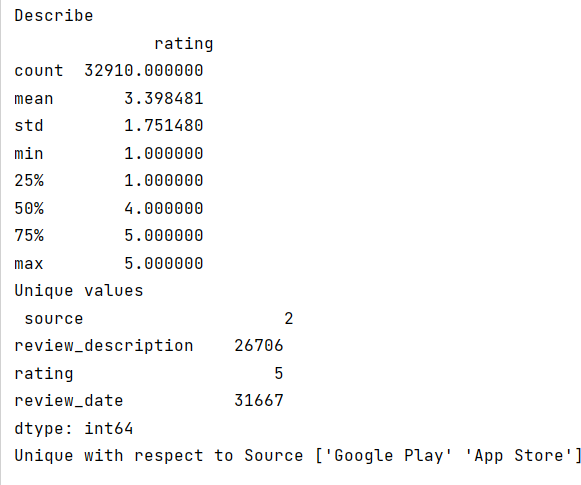
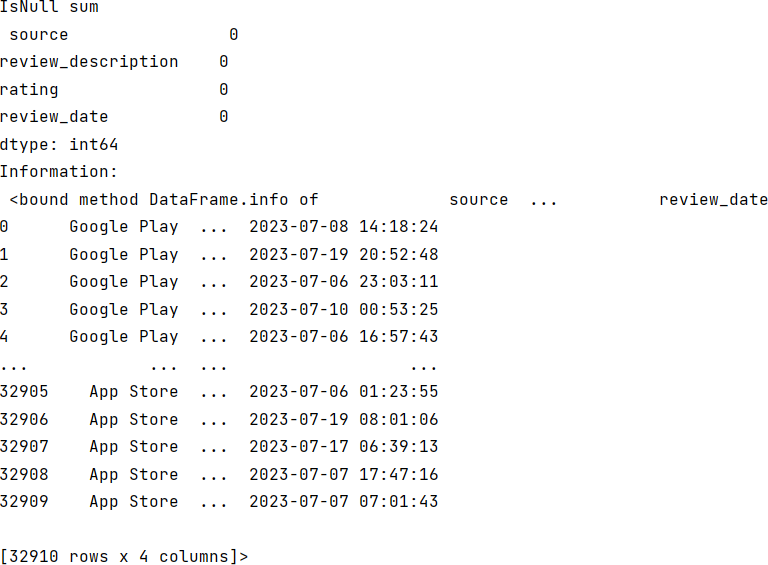
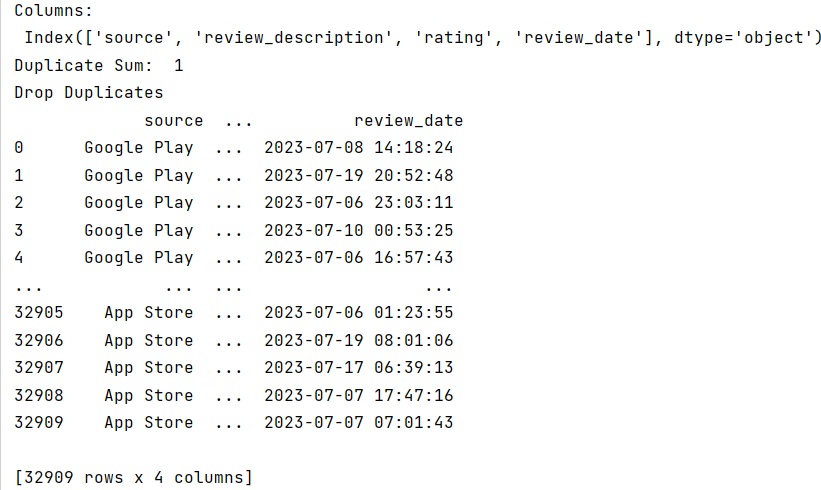
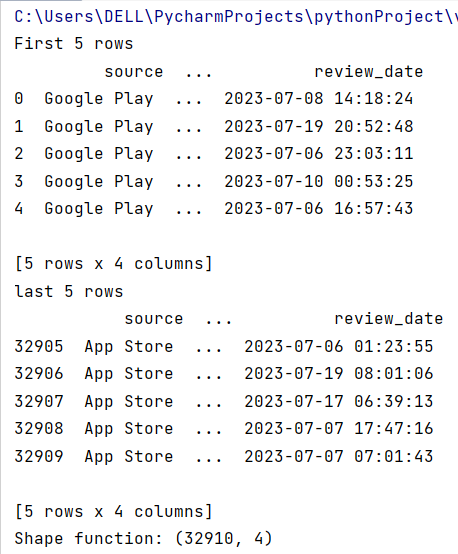
sns.countplot(x='source',data=df,palette='hls') plt.show()

plt.figure(figsize=(12,12)) counts=df['source'].value\_counts()

plt.pie(counts,labels=counts.index,autopct="%1.1f%%",colors=sns.color\_palette('hls')) plt.title("Source of reviews")

plt.show()

## Output:



4B) Aim: Data visualization of any social media post with the help of Word Cloud.

## Code:

import pandas as pd import wordcloud

from wordcloud import WordCloud import matplotlib.pyplot as plt

df = pd.read\_csv("threads\_reviews.csv") df\_new = df[['review\_description', 'rating']] print(df\_new)

def clean\_text(text): text = text.lower() return text.strip()

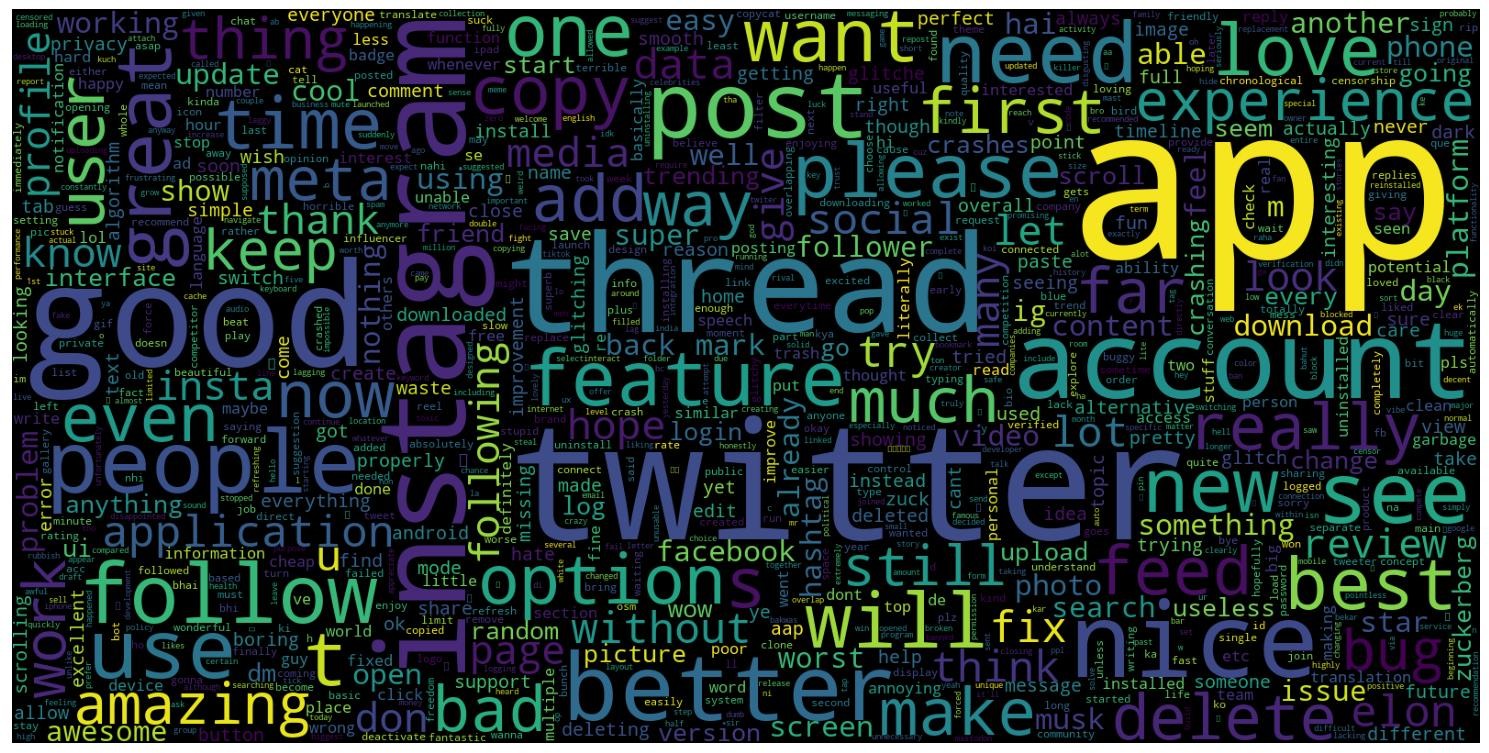
df\_new.review\_description = df\_new.review\_description.apply(lambda x: clean\_text(x)) print(df\_new.review\_description)

data=df\_new.review\_description plt.figure(figsize=(20,20))

wc=WordCloud(max\_words=900,width=1600,height=800,collocations=False).generate(" ".join(data)) plt.imshow(wc)

plt.axis('off') plt.show()

## Output:



Practical 05 Aim**:** Data visualization using pygal

## Code**:** BAR CHART

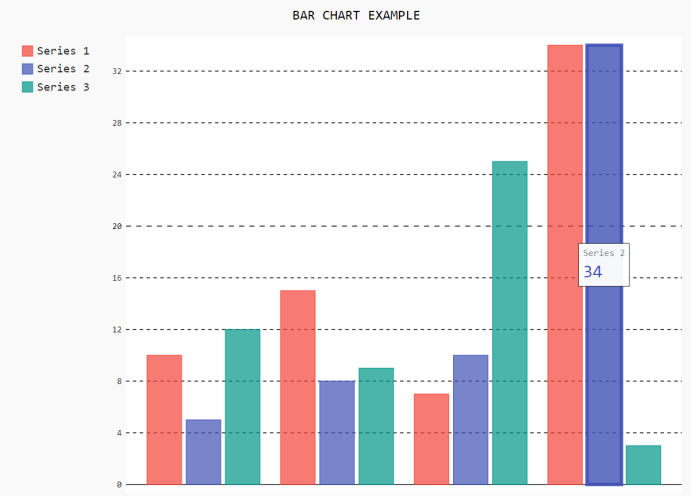
import pygal bar\_chart=pygal.Bar()

bar\_chart.add("Series 1",[10,15,7,35])

bar\_chart.add("Series 2",[5,8,10,34])

bar\_chart.add("Series 3",[12,9,25,3]) bar\_chart.title="BAR CHART EXAMPLE" bar\_chart.render\_to\_file("BAR.svg")

## Output:



Code: LINE CHART

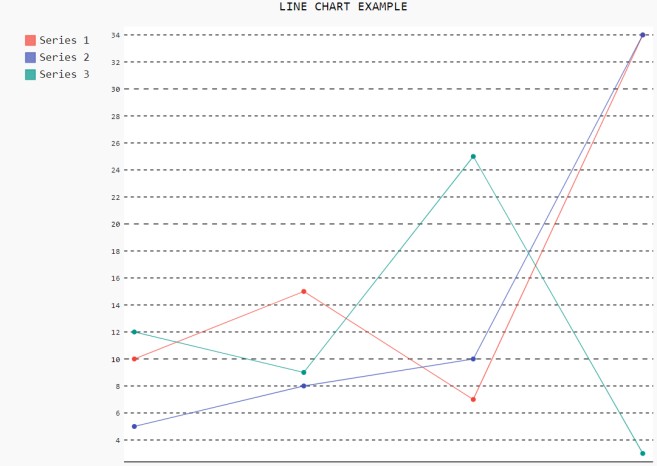
import pygal line\_chart=pygal.Line()

line\_chart.add("Series 1",[10,15,7,35])

line\_chart.add("Series 2",[5,8,10,34])

line\_chart.add("Series 3",[12,9,25,3]) line\_chart.title="LINE CHART EXAMPLE" line\_chart.render\_to\_file("LINE.svg")

## Output:

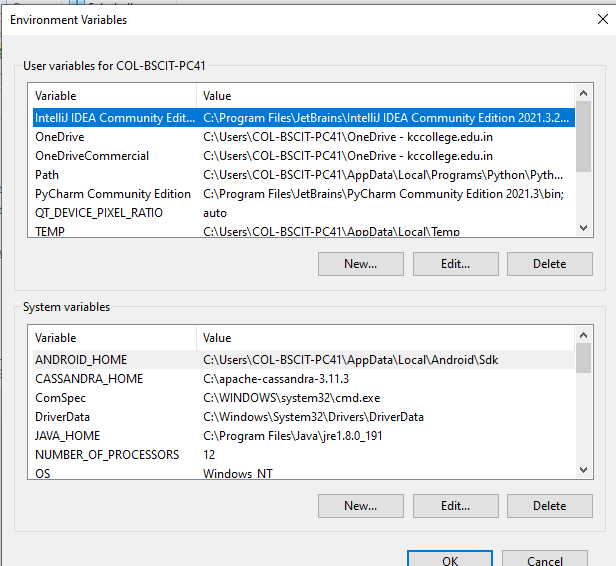


Practical 06 Aim**:** Installation of Apache Cassandra

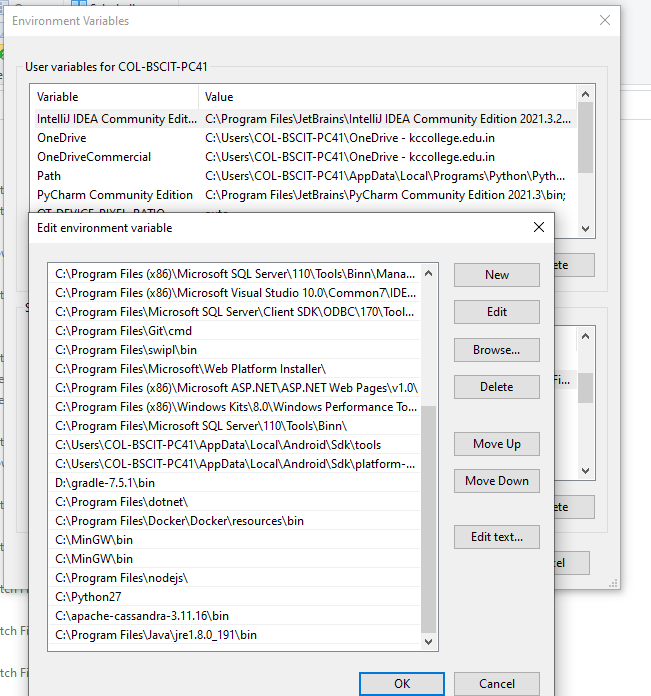
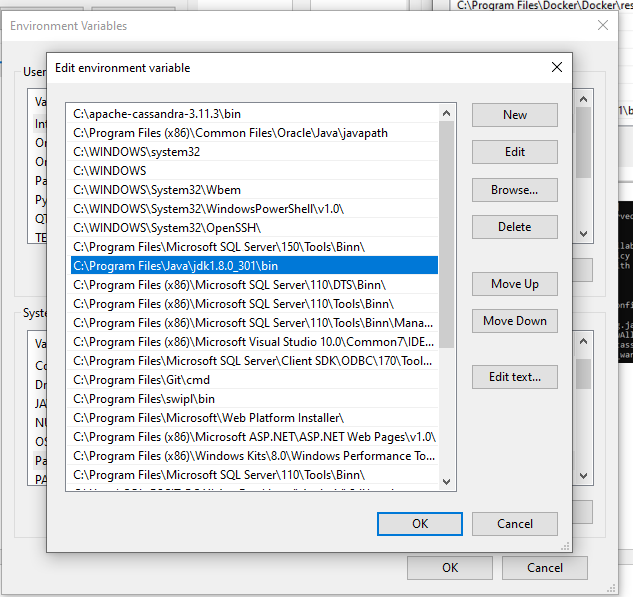
1. Install the following:

Python 2.7 , Java 1.8, Cassandra 3.11

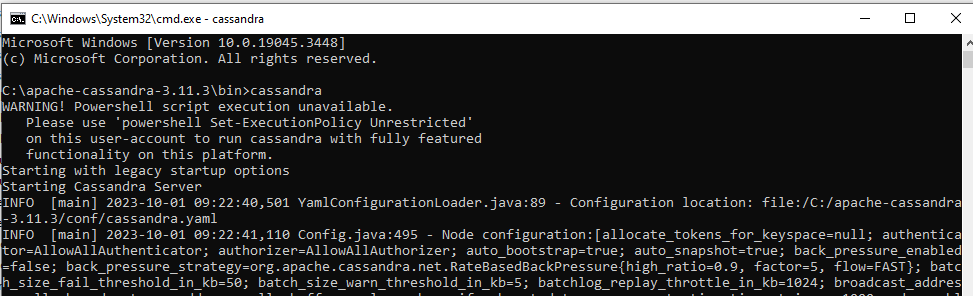
1. In the environment variables go to System variables set the path for java as JAVA\_HOME (jre path) and cassandra as CASSANDRA\_HOME as shown below



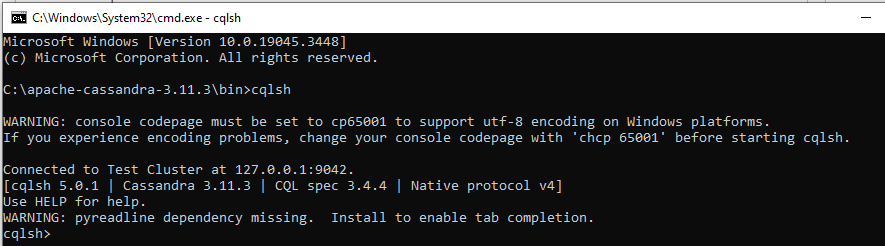
1. In System Variables go to path -> set the path for java (both jdk and jre path), python and cassandra as shown below

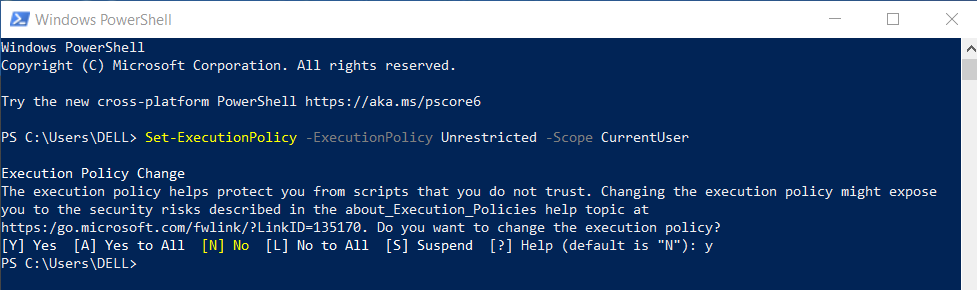


1. Open two Command Prompt -> In fisrt CMD enter the following command



1. In Second CMD enter the command **cqlsh** as shown below





## Practical 07 7A) Aim**:** Sentimental Analysis in R

Code:

install.packages('sentimentr') install.packages('tidyverse') install.packages('wordcloud') install.packages('tokenizers') install.packages('tm') install.packages('SentimentalAnalysis') library(tidyverse)

library(sentimentr) library(wordcloud) library(tokenizers) library(tm)

reviews <- read.csv("hotels.csv") head(reviews)

tokens <- tokenize\_words(reviews$Text) print(tokens)

wordcloud(words=tokens,min.freq=10,max.words = 50,colors="black") summary(tokens)

Corpus <-Corpus(VectorSource(reviews)) Corpus <- tm\_map(Corpus,removePunctuation) Corpus <- tm\_map(Corpus,removeNumbers)

Corpus <-tm\_map(Corpus,content\_transformer(tolower)) summary(Corpus)

head(Corpus)

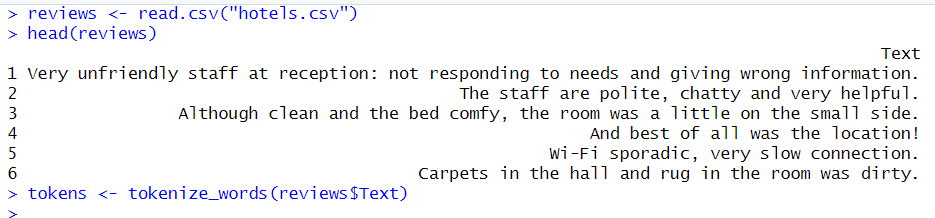
dtm <- DocumentTermMatrix(Corpus)

sentiments <- sentimentr\_data(dtm) for (i in 1:length(sentiments))

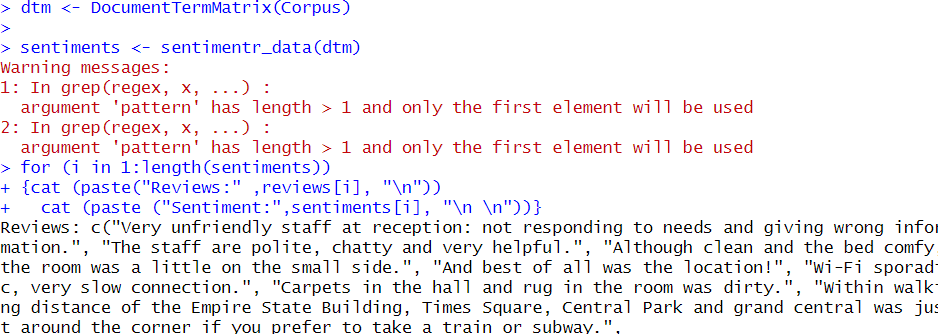
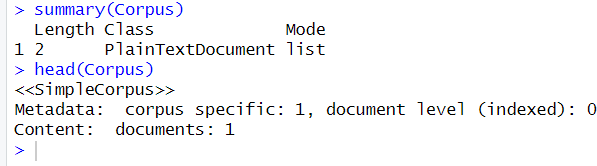
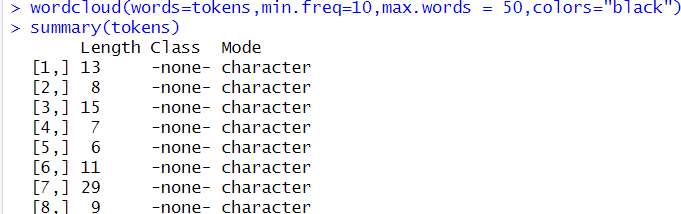
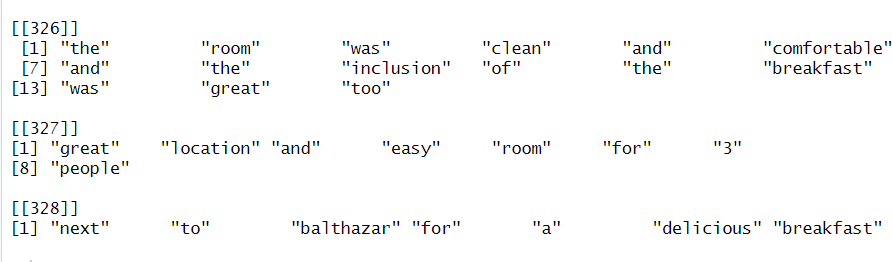
{cat (paste("Reviews:" ,reviews[i], "\n"))

cat (paste ("Sentiment:",sentiments[i], "\n \n"))}

## Output:



Print(tokens)



## 7B) Aim: Sentimental Analysis in R to analyze the data and term document matrix.

Code:

install.packages("rlang") install.packages("syuzhet") install.packages("lubridate") install.packages("ggplot2") install.packages("scales") install.packages("reshape2") install.packages("dplyr")

install.packages ("slam") install.packages("corpus") library(rlang)

library(syuzhet) #use for sentiment analysis library(lubridate)

library(ggplot2) library(scales) library(reshape2) library(dplyr)

print(getwd())

x<- read.csv("Data1.csv",header=TRUE) str(x)

tweets<-iconv(x$text, from="UTF-8", to="ASCII//TRANSLIT") str(tweets)

head(tweets)

corpus<-iconv(x$text)

corpus<-Corpus(VectorSource(corpus)) inspect(corpus[1:5])

corpus<-tm\_map(corpus,tolower) inspect(corpus[1:5])

corpus<-tm\_map(corpus,removeNumbers) inspect(corpus[1:5])

corpus<-tm\_map(corpus,removeWords,stopwords("english")) inspect(corpus[1:5])

corpus<-tm\_map(corpus,removePunctuation) inspect(corpus[1:5])

tdm<- TermDocumentMatrix(corpus) tdm<- as.matrix(tdm)

tdm[1:10,1:20]

print(tdm)

a<-rowSums(tdm) a<-subset(a,a>=45)

barplot(a,col=rainbow(50))

## Output:

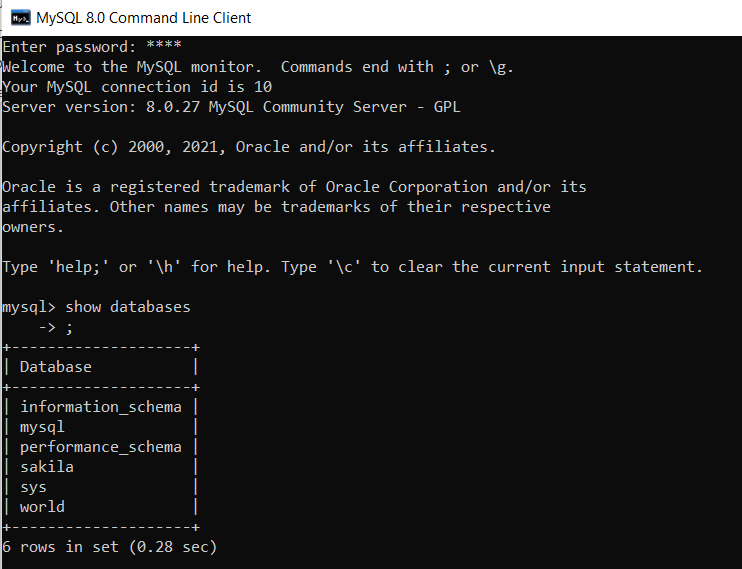
|  |
| --- |
| A screenshot of a computer error  Description automatically generated |
| A screenshot of a computer screen  Description automatically generated |
| A screenshot of a computer code  Description automatically generated |

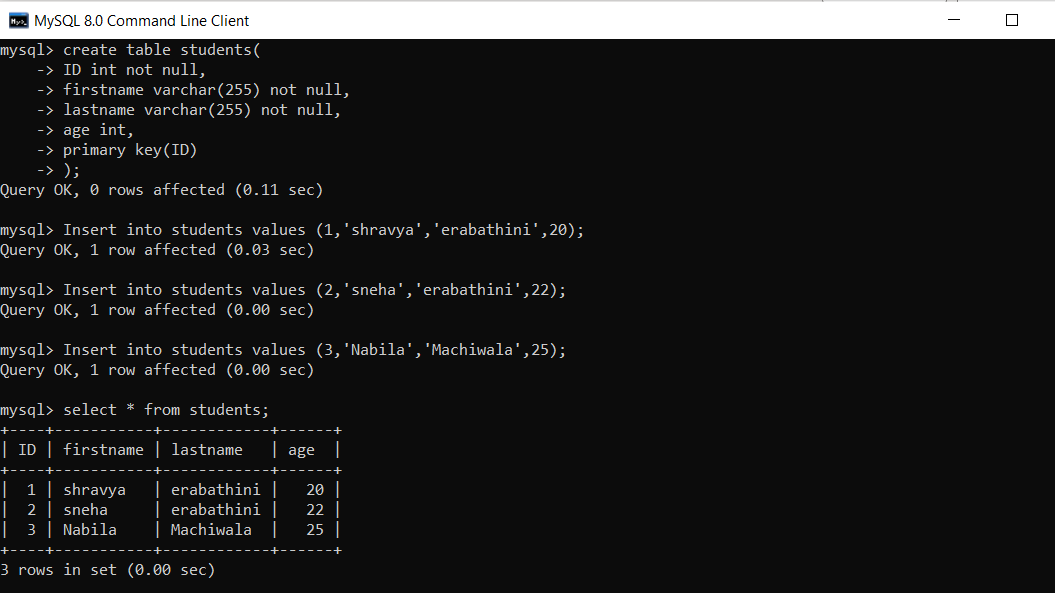
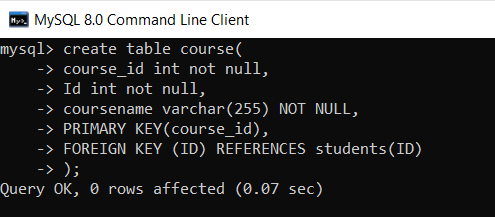
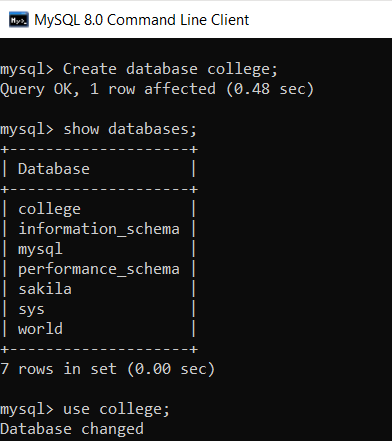
|  |
| --- |
| A screenshot of a computer  Description automatically generated |
|  |
| A screenshot of a computer program  Description automatically generated |
| A screen shot of a computer code  Description automatically generated |

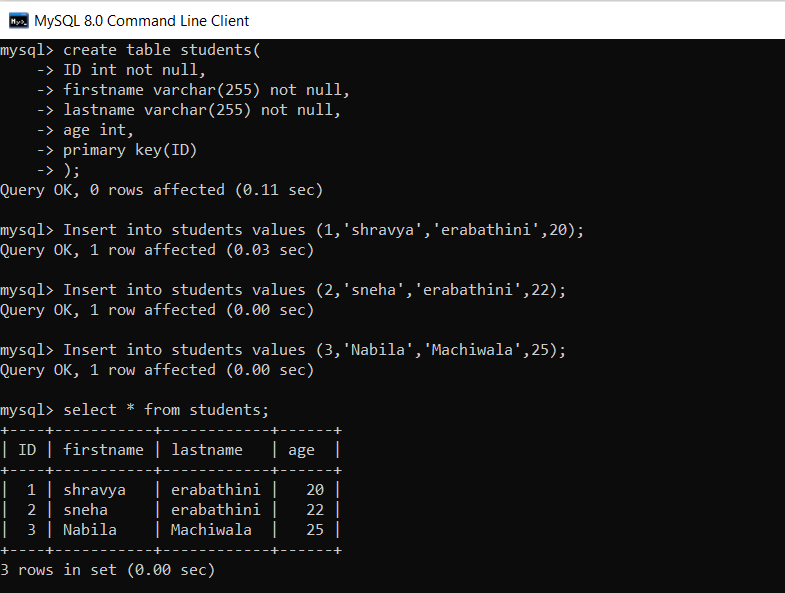
|  |
| --- |
| A screenshot of a computer code  Description automatically generated |
| A number on a white background  Description automatically generated |
| A colorful graph with text  Description automatically generated |

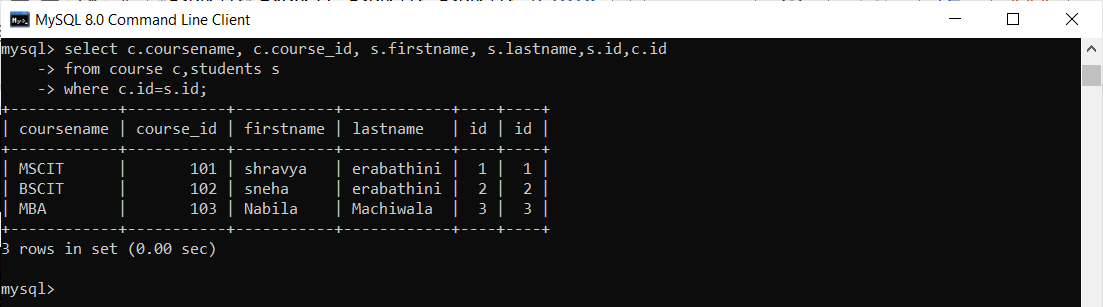
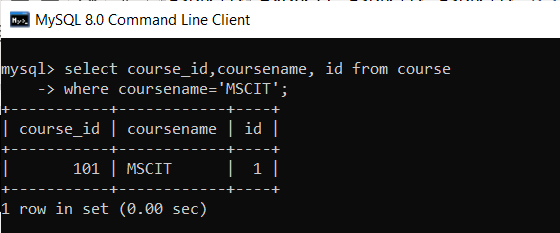
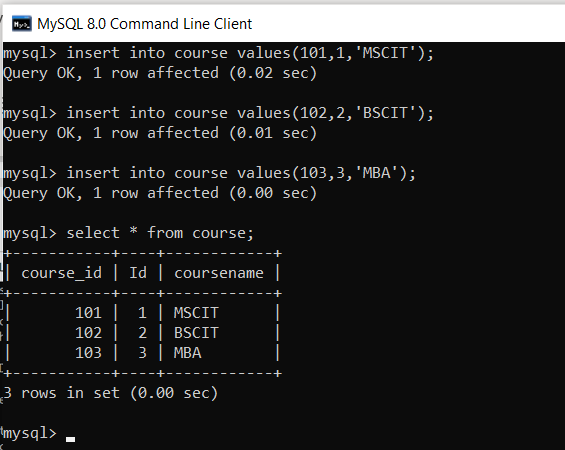
Practical 08

## Aim**:** Database - Create two tables and make primary and foreign key and use select and where clause.









Practical 09

## Aim: Calculate the sentiment analysis score and visualize the result. Code:

install.packages("rlang") install.packages("syuzhet") install.packages("lubridate") install.packages("ggplot2") install.packages("scales") install.packages("reshape2") install.packages("dplyr") install.packages("tm") install.packages("corpus")

library(rlang)

library(syuzhet) #use for sentiment analysis library(lubridate)

library(ggplot2) library(scales) library(reshape2) library(dplyr) library(tm) library(corpus)

x<- read.csv("Data1.csv",header=TRUE) str(x)

tweets<-iconv(x$text, from="UTF-8", to="ASCII//TRANSLIT") str(tweets)

head(tweets)

corpus<-iconv(x$text)

corpus<-Corpus(VectorSource(corpus)) inspect(corpus[1:5])

corpus<-tm\_map(corpus,tolower) inspect(corpus[1:5])

corpus<-tm\_map(corpus,removeNumbers) inspect(corpus[1:5])

corpus<-tm\_map(corpus,removeWords,stopwords("english")) inspect(corpus[1:5])

corpus<-tm\_map(corpus,removePunctuation) inspect(corpus[1:5])

tdm<- TermDocumentMatrix(corpus) tdm<- as.matrix(tdm)

tdm[1:10,1:20]

print(tdm)

a<-rowSums(tdm) a<-subset(a,a>=45)

barplot(a,col=rainbow(50))

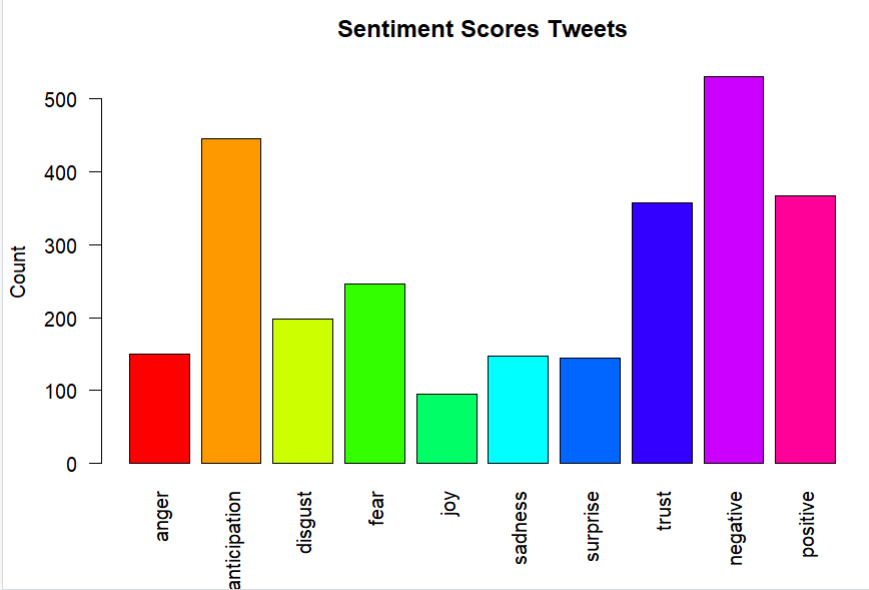
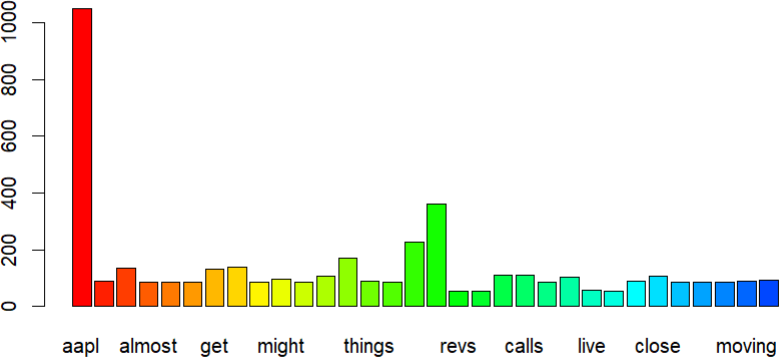
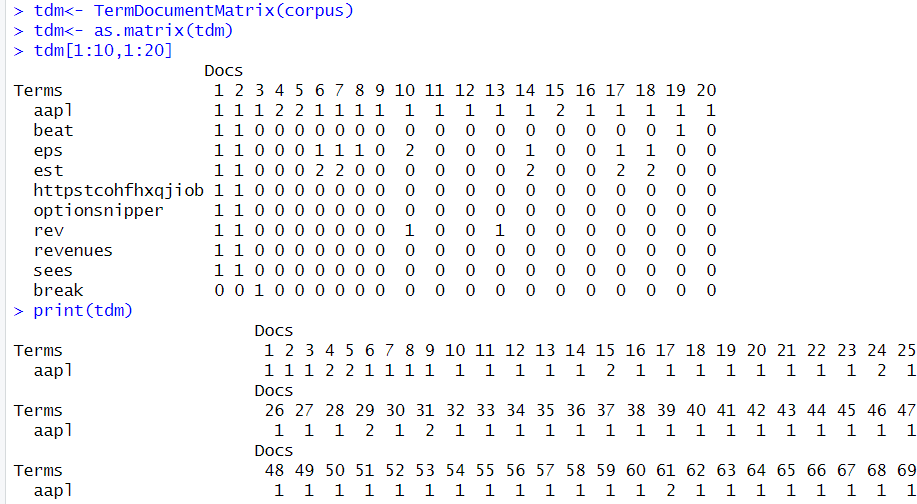
score<-get\_nrc\_sentiment(tweets) head(score)

barplot(colSums(score), las = 2, col = rainbow(10), ylab = 'Count', main = 'Sentiment Scores Tweets')

## Output:

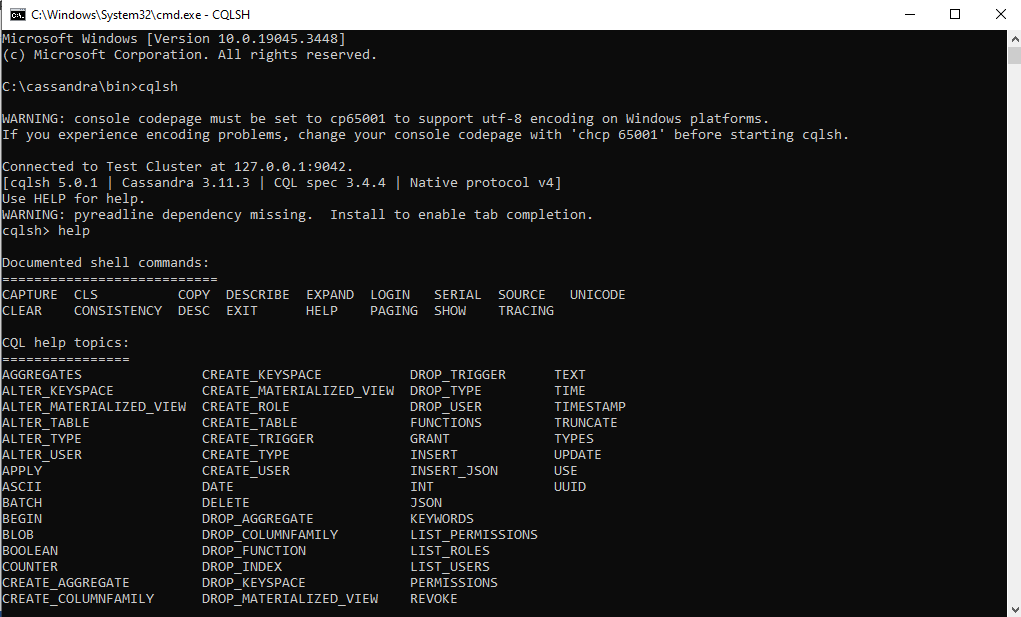
|  |
| --- |
| A screenshot of a computer  Description automatically generated |
| A screenshot of a computer program  Description automatically generated |
| A screenshot of a computer program  Description automatically generated |

|  |
| --- |
| A screenshot of a computer program  Description automatically generated |
| A screenshot of a computer program  Description automatically generated |
| A screenshot of a computer program  Description automatically generated |
| A screenshot of a computer program  Description automatically generated |



Practical 10A Aim: Creating Data Model using Cassandra

* cqlsh
* help



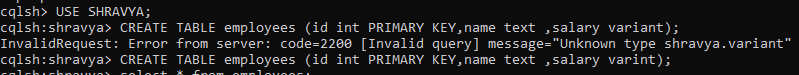
* Create Keyspace Shravya with REPLICATION =

{‘class’:’SimpleStrategy’,’replication\_factor’:3);

* Show keyspace;

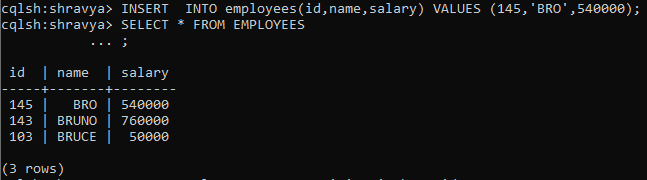


* Use shravya;
* Create table employees(id int PRIMARY KEY,name text,salary varint);

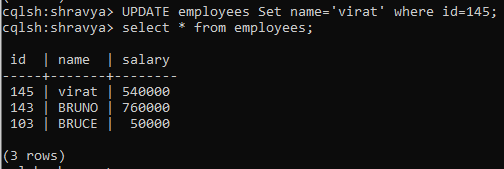


* Insert into employees(id,name,salary) values(103,’BRUNCE’,50000);
* Select \* from employees



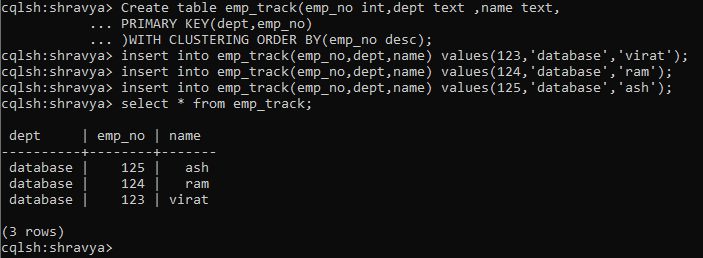


* Update employees set name=’virat’ where id=145;



Create clustering on emp\_no in the descending order

* Create table emp\_track(emp\_no int,dept text,name text,PRIMARY KEY(dept,emp\_no)) WITH Clustering ORDER BY(emp\_no desc);
* Insert into emp\_track(emp\_no,dept,name) values(123,’database’,’virat’);
* Select \* from emp\_track;



## Practical 10B

Aim: Create, Insert, Update and display the data from cassandra using python Code:

from cassandra.cluster import Cluster

*# Connect to Cassandra* cluster = Cluster(['127.0.0.1']) session = cluster.connect()

keyspace\_name = 'College' replication\_options = {

'class': 'SimpleStrategy', 'replication\_factor': 1

}

create\_keyspace\_query = f"""

CREATE KEYSPACE IF NOT EXISTS {keyspace\_name} WITH REPLICATION = {str(replication\_options)}

"""

session.execute(create\_keyspace\_query) print("Keyspace Created")

create\_query=session.prepare("CREATE TABLE College.Student (id int PRIMARY KEY, name text, address text)")

session.execute(create\_query) print("Table Created")

session.execute("INSERT INTO College.Student (id, name, address) VALUES (1, 'Shravya', 'Mumbai')")

session.execute("INSERT INTO College.Student (id, name, address) VALUES (2, 'Sneha', 'Kerala')") session.execute("INSERT INTO College.Student (id, name, address) VALUES (3, 'Manoja', 'Nashik')") print("Data Inserted")

select\_query = "SELECT \* FROM College.Student"

result = session.execute(select\_query) print("Student Details before update") for row in result:

print(f" ID: {row.id}, Name: {row.name}, Address: {row.address}")

update\_query = session.prepare("UPDATE College.Student SET address = 'Delhi' WHERE id = 2") session.execute(update\_query)

print("Data Updated") print("Data after update")

select\_query1 = "SELECT \* FROM College.Student" result = session.execute(select\_query1)

print("Student Details")

for row in result:

print(f" ID: {row.id}, Name: {row.name}, Address: {row.address}") session.shutdown()

cluster.shutdown()

## Output:

