### A LAB MANUAL

**On**

## Data Science and Big Data Analytics

**(III-B.Tech. II–Semester)**

**Course Code: 24CSPC46**

### Submitted to

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATASCIENCE)**

#### By

#### Mr. M. Raj Kumar

(Assistant Professor, Dept.of CSE(DS))



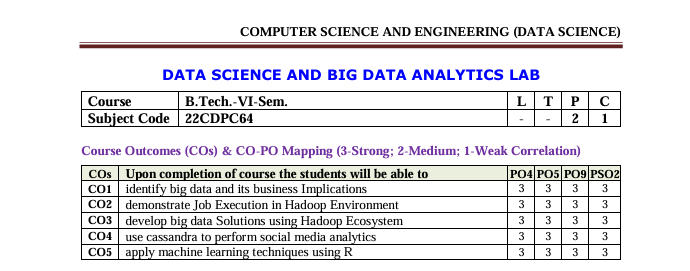
# CMR INSTITUTE OF TECHNOLOGY

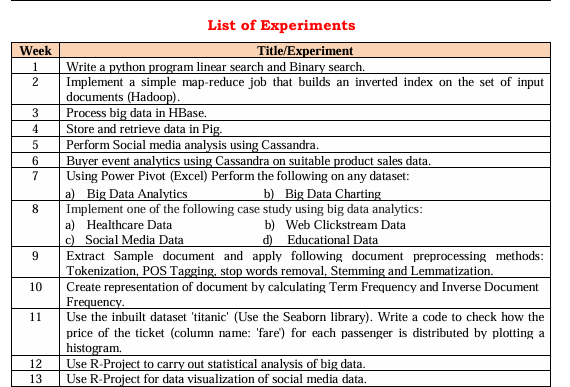
#### (UGC AUTONOMOUS)

Kandlakoya (V) , Medchal Road,Hyderabad–501401

## 

**Syllabus**





**Experiment-1**

**Aim:** Write a python program linear search and Binary search.

**Require Software& Tools: anaconda (jpyter note book)**

**Procedure:**

### Step-1: open any python IDE write the program

### Step-2: Linear Search:

### Step-3: Iterate through each element.

### Step-4: Compare with the target.

### Step-5: If found, return the index, else return -1.

### Step-6: run the program

### Step-7: Result

### Binary Search:

### Step-1: Sort the array (if needed).

### Step-2: Initialize low and high pointers

### Step-3: Find the middle element and compare with the target

### Step-4: Adjust the pointers (low or high) based on the comparison

### Step-5: Repeat until the target is found or the pointers cross

### Step-6: Return the index if found, else return -1

### Step-7: Run the program write the result.

### SOURCECODE:

### Linear search

### l=list()

### n=int(input("Enter number of elements to be inserted into list:"))

### print("Enter",n," Values")

### for i in range(n):

### l.append(int(input()))

### s=int(input("Enter element to be searched"))

### for i in range(len(l)):

### if l[i]==s:

### print(s," is found at position", i+1)

### break

### else:

### print("Element is not found")

**Binary search:**

l=list()

n=int(input("enter number of elememts"))

print("enter",n, "values")

for i in range(n):

l.append(int(input()))

l.sort()

s=int(input("enter element to be searched"))

low=0

high=len(l)-1

found=False

while low<=high:

mid=(low+high) // 2

if l[mid] == s:

print(s," is found at position ",mid+1)

found=True

break

elif l[mid]<s:

low=mid+1

else:

high=mid-1

if not found:

print(s," is not found in list")

### Out put:

### Linear search

Enter number of elements to be inserted into list: 5

Enter 5 Values

11

12

30

45

22

Enter element to be searched 30

30 is found at position 3

**Binary search:**

enter number of elememts 5

enter 5 values

10

25

30

40

50

enter element to be searched 25

25 is found at position 2

**Experiment-2**

**Aim: Implement a simple map-reduce job that builds an inverted index on the set of input documents (Hadoop).**

**Require Software& Tools: (Hadoop,java,linux,intellij)**

1.open vm ware start ubunt then open terminal

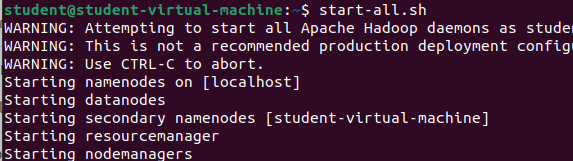
2.hadoop version



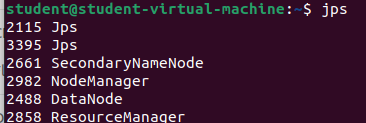
3.java –version



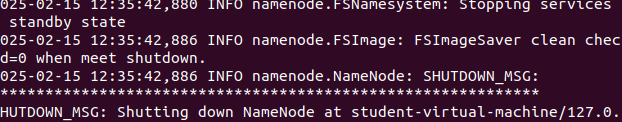
4.start-all.sh



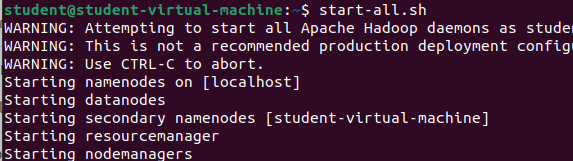
5.jps



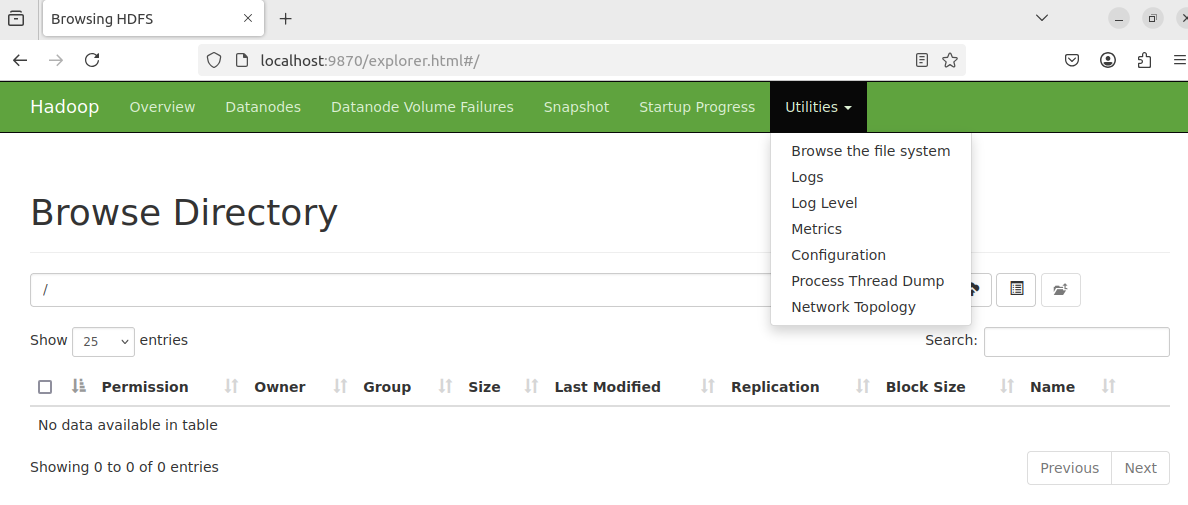
6.hadoop-3.4.0/bin/hdfs namenode –format



7.start-all.sh



8.open browser <http://localhost:9870>



9.open intellij idea click on new project give the name of the project

10.select maven goto advance give the group id name org.dsbda

11.remove main class

12.create dependencies in org.dsbda copy the dependencies code from the git hub

[**github.com/ rishikumar1992/DSBDA-LAB**](https://github.com/rishikumar1992/DSBDA-LAB)

13.go to maven click on the project name reload all projects

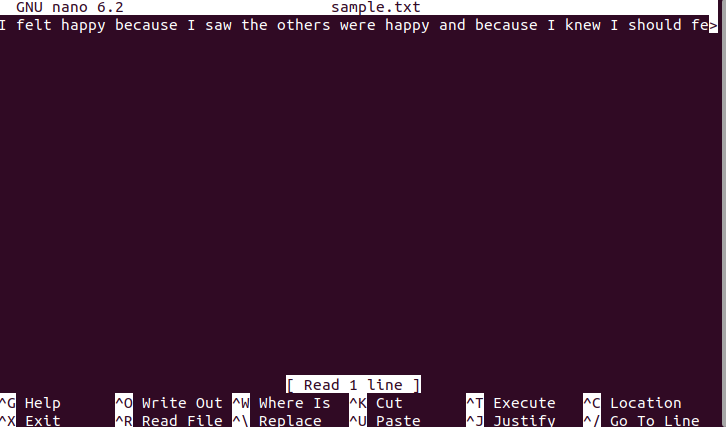
14.create the 3 java classes WC\_Mapper ,Reducer,Runner copy the code from git hub

15.create jar file ---> click on maven clean enter and maven install

16.target folder will be created which contains jar file

17.goto ubuntu tewrminal create the text file input2.txt

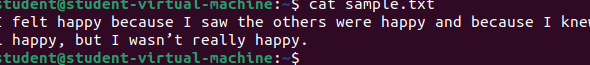
18.nano sample.txt



19.write some text with repeated words

20.cntrl+o enter cntrl+x

21.cat input.txt



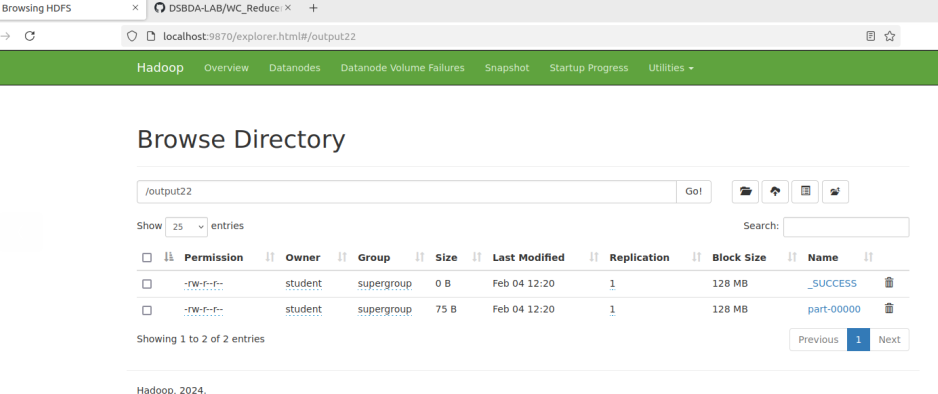
22.create the folder on the localhost hadoop fs -mkdir /input2

23.hadoop fs -put sample.txt /input2

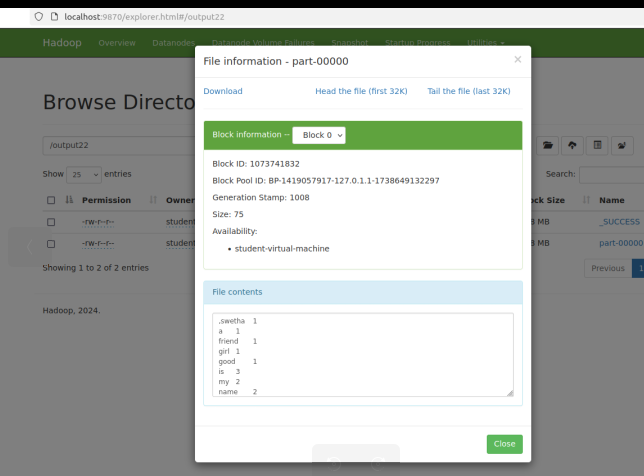
24.back to the local host check the file system

25.back to the intellij go to terminal hadoop jar target/week2-1.0-SNAPSHOT.jar org.dsbda.WC\_Runner /input2/input2.txt /output2

26.back to the local host check the file system

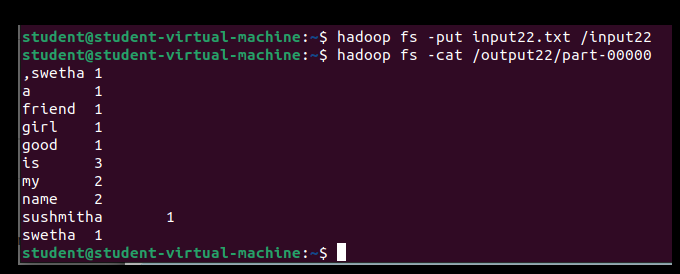


27.click on output2 file



28.back to the main terminal

29.hadoop fs -cat /output2/part-00000



**Experiment-3**

**Aim: To create table and process the big data in Hbase.**

**Require Software& Tools: Hadoop in Ubuntu, VM ware work station, Hbase**

**Installation steps for hbase**

1.download hbase from apache website(<https://dlcdn.apache.org/hbase/2.6.1/hbase-2.6.1-bin.tar.gz>)**St 2. Place the downloaded file into home folder**  
 hbase-1.1.2-bin.tar.gz in /home

3.**Unzip it by executing command $tar -xvf hbase-1.1.2-bin.tar.gz.**  
It will unzip the contents, and it will create hbase-1.1.2 in the location /home

4.**Open hbase-env.sh(to open this go to home/hadoop-2.6.1/conf right click on conf open with terminal)**

**5.in terminal enter the command: gedit hbase-env.sh**

**6.**Open hbase-env.sh as above and mention JAVA\_HOME path in the location.

**export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64 (save and close the file)**

**7.**Open ~/.bashrc file(gedit ~/.bashrc )and mention HBASE\_HOME path as shown in below

export HBASE\_HOME=/home/student/hbase-2.6.1

export PATH=$PATH:$HBASE\_HOME/bin**(save and close the file)**

8. **Add properties in the file**  
Open hbase-site.xml(gedit hbase-site.xml) and place the following properties inside the file

<property>

<name>hbase.rootdir</name>

<value>file:///home/student/hbase-2.6.1</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/home/student/hbase-2.6.1/zookeeper</value>

</property>

**(save and close the file)**

Here we are placing two properties

* One for HBase root directory and
* Second one for data directory correspond to ZooKeeper.

All HMaster and ZooKeeper activities point out to this hbase-site.xml.

9.close the terminal and go to home/hbase-2.6.1/bin open with terminal

10 type the following commands

**a.start-hbase.sh**

**b.jps**

c.**hbase shell**

Experiment:

**1.Creating table in hbase**

create ‘customer’,’customer\_info’,’customer\_details’

output: Creating table customer with column families: [customer\_info, customer\_details] 0 row(s) in 0.1230 seconds

**2.To list the tables in hbase: list**

Output:

scss

Copy

TABLE

customer

1 row(s) in 0.0100 seconds

**3.To insert values into table:**

put ‘customer’,’1’,’customer\_info:name’,’sita’

put ‘customer’,’1’,’customer\_details:mobile’,’9999999999’

**4. To display the contents of the table**

get ‘customer’,’1’



**5.To insert other values into table:**

put ‘customer’,’1’,’customer\_info:age’,’25’

put ‘customer’,’1’,’customer\_details:email’,’sita@gmail.com’

**6. To display the contents of the table**

get ‘customer’,’1’



**7. To insert other row**

put ‘customer’,’2’,’customer\_info:name’,’rama’

put ‘customer’,’2’,’customer\_details:mobile’,’9898999999’

put ‘customer’,’2’,’customer\_info:age’,’28’

put ‘customer’,’2’,’customer\_details:email’,’rama@gmail.com’

**8.To update the details**

put 'customer', '1', '’customer \_info:name', 'John' # Update name

put 'customer', '1', '’customer \_info:age', '31' # Update age

**scan customer;**



**9. Delete an entire row (all columns) from the table:**

deleteall 'customer', '1'

**10. Delete a specific column from a row:**

delete 'customer', '1', 'customer\_details:mobile’

**Experiment-4**

**Aim:** Store and retrieve data in Pig.

**Require Software& Tools: Hadoop in Ubuntu,apache Pig(0.17.0)**

**Procedure:**

1. Download Apache Pig:

First of all, download the latest version of Apache Pig from the following website − <https://pig.apache.org/>

Open the homepage of Apache Pig website. Under the section **News,** click on the link **release page,**click on **Download a release now**

**Click on pig-0.16.0/**

**Click on** [**pig-0.16.0.tar.gz**](https://dlcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz)

## Install Apache Pig

### Step 1:

Create a directory with the name Pig in the same directory where the installation directories of **Hadoop, Java,** and other software were installed.

$mkdir pig

### Step 2:

Extract the downloaded tar files as shown below.

cd Downloads/

$ tar zxvf pig-0.16.0.tar.gz

### Step 3:

Move the content of **pig-0.15.0.tar.gz** file to the **Pig** directory created earlier as shown below.

$ mv pig-0.16.0.tar.gz/\* /home/Pig/

## Configure Apache Pig

After installing Apache Pig, we have to configure it. To configure, we need to edit two files − **bashrc and pig.properties**

$pig -h properties

Open bashrc

$gedit ~/.bashrc  
  
add this path

export PIG\_HOME=/home/student/pig

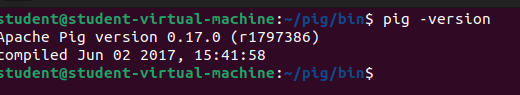
export PATH=$PATH:$PIG\_HOME/bin

export PIG\_CLASSPATH=$HADOOP\_HOME/conf  
  
 **verifying installation**

step1:open vm ware workstation and start ubuntu

Step2:click on home then go to the pig folder then select bin folder and right click open in terminal

Step3: check the pig version :$pig –version



Step4:Create a text file

Step5:gedit studata.txt write the some text

**Data**

Krishna,1,22,cse

rani,2,21,csd

raju,3,22,aiml

sita,2,20,cse

rama,1,23,csd

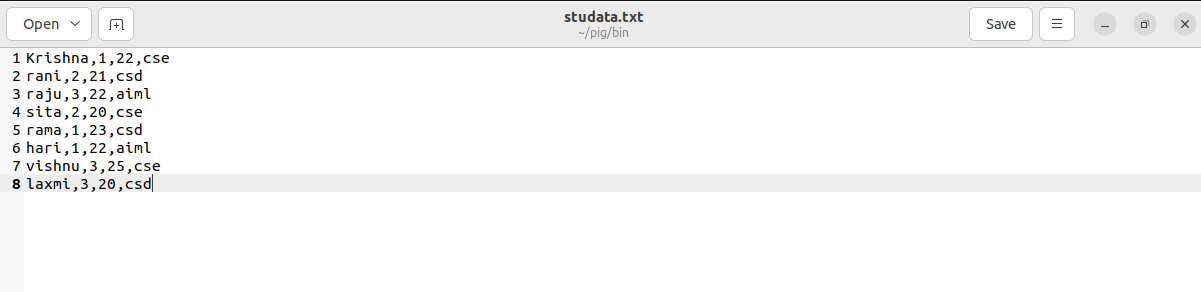
hari,1,22,aiml

vishnu,3,25,cse

laxmi,3,20,csd

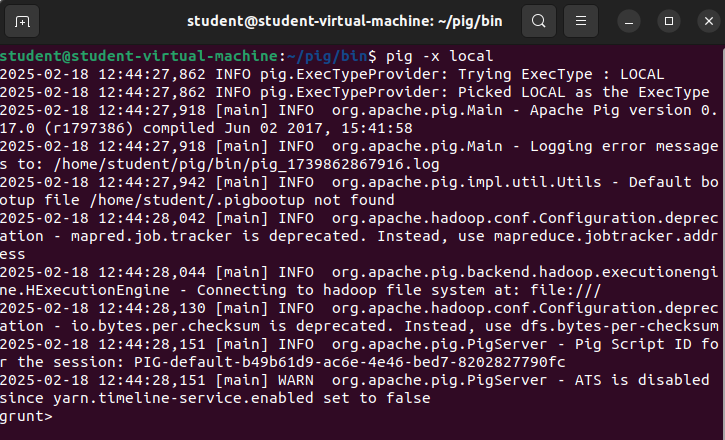
vishva,2,22,aiml

teja,4,23,cse



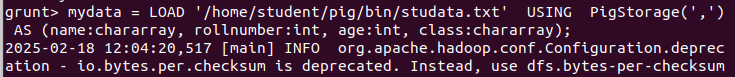
**step6:save and close**

**step7:start pig:** pig –x local

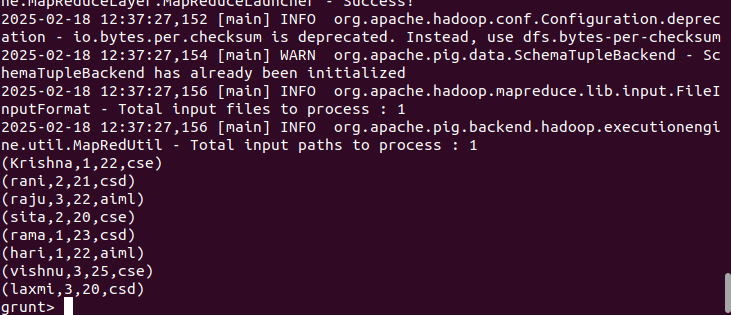
****

**step8:To load data from local system**

mydata = LOAD '/home/student/pig/bin/studata.txt' USING PigStorage(',') AS (name:chararray, rollnumber:int, age:int, class:chararray);



**step9:To dump the loaded data: dump mydata;**



step10:to describe the data :describe mydata;



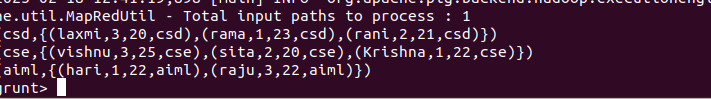
**step11:Query 1: Grouping All Records Class.**

(This command will group all the records by the column Class)

**Step12:grunt>** studentsbranch = GROUP mydata BY class;

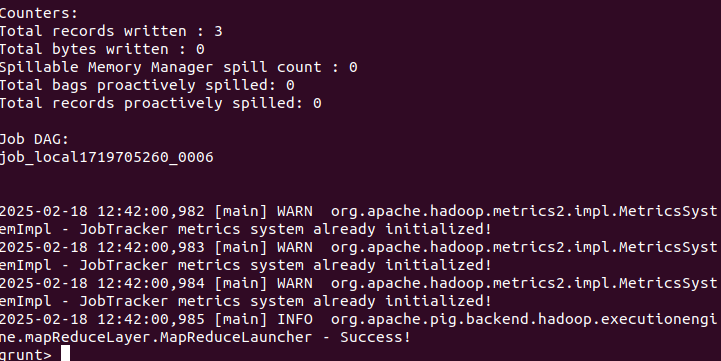
To see the output

Step13: dump studentsbranch;

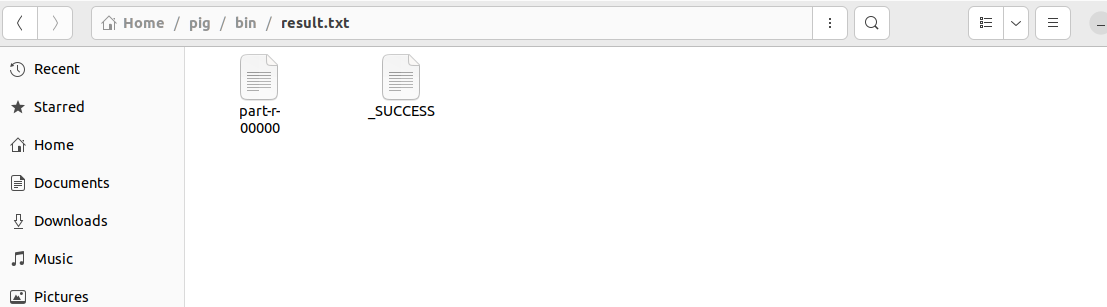


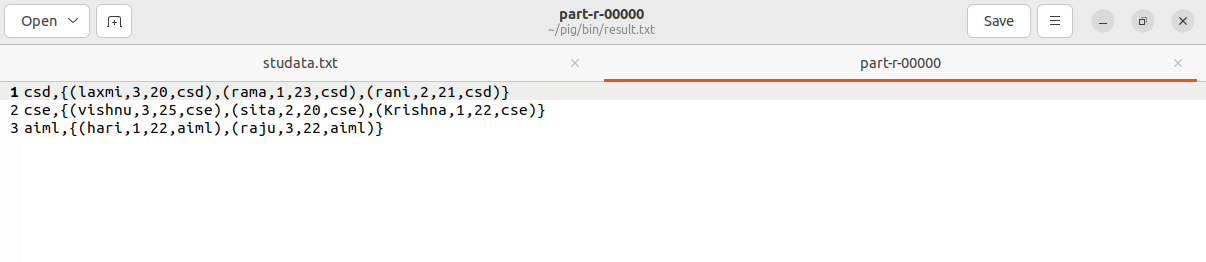
**Step14: To store data into local system**

STORE studentsbranch INTO 'result.txt' USING PigStorage(',');



Step15: see the output in bin folder select the output txt the open part-r-00000





**Experiment-5**

**Aim:** Perform Social media analysis using Cassandra.

**Require Software& Tools: Hadoop in Ubuntu, VM ware work station,Cassandra**

**Procedure: below imp queries of cassandra**

Step1: cqlsh> CAPTURE '/home/hadoop/CassandraProgs/Outputfile'

Step2: cqlsh:tutorialspoint> select \* from emp;

Step3: cqlsh:tutorialspoint> capture off;

Step4: cqlsh:tutorialspoint> CONSISTENCY

Step5: cqlsh:tutorialspoint> COPY emp (emp\_id, emp\_city, emp\_name, emp\_phone,emp\_sal)

TO ‘myfile’;

Step6: cqlsh:tutorialspoint> describe cluster;

Step7: cqlsh:tutorialspoint> describe keyspaces;

Step8: cqlsh:tutorialspoint> describe table emp;

Step9: cqlsh:tutorialspoint> describe type card\_details;

Step10: cqlsh:tutorialspoint> DESCRIBE TYPES;

Step11: cqlsh:tutorialspoint> expand on;

cqlsh:tutorialspoint> select \* from emp;

cqlsh:tutorialspoint> expand off;

Step12: cqlsh:tutorialspoint> source '/home/hadoop/CassandraProgs/inputfile';

### SOURCECODE AND OUTPUT:

Performing social media analysis using **Apache Cassandra** involves storing and processing large volumes of social media data efficiently. Cassandra is a distributed NoSQL database that is highly scalable and fault-tolerant, which makes it ideal for storing social media data like posts, comments, likes, and user interactions.

Let's go through a step-by-step example of how to perform social media analysis using Cassandra.

**Step 1: Set Up Cassandra**

Before we start, ensure that Apache Cassandra is installed and running on your system. You can download it from the official website or use Docker to quickly set up a Cassandra container.

**docker run --name cassandra -d -p 9042:9042 cassandra:latest**

This will start a Cassandra instance that listens on port 9042, the default port for CQL (Cassandra Query Language).

**Step 2: Design the Data Model**

For social media analysis, we need to design an appropriate data model. Social media data generally includes posts, comments, likes, and user data. In Cassandra, data modeling is key, as it is optimized for fast writes and specific query patterns.

Let’s consider a simplified example of the following social media data:

1. **User Table:** User profile details (e.g., user\_id, name, email).
2. **Posts Table:** Posts by users (e.g., post\_id, user\_id, content, timestamp).
3. **Comments Table:** Comments on posts (e.g., comment\_id, post\_id, user\_id, content, timestamp).
4. **Likes Table:** Likes on posts (e.g., post\_id, user\_id, timestamp).

### Step 3: Create Keyspaces and Tables in Cassandra

Once Cassandra is set up, we can create a keyspace and the necessary tables for storing social media data.

#### Create Keyspace

In Cassandra, a **keyspace** is similar to a database in relational systems. We can create a keyspace with the following command in CQL (Cassandra Query Language):

**CREATE KEYSPACE social\_media**

**WITH replication = {'class': 'SimpleStrategy', 'replication\_factor': 1};**

his command creates a keyspace named social\_media with a replication factor of 1. In production, you might want to adjust the replication factor based on fault tolerance needs.

#### Create Tables

Now, let's create the necessary tables to store social media data.

**Create User Table:**

CREATE TABLE social\_media.users (

user\_id UUID PRIMARY KEY,

name TEXT,

email TEXT

);

**Create Posts Table:**

CREATE TABLE social\_media.posts (

post\_id UUID PRIMARY KEY,

user\_id UUID,

content TEXT,

timestamp TIMESTAMP

);

**Create Comments Table:**

CREATE TABLE social\_media.comments (

comment\_id UUID PRIMARY KEY,

post\_id UUID,

user\_id UUID,

content TEXT,

timestamp TIMESTAMP

);

**Create Likes Table:**

CREATE TABLE social\_media.likes (

post\_id UUID,

user\_id UUID,

timestamp TIMESTAMP,

PRIMARY KEY (post\_id, user\_id)

);

**Step 4: Insert Data**

Now that the tables are created, let’s insert some sample data into them.

**Insert Users:**

INSERT INTO social\_media.users (user\_id, name, email) VALUES (uuid(), 'Alice', 'alice@example.com');

INSERT INTO social\_media.users (user\_id, name, email) VALUES (uuid(), 'Bob', 'bob@example.com');

**Insert Posts:**

INSERT INTO social\_media.posts (post\_id, user\_id, content, timestamp) VALUES (uuid(), <user\_id\_Alice>, 'This is Alice\'s first post!', toTimestamp(now()));

INSERT INTO social\_media.posts (post\_id, user\_id, content, timestamp) VALUES (uuid(), <user\_id\_Bob>, 'This is Bob\'s first post!', toTimestamp(now()));

**Insert Comments:**

INSERT INTO social\_media.comments (comment\_id, post\_id, user\_id, content, timestamp) VALUES (uuid(), <post\_id\_1>, <user\_id\_Bob>, 'Great post, Alice!', toTimestamp(now()));

INSERT INTO social\_media.comments (comment\_id, post\_id, user\_id, content, timestamp) VALUES (uuid(), <post\_id\_2>, <user\_id\_Alice>, 'Thanks for the post, Bob!', toTimestamp(now()));

**Insert Likes:**

INSERT INTO social\_media.likes (post\_id, user\_id, timestamp) VALUES (<post\_id\_1>, <user\_id\_Bob>, toTimestamp(now()));

INSERT INTO social\_media.likes (post\_id, user\_id, timestamp) VALUES (<post\_id\_2>, <user\_id\_Alice>, toTimestamp(now()));

**Get Posts by User:**

SELECT \* FROM social\_media.posts WHERE user\_id = <user\_id\_Alice>;

This will return all posts made by Alice.

**Get Comments on a Post:**

SELECT \* FROM social\_media.comments WHERE post\_id = <post\_id\_1>;

This will return all comments for a particular post.

**Get Likes on a Post:**

SELECT \* FROM social\_media.likes WHERE post\_id = <post\_id\_1>;

This will return all users who liked a particular post.

**Advanced Analysis**

For more complex analysis, such as finding the most liked posts or analyzing the sentiment of comments, you can use additional tools alongside Cassandra.

* **Apache Spark:** You can integrate **Apache Spark** with Cassandra for large-scale data processing and advanced analytics, such as aggregations, sentiment analysis, or recommendations.
* **Machine Learning Models:** You can apply machine learning algorithms to analyze user behavior, predict trends, or classify posts/comments.

**For example, to calculate the most liked posts:**

SELECT post\_id, COUNT(user\_id) AS likes\_count

FROM social\_media.likes

GROUP BY post\_id

ORDER BY likes\_count DESC;

However, as Cassandra doesn’t support complex aggregation queries like SQL databases, tools like **Apache Spark** are typically used to process the data and perform such analysis.

**Visualize the Results**

Once the data has been queried and processed, you can visualize the results using tools like:

* **Apache Superset**
* **Tableau**
* **Power BI**

These tools can be connected to Cassandra or Apache Spark to create dashboards for monitoring social media trends, user engagement, and more.

**Experiment-6**

**Aim:** To perform the buyer event analysis using Cassandra on sales data

**Require Software& Tools: Hadoop in Ubuntu, VM ware work station, Cassandra**

### Procedure: Step 1: Setting Up Apache Cassandra

Before starting with the analysis, ensure that you have Apache Cassandra installed and running. If you don't have it installed, download and follow the [installation guide](https://cassandra.apache.org/_/download.html).

### Step 2: Data Modeling for Sales Data

Sales data typically includes information such as:

* **Sale ID** (Unique identifier for the sale)
* **Buyer ID** (Unique identifier for the buyer)
* **Item ID** (Product purchased)
* **Quantity** (Amount of the product bought)
* **Price** (Price of the product)
* **Timestamp** (When the sale occurred)

For the purpose of analysis, you might want to store the data in a way that allows you to quickly query buyer behavior.

#### Example Schema Design:

In Cassandra, it’s important to design your tables based on the queries you intend to perform. For buyer event analysis, you might want to track the sales per buyer or analyze buyer activity over time.

##### Table 1: sales\_by\_buyer

This table will store sales data for each buyer.

CREATE TABLE sales\_by\_buyer (

buyer\_id UUID,

sale\_id UUID,

item\_id UUID,

quantity INT,

price DECIMAL,

timestamp TIMESTAMP,

PRIMARY KEY (buyer\_id, timestamp, sale\_id)

);

**Explanation:**

* buyer\_id: Partition key, ensures that data is grouped by buyer.
* timestamp: Clustering key to store sales chronologically per buyer.
* sale\_id: Uniquely identifies each sale for a buyer.

##### **Table 2: sales\_by\_item**

This table stores sales data per item, allowing you to track item-specific buyer events.

CREATE TABLE sales\_by\_item (

item\_id UUID,

sale\_id UUID,

buyer\_id UUID,

quantity INT,

price DECIMAL,

timestamp TIMESTAMP,

PRIMARY KEY (item\_id, timestamp, sale\_id)

);

### Step 3: Insert Sales Data into Cassandra

Once your tables are set up, you can insert sample sales data into the tables using INSERT INTO.

#### Sample Query to Insert Data:

INSERT INTO sales\_by\_buyer (buyer\_id, sale\_id, item\_id, quantity, price, timestamp)

VALUES (uuid(), uuid(), uuid(), 2, 100.50, toTimestamp(now()));

INSERT INTO sales\_by\_item (item\_id, sale\_id, buyer\_id, quantity, price, timestamp)

VALUES (uuid(), uuid(), uuid(), 2, 100.50, toTimestamp(now()));

You can repeat this process to insert multiple rows.

### Step 4: Perform Analysis Queries

#### 1. ****Retrieve Total Sales per Buyer****

You can query the sales\_by\_buyer table to get the total sales for a specific buyer.

SELECT buyer\_id, SUM(quantity \* price) AS total\_spent

FROM sales\_by\_buyer

WHERE buyer\_id = <specific\_buyer\_id>

GROUP BY buyer\_id;

#### 2. ****Find All Purchases for a Specific Buyer****

If you want to analyze the buying pattern of a specific buyer, you can retrieve all of their purchases.

SELECT sale\_id, item\_id, quantity, price, timestamp

FROM sales\_by\_buyer

WHERE buyer\_id = <specific\_buyer\_id>;

#### 3. ****Find Popular Items****

To find out which items are being bought the most across all buyers, you can query the sales\_by\_item table.

SELECT item\_id, SUM(quantity) AS total\_sold

FROM sales\_by\_item

GROUP BY item\_id

ORDER BY total\_sold DESC;

#### 4. ****Find Buyers Who Purchased a Specific Item****

If you’re interested in finding which buyers bought a specific item, you can query the sales\_by\_item table.

SELECT buyer\_id, sale\_id, quantity, price, timestamp

FROM sales\_by\_item

WHERE item\_id = <specific\_item\_id>;

### Step 5: Analyze Buyer Events Using Aggregations

After querying the data, you may want to perform aggregations on the results.

#### 1. ****Buyer Retention Analysis****

You can group buyers by their last purchase timestamp to check for repeat buyers. For example, buyers who made a purchase within the last 30 days are considered retained.

SELECT buyer\_id, MAX(timestamp) AS last\_purchase

FROM sales\_by\_buyer

GROUP BY buyer\_id

HAVING MAX(timestamp) > toTimestamp(now()) - 30;

#### 2. ****Frequent Buyers****

Find the top N buyers who have made the most purchases:

SELECT buyer\_id, COUNT(sale\_id) AS purchase\_count

FROM sales\_by\_buyer

GROUP BY buyer\_id

ORDER BY purchase\_count DESC

LIMIT 10;

#### 3. ****Item Purchase Analysis****

Analyze which buyers tend to buy specific combinations of items:

SELECT buyer\_id, item\_id, SUM(quantity) AS total\_quantity

FROM sales\_by\_buyer

WHERE item\_id IN (<item\_id1>, <item\_id2>)

GROUP BY buyer\_id, item\_id;

### Step 6: Visualize the Data

While Cassandra is great for handling large-scale data, you may want to visualize the results using a tool like **Apache Spark** with **Cassandra Connector** or use an external tool like **Tableau** or **Grafana** for better insights.

#### Example Visualization:

* **Total spending by each buyer** could be visualized as a bar chart.
* **Most purchased items** could be visualized as a pie chart.
* **Buyer retention** could be visualized as a line graph over time.

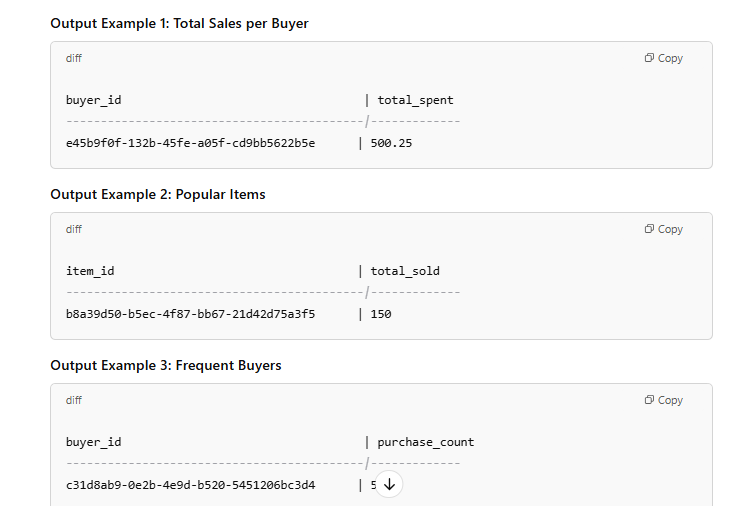
### Step 7: Optimization & Scaling

For large-scale data, consider:

* **Data Modeling**: Proper design of partition keys and clustering keys ensures efficient querying.
* **Indexing**: Use secondary indexes carefully as they may not be ideal for large datasets. Consider using **Materialized Views** or **Search** in Elasticsearch for more complex querying.

### Step 8: Output & Results

When querying the tables as shown in the previous steps, you will get output that looks like this (the exact results depend on your inserted data):



**Experiment-7**

**Aim:** Using Power Pivot (Excel) Perform the following on any dataset:

a) Big Data Analytics b) Big Data Charting

**Require Software& Tools: Ubuntu, VM ware work station, Libre office Calc**

**Procedure:**   
1. Open Ubuntu

2.start libra office calc

3. got to file -> open -> load data to libra office calc

4. Formatting date using function

5. Go to cell D -> right click and inser column before

New column will be created

Name it as “Formatted Dates”

6. Formula for converting date dd/mm/yyyy to date month year

=TEXT(C2,”d mmmm yyyy”)

7. Upon cell D -> right click and insert column after

New column will be created

Name it as “Year”

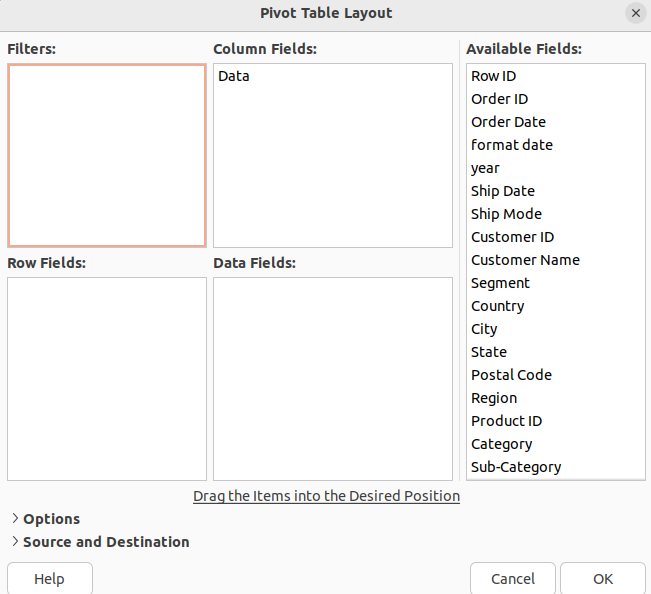
8.Select E column -> data -> sort ascending then click on extend selection an click on ok

**Data Analysis:**

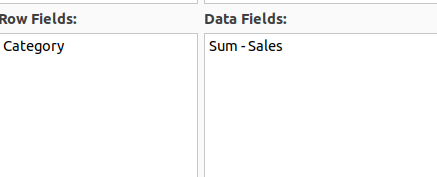
* Which category of products with the highest sales?
* Which sub-category has the highest sales?
* Which region drives the most sales to this super store?

**Pivot Tables and Charts:**

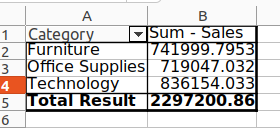
* Which category of products with the highest sales?
  1. Select data tab and go to pivot table (insert or edit) select source current selection then click on ok



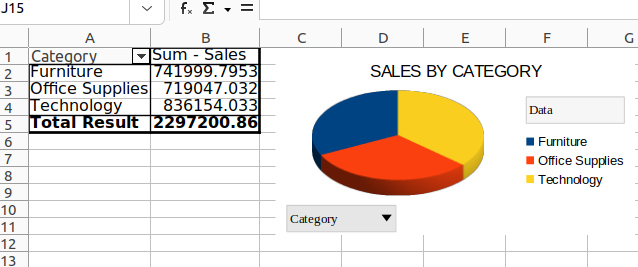
* 1. Drag category into row fileds and sales into data fileds then click on ok



* 1. It will be display like this



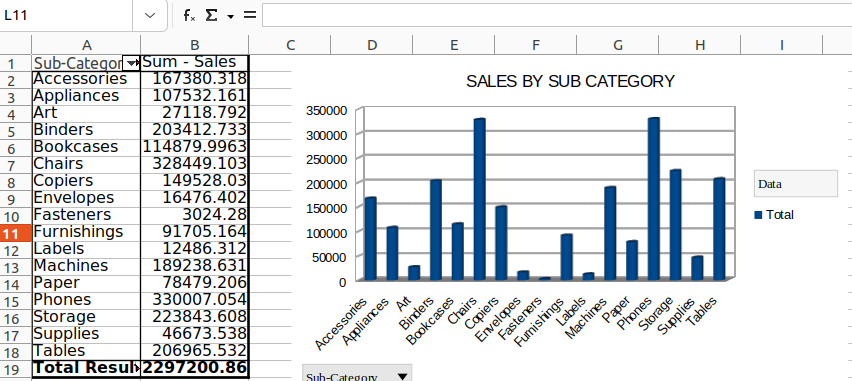
* 1. Go to insert >chart>pie>3D look creat tittle name “sales by catogory”



* Which sub-category has the highest sales

Drag sub-category into row fileds and sales into data fileds then click on ok

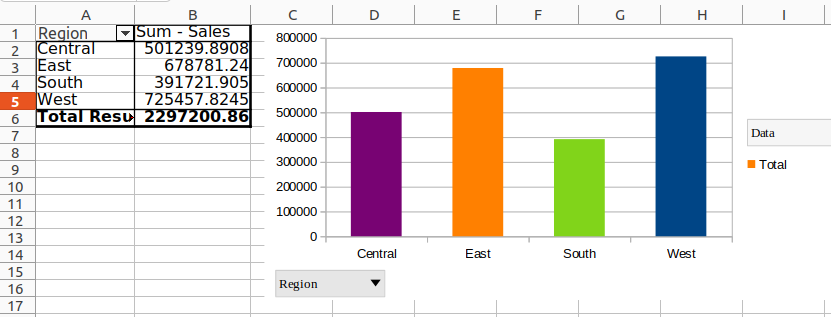
Go to insert >chart>bar chart>3D look creat tittle name “sales by sub-catogory”



Which region drives the most sales to this super store.

Drag region into row fileds and sales into data fileds then click on ok

Go to insert >chart>bar chart>3D look creat tittle name “sales by region”



**.**

**Experiment-8**

**AIM: Implement one of the following case studies using big data analytics:**

a) Healthcare Data b) Web Clickstream Data

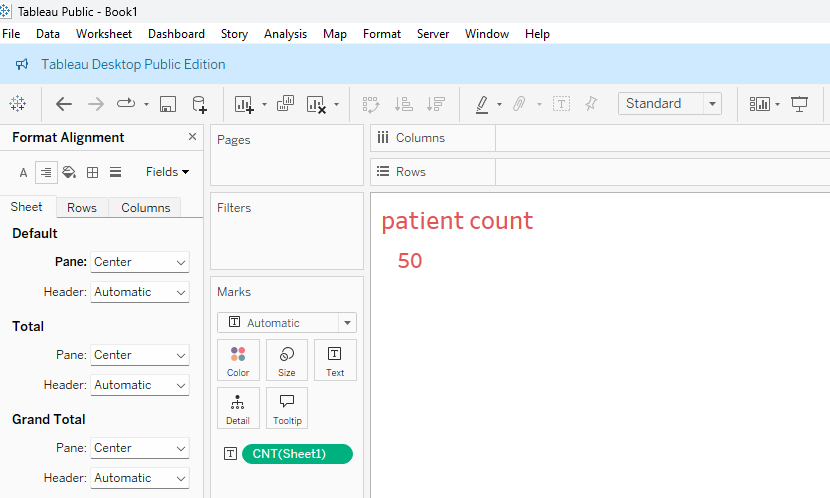
c) Social Media Data d) Educational Data

**Require Software& Tools: Tableau Public**

Procedure: step:-1

* Open tableau public and insert the exel sheet data (pateint\_blood\_test)
* Create new sheet and rename as **patient count** ,set the font style,size,color
* Drag sheet1(count) to rows then select text table
* Right click on count(50) format select worksheet -> change font color and size

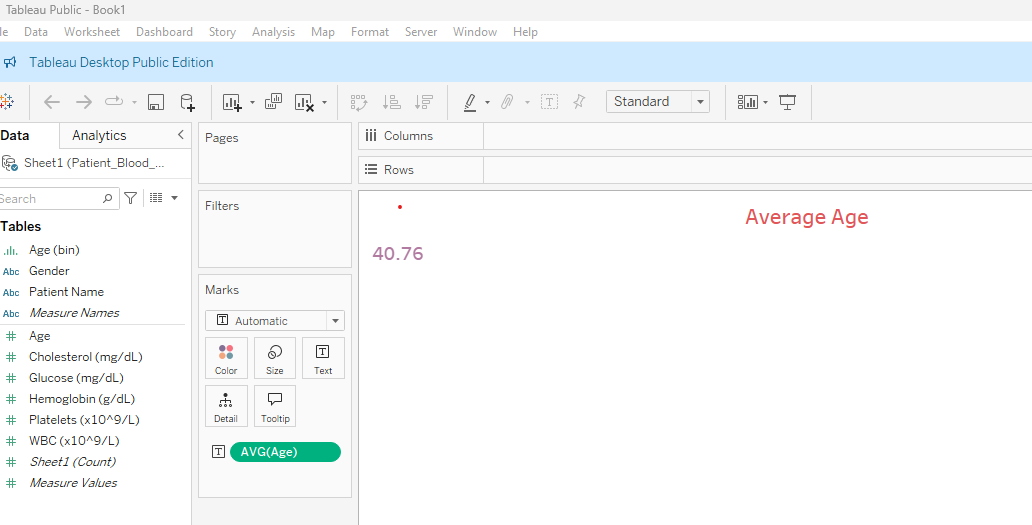
Then go to allignment -> pane-> center



step:-2

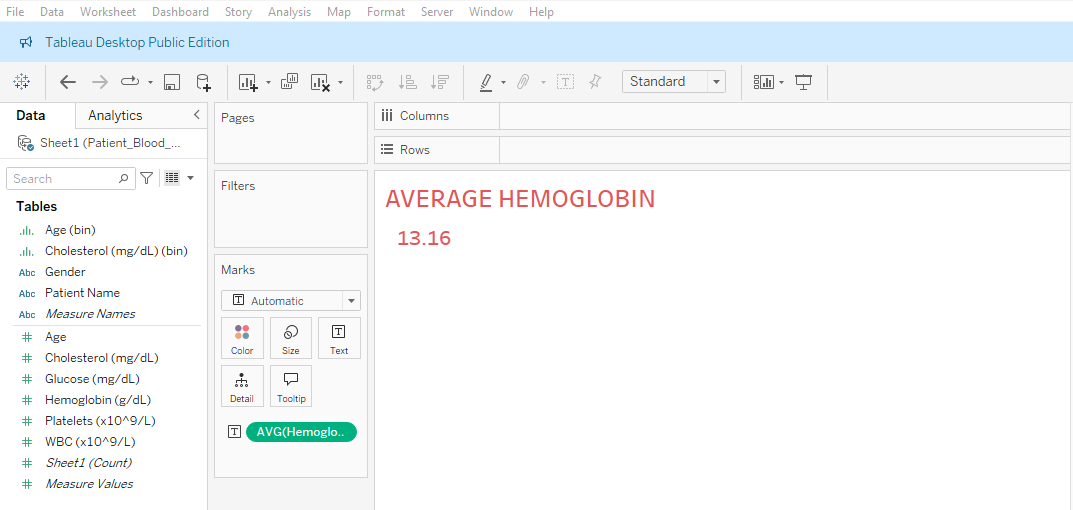
* Duplicat **patient count and** rename **avarage age**
* Remove previous sheet count by draging left side of the screen
* Drag age to rows and convert SUM( age) to AVG(age)
* Right click on **avarage age** format select worksheet -> change font color and size

Then go to allignment -> pane-> center



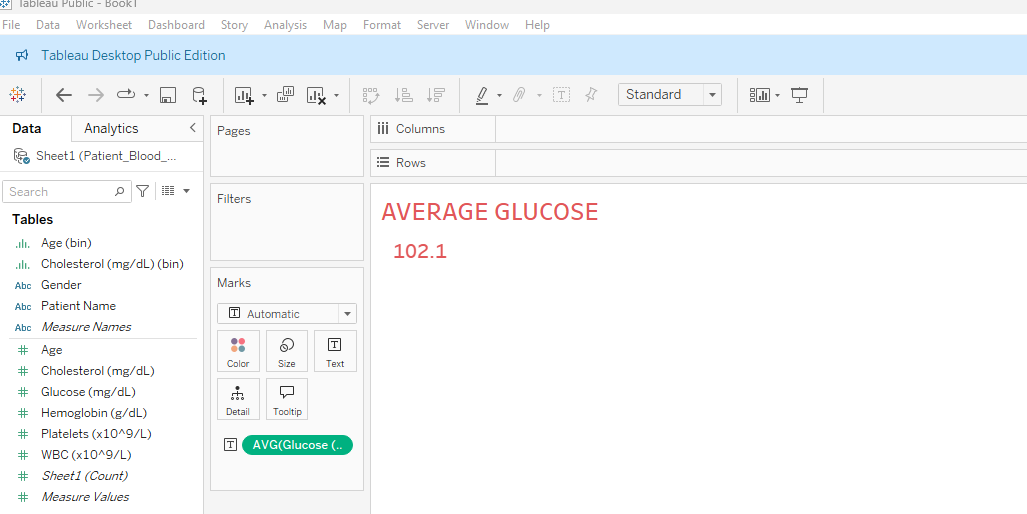
STEP 3:

* Duplicat **average age and** rename **avarage Hemoglobin**
* Remove previous sheet average age by draging left side of the screen
* Drag age to rows and convert SUM( Hemoglobin) to AVG(Hemoglobin)
* Right click on **avarage Hemoglobin** format select worksheet -> change font color and size Then go to allignment -> pane-> center



Step:-4

* Duplicate **average Hemoglobin and** rename **average Glucose**
* Remove previous sheet average Hemoglobin by draging left side of the screen
* Drag age to rows and convert SUM( Glucose) to AVG(Glucose)
* Right click on **avarage Glucose** format select worksheet -> change font color and size Then go to allignment -> pane-> center



Step:-5

* Click on dash board symbol.
* Click on objects->Text and type Patient blood test analysis dash board.
* Right Click on patient blood test analysis dash board ->size-> change maximum size to 950px
* Drag from sheets paitent count,average age, average hemoglobin,average Glucose .
* Select patient count window and right click and select fit-> fit width
* Select Average age window and right click and select fit-> fit width
* Select Average Hemoglobin window and right click and select fit-> fit width
* Select Average Glucose window and right click and select fit-> fit width

Step:6

* Select new sheet and name it as count of patient by gender
* Drag Gender to columns and Sheet1(count) to rows
* Select pie chart
* To label the pie chart drag gender to label and sheet1 (count) to label.
* Go to dash board and from sheets drag patient by gender to the down of the

Patient count.

Step 7:- Select new sheet and name it as Age Distribution.

* Drag Age to columns and Sheet1(count) to rows
* And select histogram
* Left hand side table click on Age(bin)->edit-> change the bin size to 10.
* Go to dash board and from sheets drag Age distribution to the down of the

Avg age.

Step 8:- Select new sheet and name it as WBC VS PATELETS

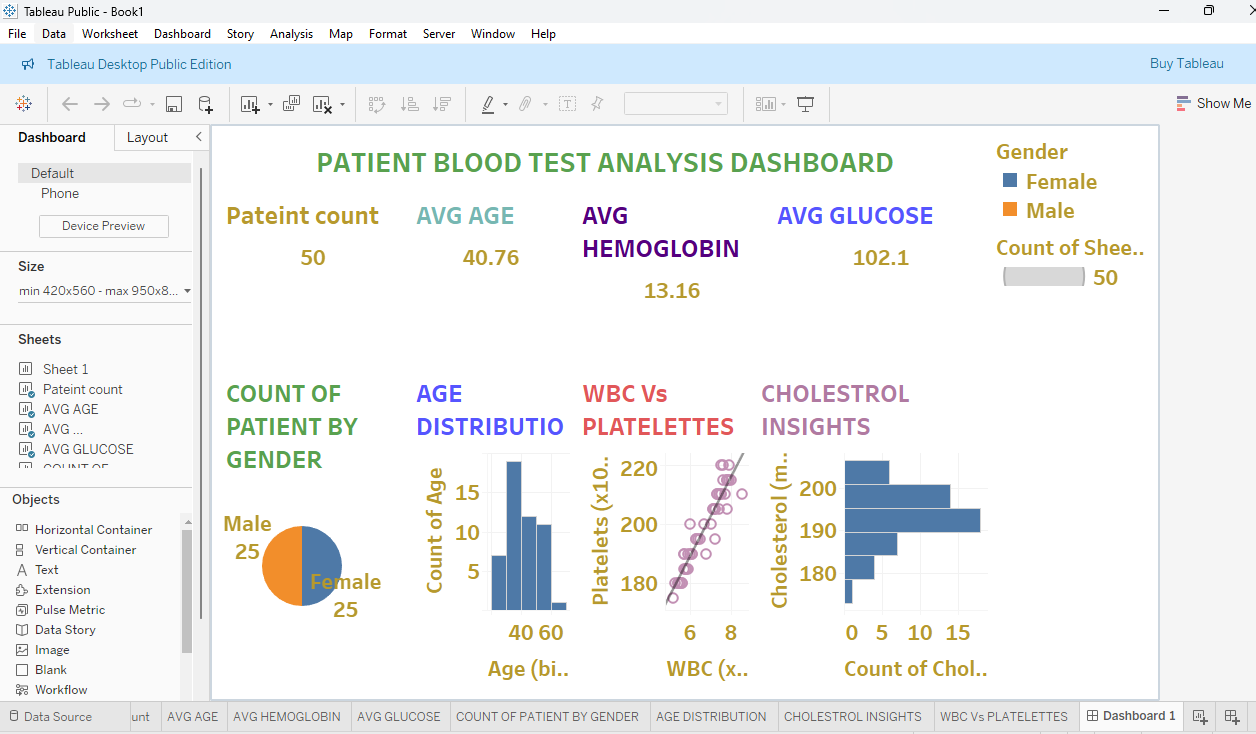
* Drag platelets to columns and WBC to rows or vice versa
* Select Scatter plots
* Go to Analysis on menu bar-->Aggregate measures
* Go to Analysis on menu bar-->trend lines-->show trend lines
* Go to dash board and from sheets drag WBC VS PATELETS to the down of the avg hemoglobin.

Step 9:- Select age distribution sheet and rename it as cholesterol insights

* Drag cholesterol to columns
* Select histogram and click on swap rows and column symbol

Go to dash board and from sheets drag cholesterol insights to the down of the avg glucose.

**Out put:**



**Experiment-9**

**AIM: Extract Sample document and apply following document preprocessing methods:**

**Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.**

**Require Software& Tools: Hadoop in Ubuntu, VM ware work station,Power Pivot(Excel)**

**PROCEDURE:**

To implement a Python program that reads text from a file and applies the following

preprocessing techniques:

1. Tokenization
2. POS Tagging
3. Stop Words Removal
4. Stemming
5. Lemmatization

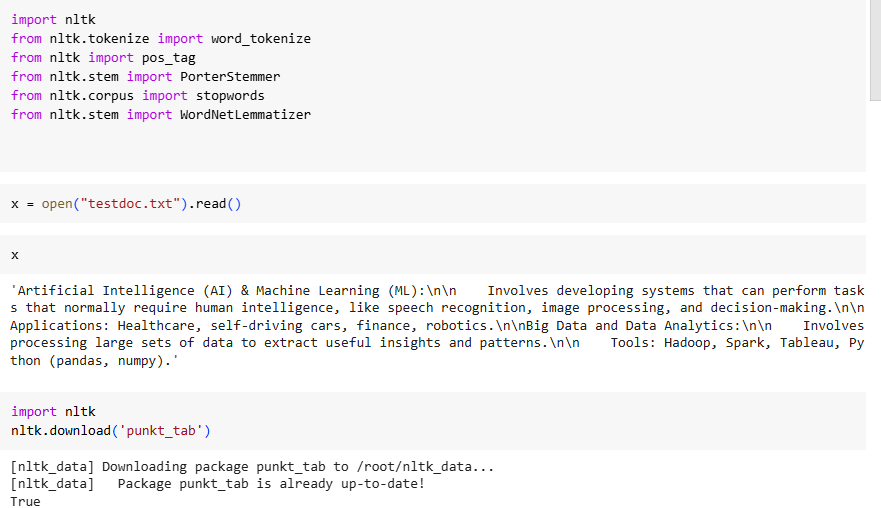
In Natural Language Processing (NLP), text preprocessing is an essential step that ensures raw data is prepared for analysis. The steps involved are:

1. Tokenization: Breaking the text into smaller meaningful units such as words or sentences.
2. POS Tagging: Identifying the part of speech (e.g., noun, verb, adjective) for each token in the text.
3. Stop Words Removal: Removing common words (e.g., *is*, *the*, *and*) that are not significant for the analysis.
4. Stemming: Reducing words to their root form (e.g., *running* → *run*).
5. Lemmatization: Converting words to their dictionary form (e.g., *better* → *good*), considering grammar and context.

**Process**

1. Step 1: Install and import the required libraries (nltk, python-docx).
2. Step 2: Define a function to read text from a .docx file.
3. Step 3: Apply tokenization to the text.
4. Step 4: Perform POS tagging on the tokens.
5. Step 5: Remove stop words using NLTK's predefined stop words list.
6. Step 6: Apply stemming using the Porter Stemmer algorithm.
7. Step 7: Apply lemmatization using WordNet Lemmatizer.
8. Step 8: Print the results for each preprocessing step.

**Code:**

1. 
2. #Tokenization
3. tokens = word\_tokenize(x)
4. print(tokens)
5. 'Artificial', 'Intelligence', '(', 'AI', ')', '&', 'Machine', 'Learning', '(', 'ML', ')', ':', 'Involves', 'developing', 'systems', 'that', 'can', 'perform', 'tasks', 'that', 'normally', 'require', 'human', 'intelligence', ',', 'like', 'speech', 'recognition', ',', 'image', 'processing', ',', 'and', 'decision-making', '.', 'Applications', ':', 'Healthcare', ',', 'self-driving', 'cars', ',', 'finance', ',', 'robotics', '.', 'Big', 'Data', 'and', 'Data', 'Analytics', ':', 'Involves', 'processing', 'large', 'sets', 'of', 'data', 'to', 'extract', 'useful', 'insights', 'and', 'patterns', '.', 'Tools', ':', 'Hadoop', ',', 'Spark', ',', 'Tableau', ',', 'Python', '(', 'pandas', ',', 'numpy', ')', '.']
6. import nltk
7. nltk.download('averaged\_perceptron\_tagger\_eng')
8. [nltk\_data] Downloading package averaged\_perceptron\_tagger\_eng to
9. [nltk\_data] /root/nltk\_data...
10. [nltk\_data] Package averaged\_perceptron\_tagger\_eng is already up-to-
11. [nltk\_data] date!
12. True
13. #POS Tagging
14. postags = pos\_tag(tokens)
15. print(postags)
16. [('Artificial', 'JJ'), ('Intelligence', 'NNP'), ('(', '('), ('AI', 'NNP'), (')', ')'), ('&', 'CC'), ('Machine', 'NNP'), ('Learning', 'NNP'), ('(', '('), ('ML', 'NNP'), (')', ')'), (':', ':'), ('Involves', 'VBZ'), ('developing', 'VBG'), ('systems', 'NNS'), ('that', 'WDT'), ('can', 'MD'), ('perform', 'VB'), ('tasks', 'NNS'), ('that', 'WDT'), ('normally', 'RB'), ('require', 'VBP'), ('human', 'JJ'), ('intelligence', 'NN'), (',', ','), ('like', 'IN'), ('speech', 'NN'), ('recognition', 'NN'), (',', ','), ('image', 'NN'), ('processing', 'NN'), (',', ','), ('and', 'CC'), ('decision-making', 'NN'), ('.', '.'), ('Applications', 'NNS'), (':', ':'), ('Healthcare', 'NNP'), (',', ','), ('self-driving', 'JJ'), ('cars', 'NNS'), (',', ','), ('finance', 'NN'), (',', ','), ('robotics', 'NNS'), ('.', '.'), ('Big', 'NNP'), ('Data', 'NNP'), ('and', 'CC'), ('Data', 'NNP'), ('Analytics', 'NNS'), (':', ':'), ('Involves', 'VBZ'), ('processing', 'VBG'), ('large', 'JJ'), ('sets', 'NNS'), ('of', 'IN'), ('data', 'NNS'), ('to', 'TO'), ('extract', 'VB'), ('useful', 'JJ'), ('insights', 'NNS'), ('and', 'CC'), ('patterns', 'NNS'), ('.', '.'), ('Tools', 'NNS'), (':', ':'), ('Hadoop', 'NNP'), (',', ','), ('Spark', 'NNP'), (',', ','), ('Tableau', 'NNP'), (',', ','), ('Python', 'NNP'), ('(', '('), ('pandas', 'NN'), (',', ','), ('numpy', 'RB'), (')', ')'), ('.', '.')]
17. #Removing stop words
18. stop\_words = set(stopwords.words('english'))
19. print(stop\_words)
20. {'when', 'your', "doesn't", 'at', 'mustn', 'until', 'these', 'own', "that'll", 'for', 'isn', 'what', 'nor', 'how', 'did', 's', "you're", 'yourselves', 'wouldn', 'same', 'those', 'below', 'about', "they're", 'but', 'only', 'was', "needn't", 'will', 'so', 'weren', 'by', 'to', 'been', 'on', 'further', 'her', 'against', "shouldn't", 't', 'while', 'after', 'do', 'didn', 'again', 'being', 'ma', "you've", 'can', 'i', 'such', 'o', 'who', "they'll", 'his', 'does', 've', 'here', 'theirs', "don't", 'very', 'other', "wasn't", "we've", 'haven', "shan't", "haven't", "i'd", "hasn't", 'doesn', 'out', 'yours', 'because', 'that', "isn'
21. li = []
22. for words in tokens:
23. if words not in stop\_words:
24. li.append(words)
25. print(li)
26. 'Artificial', 'Intelligence', '(', 'AI', ')', '&', 'Machine', 'Learning', '(', 'ML', ')', ':', 'Involves', 'developing', 'systems', 'perform', 'tasks', 'normally', 'require', 'human', 'intelligence', ',', 'like', 'speech', 'recognition', ',', 'image', 'processing', ',', 'decision-making', '.', 'Applications', ':', 'Healthcare', ',', 'self-driving', 'cars', ',', 'finance', ',', 'robotics', '.', 'Big', 'Data', 'Data', 'Analytics', ':', 'Involves', 'processing', 'large', 'sets', 'data', 'extract', 'useful', 'insights', 'patterns', '.', 'Tools', ':', 'Hadoop', ',', 'Spark', ',', 'Tableau', ',', 'Python', '(', 'pandas', ',', 'numpy', ')', '.']
27. 0s
28. #Stemming
29. ps = PorterStemmer()
30. stemlist = []
31. for words in li:
32. stemlist.append([words, ps.stem(words)])
33. print(stemlist)
34. ['Artificial', 'artifici'], ['Intelligence', 'intellig'], ['(', '('], ['AI', 'ai'], [')', ')'], ['&', '&'], ['Machine', 'machin'], ['Learning', 'learn'], ['(', '('], ['ML', 'ml'], [')', ')'], [':', ':'], ['Involves', 'involv'], ['developing', 'develop'], ['systems', 'system'], ['perform', 'perform'], ['tasks', 'task'], ['normally', 'normal'], ['require', 'requir'], ['human', 'human'], ['intelligence', 'intellig'], [',', ','], ['like', 'like'], ['speech', 'speech'], ['recognition', 'recognit'], [',', ','], ['image', 'imag'], ['processing', 'process'], [',', ','], ['decision-making', 'decision-mak'], ['.', '.'], ['Applications', 'applic'], [':', ':'], ['Healthcare', 'healthcar'], [',', ','], ['self-driving', 'self-driv'], ['cars', 'car'], [',', ','], ['finance
35. #Lemmatization
36. wl = WordNetLemmatizer()
37. lemilist = []
38. for words in li:
39. lemilist.append([words, wl.lemmatize(words)])
40. print(lemilist)
41. ['Artificial', 'Artificial'], ['Intelligence', 'Intelligence'], ['(', '('], ['AI', 'AI'], [')', ')'], ['&', '&'], ['Machine', 'Machine'], ['Learning', 'Learning'], ['(', '('], ['ML', 'ML'], [')', ')'], [':', ':'], ['Involves', 'Involves'], ['developing', 'developing'], ['systems', 'system'], ['perform', 'perform'], ['tasks', 'task'], ['normally', 'normally'], ['require', 'require'], ['human', 'human'], ['intelligence', 'intelligence'], [',', ','], ['like', 'like'], ['speech', 'speech'], ['recognition', 'recognition'], [',', ','], ['image', 'image'], ['processing', 'processing'], [',', ','], ['decision-making', 'decision-making'], ['.', '.'], ['Applications', 'Applications'], [':', ':'], ['Healthcare', 'Healthcare'], [',', ','], ['self-driving', 'self-driving'], ['cars', 'car'], [',', ','], ['finance', 'finance'], [',', ','], ['robotics', 'robotics'], ['.', '.'], ['Big', 'Big'], ['Data', 'Data'], ['Data', 'Data'], ['Analytics', 'Analytics'], [':', ':'], ['Involves', 'Involves'], ['processing', 'processing'], ['large', 'large'], ['sets', 'set'], ['data', 'data'], ['extract', 'extract'], ['useful', 'useful'], ['insights', 'insight'], ['patterns', 'pattern'], ['.', '.'], ['Tools', 'Tools'], [':', ':'], ['Hadoop', 'Hadoop'], [',', ','], ['Spark', 'Spark'], [',', ','], ['Tableau', 'Tableau'], [',', ','], ['Python', 'Python'], ['(', '('], ['pandas', 'panda'], [',', ','], ['numpy', 'numpy'], [')', ')'], ['.', '.']]

**Experiment-10**

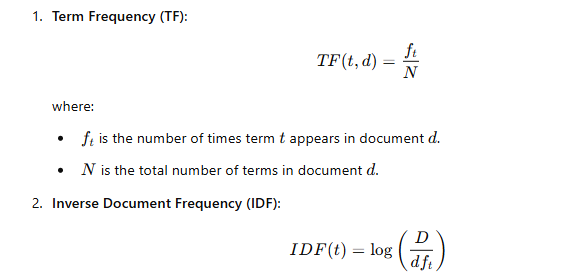
**AIM:** Create representation of document by calculating Term Frequency and Inverse Document Frequency.

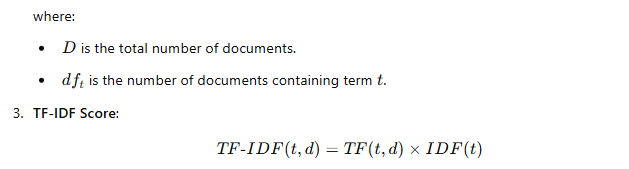
**Require Software& Tools: Hadoop in Ubuntu, VM ware work station,Power Pivot(Excel)**

**PROCEDURE:**

[10 .Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure used to evaluate the importance of a word in a document relative to a collection (or corpus) of documents. The formula for calculating TF-IDF is:





from sklearn.feature\_extraction.text import TfidfVectorizer

def compute\_tfidf\_from\_file(file\_path):

# Open and read the file

with open(file\_path, 'r') as file:

documents = file.readlines()

# Create TF-IDF vectorizer and compute TF-IDF matrix

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform(documents)

feature\_names = vectorizer.get\_feature\_names\_out()

# Print out the TF-IDF score for each word in each document

for i, doc in enumerate(documents):

print(f"Document {i + 1} TF-IDF scores:")

for word, score in zip(feature\_names, tfidf\_matrix[i].toarray()[0]):

if score > 0: # Only print words with a non-zero score

print(f" {word}: {score:.4f}")

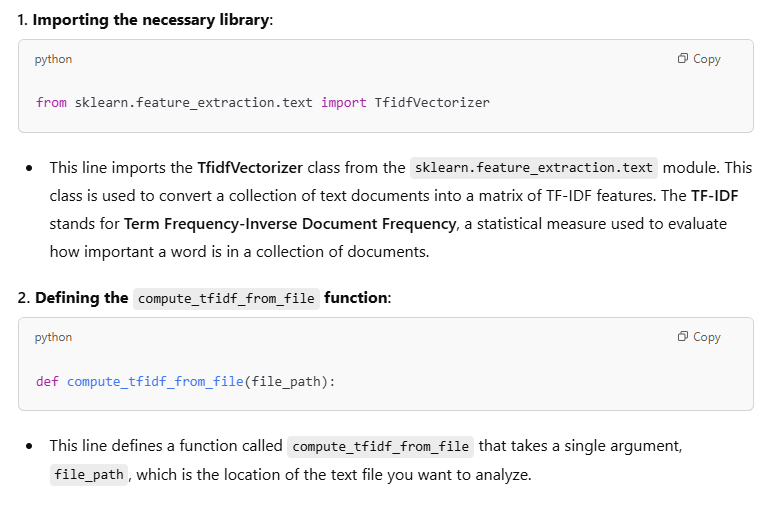
print("\n")

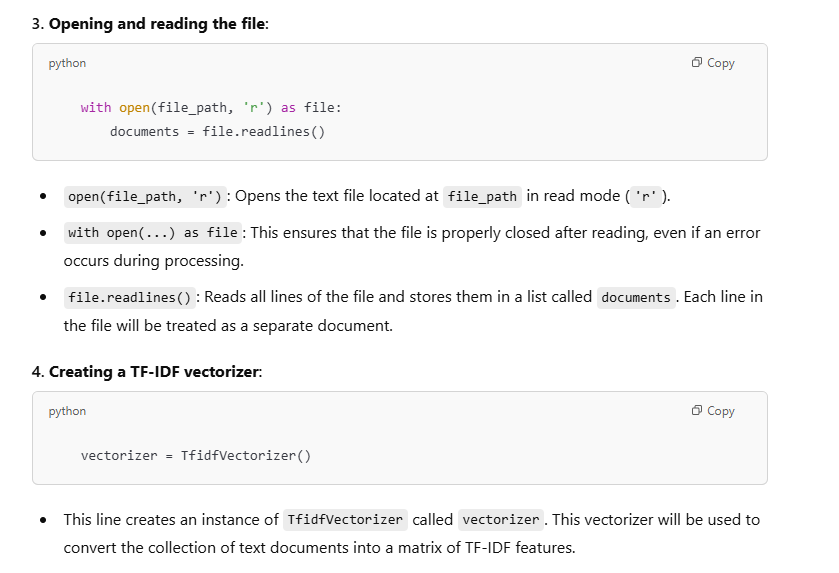
# Example usage

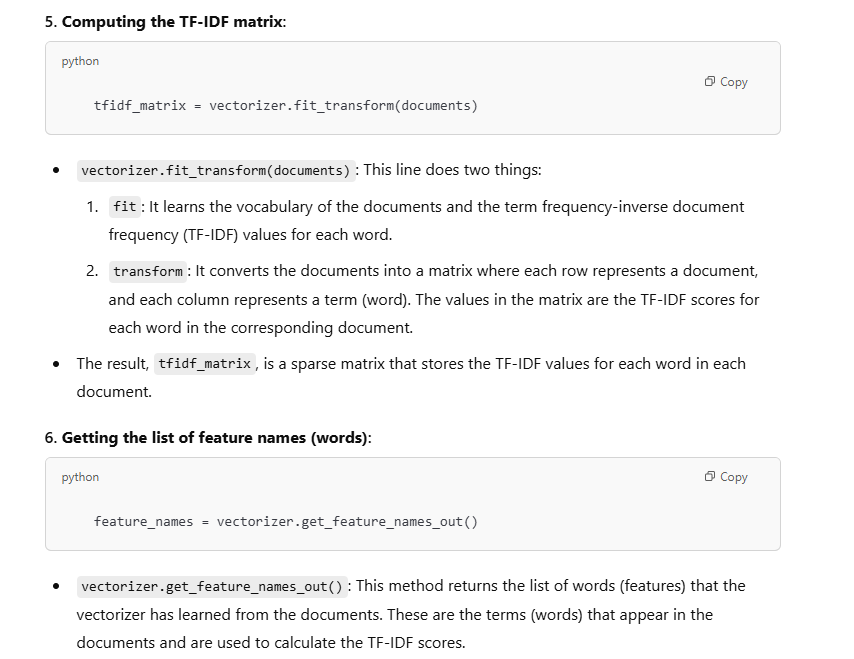
file\_path = "doc.txt" # Replace this with the path to your text file

compute\_tfidf\_from\_file(file\_path)

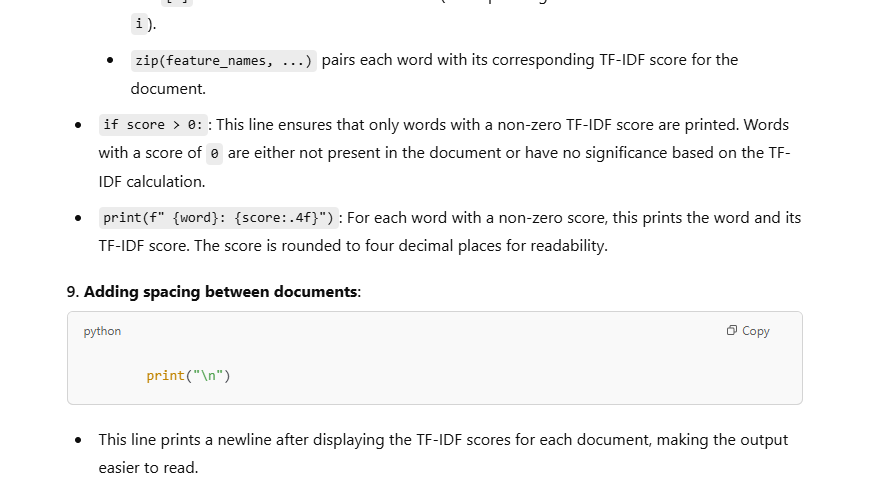


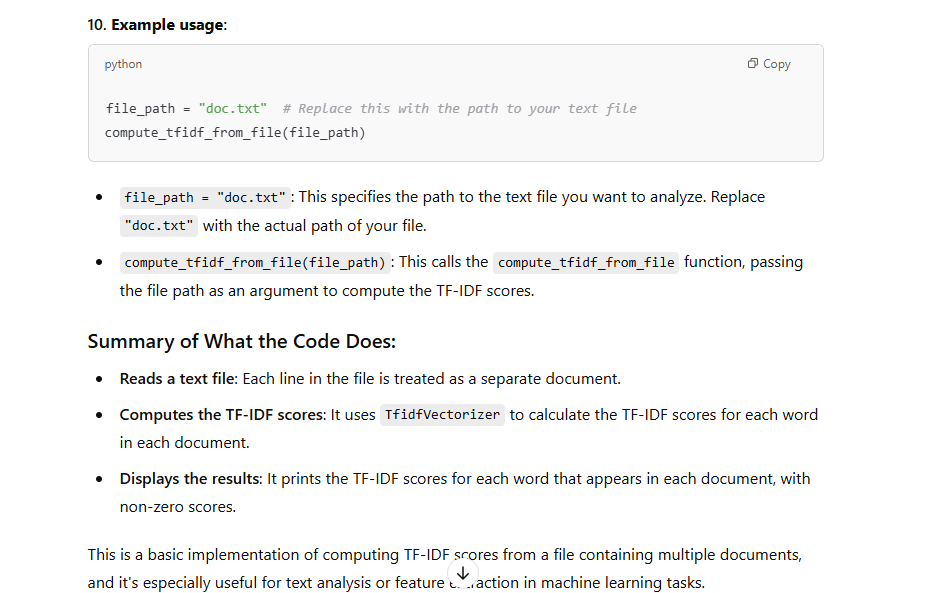




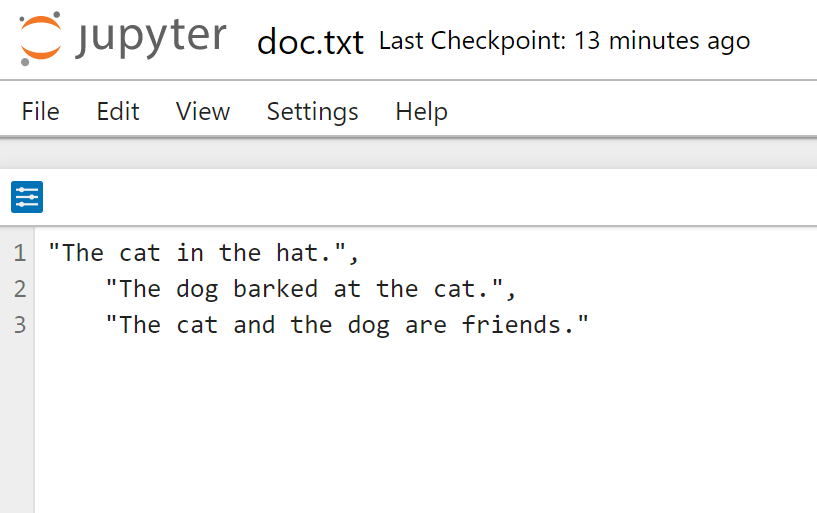




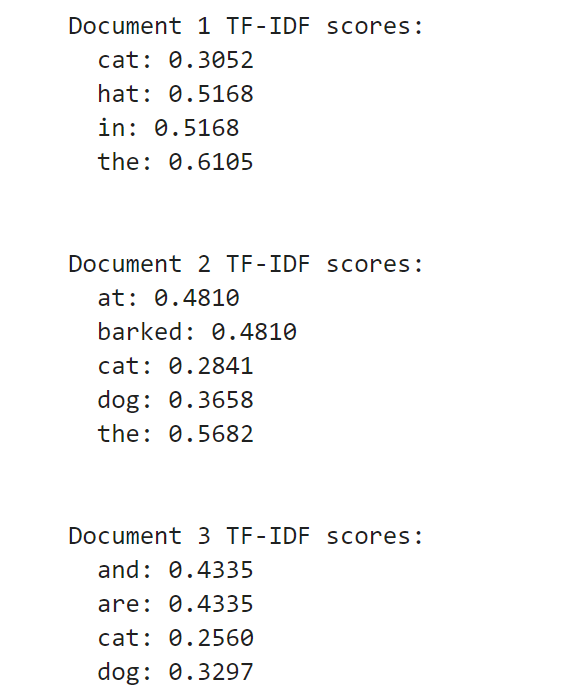




**input**



**Output:**

****

**Experiment-11**

**AIM**: Use the inbuilt dataset 'titanic' (Use the Seaborn library). Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram

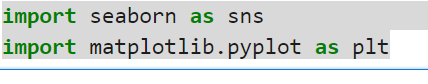
To visualize the distribution of ticket fares for passengers aboard the Titanic using a histogram.

**Require Software& Tools: Anaconda(jupyternotebook)**

**PROCEDURE:**

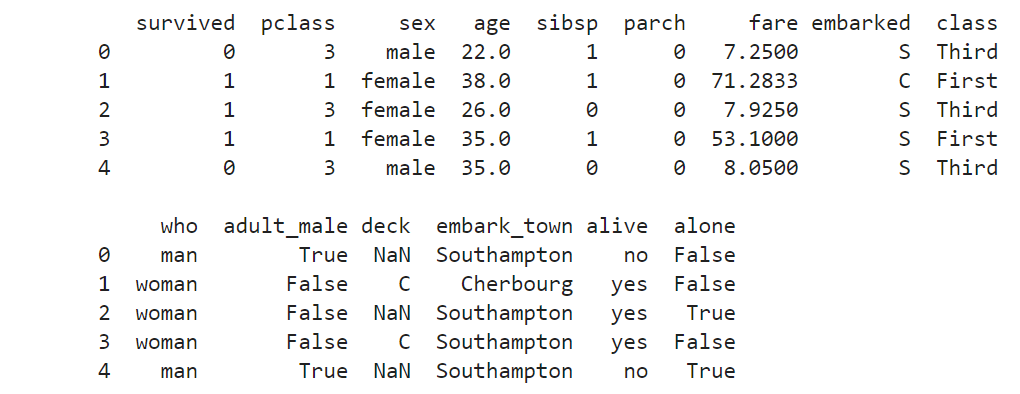
1. Import necessary libraries:
   * Import seaborn (for loading the Titanic dataset and plotting) and matplotlib.pyplot (for customizing the plot and displaying it).
2. Load the Titanic dataset:
   * Seaborn provides an inbuilt Titanic dataset, which contains information about passengers such as age, class, fare, and survival status. We'll use this dataset to explore the fare distribution.
3. Preview the data:
   * Use the head() function to check the first few rows of the dataset. This helps ensure that the data is loaded correctly and gives an overview of the available columns.
4. Plotting the histogram:
   * We will use Seaborn's histplot() function to plot the distribution of the "fare" column. This function will create a histogram and optionally include a Kernel Density Estimate (KDE) curve to visualize the data distribution.
5. Customize the plot:
   * Add a title, x-axis label, and y-axis label to make the plot informative.
6. Display the plot:
   * Use plt.show() to display the generated histogram.
7. Import necessary libraries:
   * Import seaborn (for loading the Titanic dataset and plotting) and matplotlib.pyplot (for customizing the plot and displaying it).
8. Load the Titanic dataset:
   * Seaborn provides an inbuilt Titanic dataset, which contains information about passengers such as age, class, fare, and survival status. We'll use this dataset to explore the fare distribution.
9. Preview the data:
   * Use the head() function to check the first few rows of the dataset. This helps ensure that the data is loaded correctly and gives an overview of the available columns.
10. Plotting the histogram:
    * We will use Seaborn's histplot() function to plot the distribution of the "fare" column. This function will create a histogram and optionally include a Kernel Density Estimate (KDE) curve to visualize the data distribution.
11. Customize the plot:
    * Add a title, x-axis label, and y-axis label to make the plot informative.
12. Display the plot:
    * Use plt.show() to display the generated histogram.

SOURCE CODE AND OUTPUT:



titanic=sns.load\_dataset('titanic')

print(titanic.head())



plt.figure(figsize=(8,6))

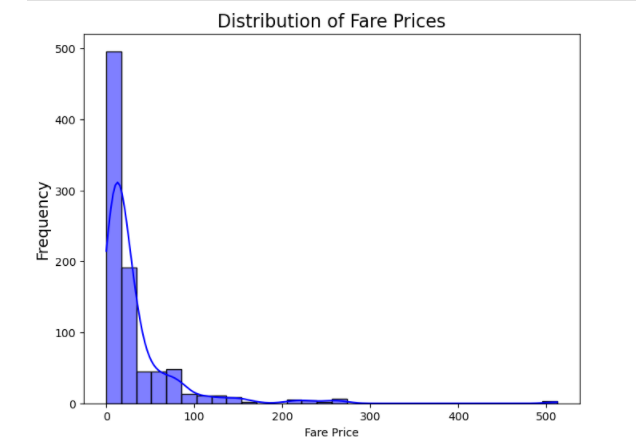
sns.histplot(titanic['fare'],kde=True,bins=30,color='blue')

plt.title('Distribution of Fare Prices',fontsize=16)

plt.xlabel("Fare Price",fontsize=10)

plt.ylabel('Frequency',fontsize=14)

plt.show()



**EXPERIMENT-12:**

**Aim:**Use R-Project to carry out statistical analysis of big data

**PROCEDURE:**

Installation of R and Rstudio

Step 1: sudo apt-get update

sudo apt-get install r-base

Step 2: Installation of R studio

Step 3: step 1 download R studio for Ubuntu

Step 4: step2:sudo dpkg -i rstudio-2022.07.2-576-amd64 . deb

Step 5*:* step 3 :sudo apt install –f

Step 6: Open rstudio

Step 7: procedure:-->install.packages("gapminder")-->library(gapminder)

Step 8: >data(gapminder)

Step 9:boxplot(lifeExp)

Source Code:

# Install necessary packages

install.packages(c("dplyr", "ggplot2", "gapminder"))

library(dplyr)

library(ggplot2)

library(gapminder)

#Load the Gapminder Dataset

data(gapminder)

head(gapminder) # View the first few rows

# Exploring the Data

summary(gapminder)

# Filter for the year 2007 (the most recent in the dataset)

gapminder\_2007 <- filter(gapminder, year == 2007)

# View the data for 2007

head(gapminder\_2007)

#Statistical Analysis

#Average Life Expectancy by Continent

life\_expectancy\_by\_continent <- gapminder %>%

group\_by(continent) %>%

summarize(avg\_lifeExp = mean(lifeExp, na.rm = TRUE))

print(life\_expectancy\_by\_continent)

# GDP and Life Expectancy Correlation

cor(gapminder$gdpPercap, gapminder$lifeExp, use = "complete.obs")

#Linear Regression (Life Expectancy ~ GDP)

model <- lm(lifeExp ~ gdpPercap, data = gapminder)

summary(model)

#Visualization

#GDP vs. Life Expectancy

ggplot(gapminder, aes(x = gdpPercap, y = lifeExp)) +

geom\_point(aes(color = continent), alpha = 0.7) +

scale\_x\_log10() +

labs(title = "GDP vs Life Expectancy") +

theme\_minimal()

#Life Expectancy Over Time (by Continent)

ggplot(gapminder, aes(x = year, y = lifeExp, color = continent)) +

geom\_line() +

labs(title = "Life Expectancy Over Time") +

theme\_minimal()

#GDP vs. Population (2007)

gapminder\_2007 <- filter(gapminder, year == 2007)

ggplot(gapminder\_2007, aes(x = gdpPercap, y = pop)) +

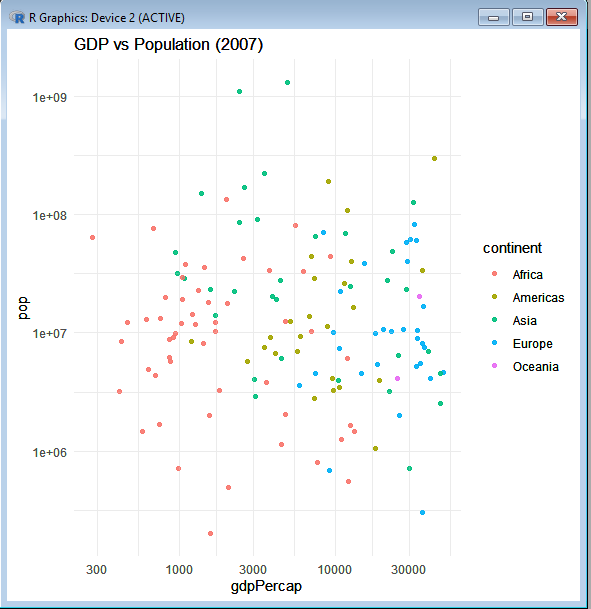
geom\_point(aes(color = continent), alpha = 0.7) +

scale\_x\_log10() + scale\_y\_log10() +

labs(title = "GDP vs Population (2007)") +

theme\_minimal()

**Ouput:**

****

**EXPERIMENT-13:**

**Aim:** Use R-Project for data visualization of social media data.

**Require Software& Tools: R studio**

**PROCEDURE:**

library(rtweet)

# Replace with your actual Twitter API credentials

token <- create\_token(

app = "MyTwitterApp",

consumer\_key = "abcd1234xyz",

consumer\_secret = "abcd5678xyz",

access\_token = "1234-xyz5678",

access\_secret = "xyz1234abcd"

)

**Expected Output in R Console:**

If authentication is successful, you will see something like:

<Token>

<rtweet>

app: MyTwitterApp

auth\_type: oauth1.0

**Checking Twitter API Rate Limits**

To verify your API access, run:

rate\_limit()

**Expected Output:**

query n remaining reset

1 search\_tweets 180 179 900

2 get\_timeline 900 890 900

This shows how many API requests you have left.

Fetching Tweets

Now, retrieve tweets using:

tweets <- search\_tweets("#DataScience", n = 5, lang = "en", include\_rts = FALSE)

head(tweets)

**Expected Output in R Console**:

created\_at screen\_name text

1 2025-03-26 12:45 DataGuy123 "I love #DataScience!"

2 2025-03-26 12:44 MLExpert99 "Machine learning is amazing!"

3 2025-03-26 12:42 AI\_Fanatic "AI is the future #BigData"

This confirms that Twitter is returning real tweets.

Word Cloud Visualization:

library(wordcloud)

wordcloud(words = tweet\_corpus, min.freq = 2, max.words = 100, colors = brewer.pal(8, "Dark2"))

## **Sentiment Analysis Bar Chart**

ggplot(sentiment\_df, aes(x = sentiment, y = score, fill = sentiment)) +

geom\_bar(stat = "identity") +

theme\_minimal() +

labs(title = "Sentiment Analysis of Tweets", x = "Sentiment", y = "Score")