PRACTICAL JOURNAL

in

APPLIED BIG DATA ANALYTICS

Submitted by

KSMSCIT010 HIMANSHU SINGH

for the award of the Degree of

MASTERS OF SCIENCE (INFORMATION TECHNOLOGY)

PART – II

DEPARTMENT OF INFORMATION TECHNOLOGY
KISHINCHAND CHELLARAM COLLEGE
(Affiliated to University of HSNCU)
MUMBAI,400020
MAHARASHTRA
2024-25

SUBJECT CODE – BIT614D

Applied Big Data Analytics

KISHINCHAND CHELLARAM COLLEGE

CHURCHGATE, MUMBAI – 400 020.

DEPARTMENT OF INFORMATION TECHNOLOGY

M.SC.I. T PART- II

CERTIFICATE

This is to certify that the Practical conducted by Mr. Himanshu Singh for M.Sc. (IT) Part- II Semester- IV, Seat No: **KSMSCIT010** at Kishinchand Chellaram College in partial fulfillment for the MASTERS OF SCIENCE (INFORMATION TECHNOLOGY). Degree Examination for semester IV has been periodically examined and signed, and the course of term work has been satisfactorily carried out for the year 2024 - 2025. 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

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

INDEX

Sr. No	Topic	Date	Signature
1	Recommendation System	08-12-2024	
2	Processing Data from Social Media Platforms (Raw Data Fetching)	29-12-2024	
3	Collecting and Ingesting Data into Big Data Storage using Data Access Connectors	29-12-2024	
4	Genome Analysis: Calculating GC Content and Its Significance	12-01-2025	
5	MVDI and Sentinel Hub: Remote Sensing Data Processing	19-01-2025	
6	Exploratory Data Analysis (EDA) on E-Commerce Reviews	16-02-2025	
7	Sentiment Analysis on IMDb Dataset	02-03-2025	
8	Python/R Program for Selecting Billboard Content from Given Data	23-03-2025	
9	Data Visualization using PYGAL	23-03-2025	
10	Processing Balance Sheet Data to Ensure Quality Filtering	23-03-2025	
11	Working with MongoDB	23-03-2025	

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 1

Recommendation System

Aim: Recommendation System.

Code:

```
#Importing the required packages

import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

df1 = pd.read_csv(r'movies.csv')
df2 = pd.read_csv(r'ratings.csv')
df = df2.merge(df1, left_on='movieId', right_on='movieId', how='left')
df

del df['timestamp']
del df['genres']
df.head()
user_movie_matrix = pd.pivot_table(df, values = 'rating', index='movieId', columns = 'userId')
user_movie_matrix
user_movie_matrix = user_movie_matrix.fillna(0)
user_movie_matrix.head()

#user-based collaborative filtering

user_user_matrix = user_movie_matrix.corr(method='pearson')
user_user_matrix
```

```
#Extarcing top 10 similar users for User2 by sorting them in descending order based on their similarties

user_user_matrix.loc[2].sort_values(ascending=False).head(10)
df_2 = pd.DataFrame(user_user_matrix.loc[2].sort_values(ascending=False).head(10))
df_2 = df_2.reset_index()
df_2.columns = ['userId', 'similarity']
df_2 = df_2.drop((df_2[df_2['userId'] ==2]).index)
df_2

#Now we are creating a new DF which has all the similar users and their rated movies

final_df = df_2.merge(df, left_on='userId', right_on='userId', how='left')
final_df
final_df['score'] = final_df['similarity']*final_df['rating']
final_df
watched_df = df[df['userId'] == 2]
watched_df
cond = final_df['movieId'].isin(watched_df['movieId'])
final_df.drop(final_df[cond].index, inplace = True)
recommended_df = final_df.sort_values(by = 'score', ascending = False)['title'].head(10)
recommended_df = recommended_df.reset_index()
del recommended_df['index']
recommended_df
```

Output:

KSMSCIT010 – Himanshu Singh

	title
0	Reservoir Dogs (1992)
1	Pulp Fiction (1994)
2	Trainspotting (1996)
3	Seven (a.k.a. Se7en) (1995)
4	American History X (1998)
5	The Butterfly Effect (2004)
6	Lord of the Rings: The Return of the King, The...
7	City of God (Cidade de Deus) (2002)
8	Lord of the Rings: The Two Towers, The (2002)
9	Beautiful Mind, A (2001)

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 2

Processing data generated by social media platform (raw data fetching)

Aim: data generated by social media platform (raw data fetching).

Code:

```
#Importing the required packages

import os
import sys
from pathlib import Path


import pandas as pd
import numpy as np


# Options for pandas
pd.options.display.max_columns = 50
pd.options.display.max_rows = 30
pd.options.display.float_format = '{:,.4f}'.format


# autoreload extension
%load_ext autoreload
%autoreload 2


sys.path.insert(0, str(Path.cwd().parent))


import ast
from PIL import Image
REPO_PATH = Path.cwd()
```

```
report_path = '/content/BigData_sizes.csv'
df = pd.read_csv(report_path, converters={'logo_path': str, 'description_html': str,
                                          'logo_rendering': ast.literal_eval,
                                          'arrow_specs': ast.literal_eval
                                          })


df


sort_idx = df.sort_values('size_PB').index
df.loc[sort_idx]


import plotly.graph_objects as go


big_lbls = df.loc[df.size_label.str.contains('EB'), 'size_label']
df.loc[df.size_label.str.contains('EB'), 'size_label'] = ''
```

```
layout = {
    'template': "plotly_white",
    'paper_bgcolor': 'rgba(0,0,0,0)',
    'plot_bgcolor': 'rgba(0,0,0,0)',
    'title': {
        'x': 0.5, 'xanchor': 'center'
    },
    'font': dict(
        family="Helvetica",
        size=18,
    ),
    'showlegend': False,
    'autosize': False,
    'width': 1400,
    'height': 720,
    'margin': dict(l=0, r=0, t=0, b=0),
}
```

```
# Bubble plot
fig = go.Figure(data=[
    go.Scatter(
        x=df.x, y=df.size_PB,
        #         x0=1,dx=6,
        mode='markers+text',
        marker=dict(
            size=df.area_size,
            color=df.color,
            opacity=[0.7]*(df.shape[0]-1) + [0.4],
            sizemin=12,
            sizemode='area',
            sizeref=2. * df.area_size.max() / (840 ** 2)
        ),
        text=df.size_label.str.replace('yr', 'y'),
        textposition='bottom center',
        textfont=dict(size=14),
    )
]
)
```

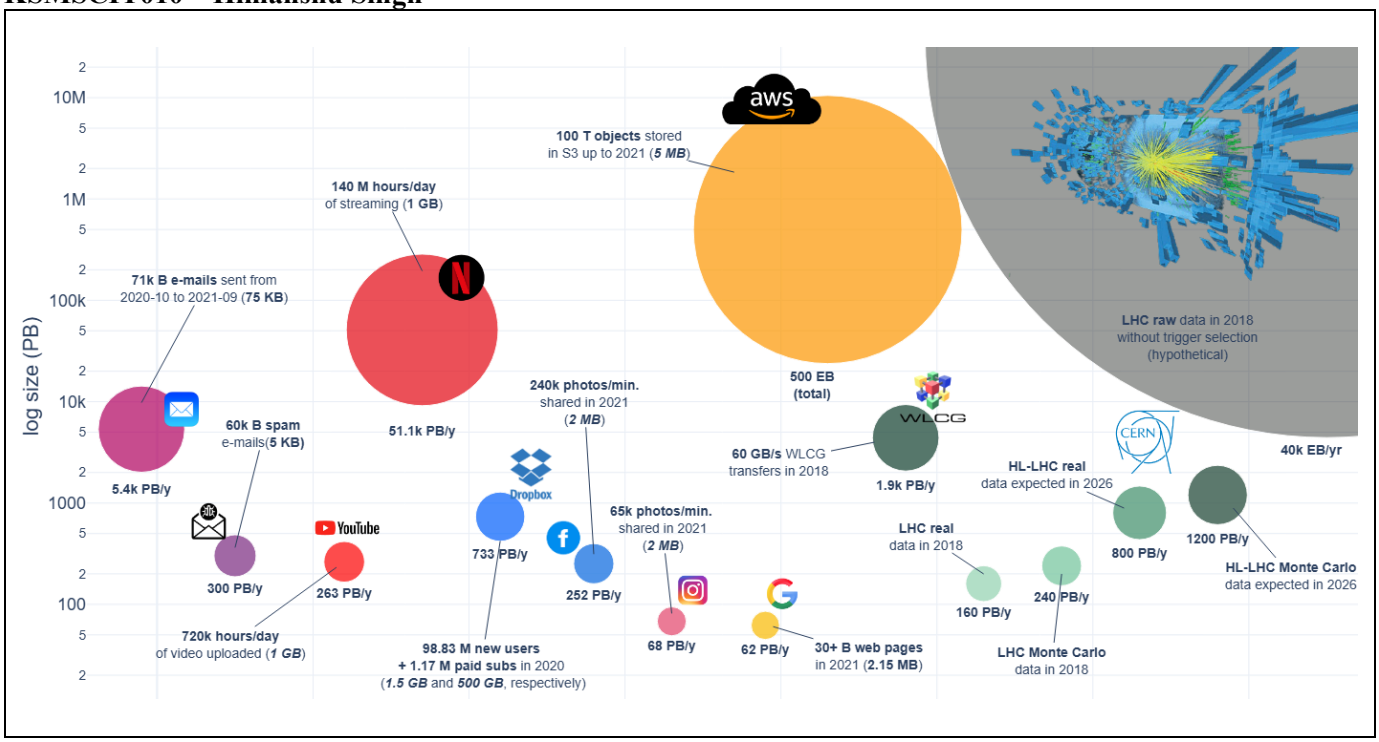
```
 # Big bubbles labels
 fig.add_trace(go.Scatter(
     x=[42, 74, 71],
     y=[20000, 3800, 25],
     mode="text",
     text=big_lbls,
     textposition="bottom center",
     textfont=dict(size=[14, 14, 18])
 ))
```

```
# Image annotations
logos_path = REPO_PATH / "/content/"


for v in df[['logo_path', 'logo_rendering']].itertuples():
    if not v.logo_path:
        continue
    src = Image.open(logos_path / v.logo_path)  # logos_path + v.logo_path


    xpos, ypos, xs, ys = v.logo_rendering
    fig.add_layout_image(
        dict(
            source=src,
            xref="paper", yref="paper",
            x=xpos, y=ypos,
            sizex=xs, sizey=ys,
            xanchor="right", yanchor="bottom"
        )
    )
```

```
# Text annotations
for v in df[['description_html', 'arrow_specs']].itertuples():
    if not v.arrow_specs:
        continue
    xpos, ypos, xlen, ylen = v.arrow_specs
    fig.add_annotation(
        xref="x", yref="y domain",
        x=xpos, y=ypos,
        ax=xlen, ay=ylen,
        text=v.description_html,
        showarrow=True,
        arrowhead=0,
        font=dict(size=14),
    )
# Layout
fig.update_layout(
    yaxis=dict(
        type="log",
        range=[1, 7.5],
        visible=True,
    ),

    yaxis_title="log size (PB)",
    xaxis_title="source",
    xaxis=dict(
        range=[-4.5, df.x.max()+2],
        visible=True,
        showticklabels=False,
    ),
)


fig.update_layout(layout)


fig.show()
```

Output:

KSMSCIT010 – Himanshu Singh



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 3

Collecting and ingesting data from various sources into the big data storage using Data Access Connectors

Aim: Collecting and ingesting data from various sources into the big data storage using Data Access Connectors.

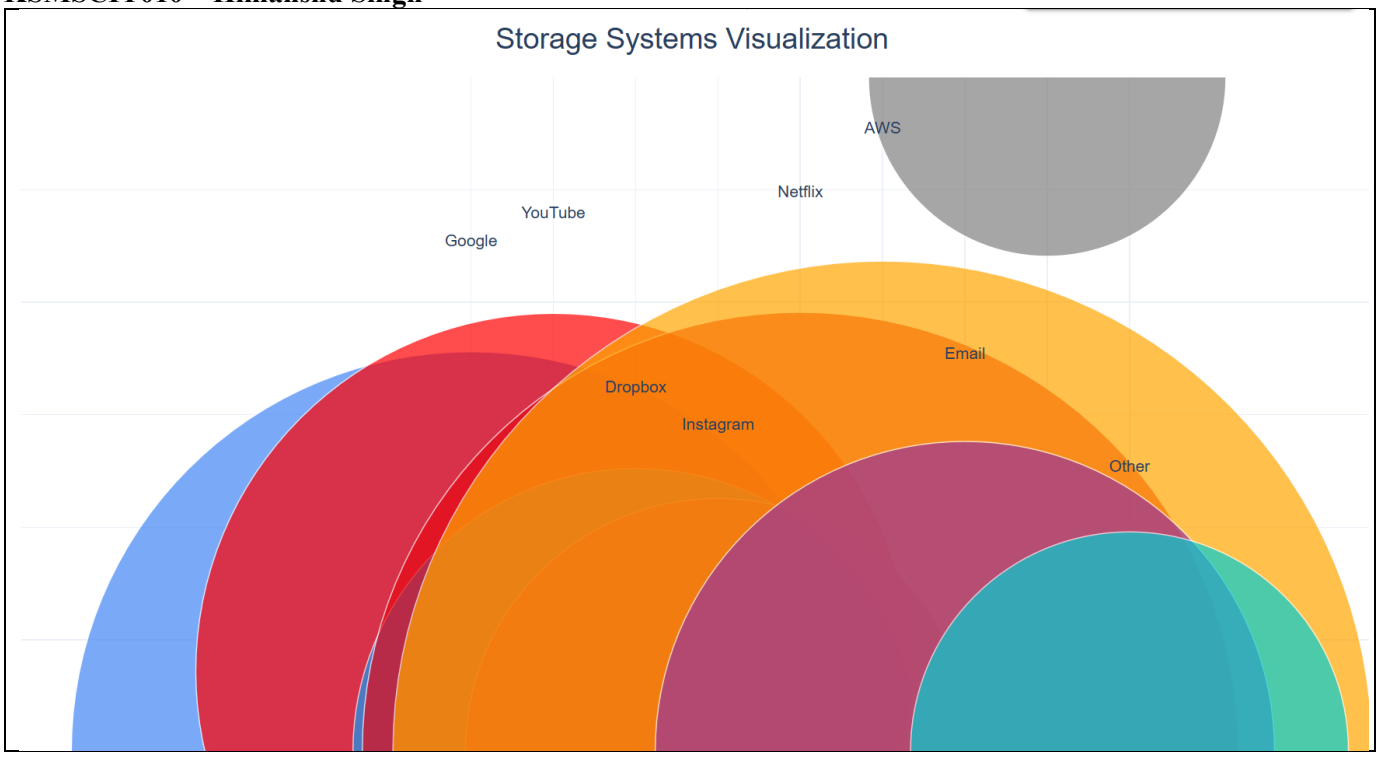
Code:

```
import pandas as pd
import plotly.graph_objects as go

# Create the dataset
data = {
    'source': ['Google', 'YouTube', 'Dropbox', 'Instagram', 'Netflix', 'AWS', 'Email', 'Spam', 'Other'],
    'size_PB': [30, 7200, 60, 65, 140, 100, 71, 60000, 98.83],
    'area_size': [1000, 800, 500, 400, 1200, 1500, 600, 200, 300],
    'color': ['#4285F4', '#FF0000', '#1098F7', '#E1306C', '#E50914', '#FFA500', '#993399', '#808080', '#00CED1'],
    'image_link': [
        '/content/google_icon.png',  # Replace with actual image paths
        '/content/youtube_icon.png',
        '/content/dropbox_icon.png',
        '/content/instagram_icon.png',
        '/content/netflix_icon.png',
        '/content/aws_icon.png',
        '/content/email_icon.png',
        '/content/spam_icon.png',
        '/content/other_icon.png'
    ]
}
df = pd.DataFrame(data)

# Calculate sizeref
sizeref = 2. * df['area_size'].max() / (840 ** 2)

# Prepare the layout for the plot
layout = {
    'template': "plotly_white",
    'paper_bgcolor': 'rgba(0,0,0,0)',
    'plot_bgcolor': 'rgba(0,0,0,0)',
    'title': {
        'text': "Storage Systems Visualization",
        'x': 0.5, 'xanchor': 'center'
    },
    'font': dict(
        family="Helvetica",
        size=18,
    ),
    'showlegend': False,
    'autosize': False,
    'width': 1400,
    'height': 720,
    'margin': dict(l=0, r=0, t=50, b=0),
    'images': []
}
```

```
# Add images to the layout
for i in range(len(df)):
    image_layout = {
        'source': df['image_link'][i],
        'xref': 'x',
        'yref': 'y',
        'x': df['source'][i],
        'y': df['size_PB'][i],
        'sizex': df['area_size'][i] / max(df['area_size']) * 0.2,  # Adjust sizing as needed
        'sizey': df['area_size'][i] / max(df['area_size']) * 0.2,  # Adjust sizing as needed
        'opacity': 0.7,
        'layer': 'above'
    }
    layout['images'].append(image_layout)
```

```
# Create a bubble plot
fig = go.Figure(data=[
    go.Scatter(
        x=df['source'],
        y=df['size_PB'],
        mode='markers+text',
        marker=dict(
            size=df['area_size'],
            color=df['color'],
            opacity=0.7,
            sizemode='area',
            sizeref=sizeref
        ),
        text=df['source'],
        textposition='top center',
        textfont=dict(size=14),
    )
])
```

```
# Layout settings
fig.update_layout(
    yaxis=dict(
        type="linear",
        title="Size",
        visible=True,
    ),
    xaxis=dict(
        title="Storage",
        visible=True,
    )
)

fig.update_layout(layout)

# Show the plot
fig.show()
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 4

Genome

Aim: Genome.

Code:

```
#Importing the required packages

!pip install biopython
from  Bio import SeqIO
fp="/content/sample.fasta"
for seq_record in SeqIO.parse(fp, "fasta"):
    print(seq_record.id)
    print(repr(seq_record.seq))
    print(len(seq_record))

def calculate_gc_content(sequence):
    gc_count = sequence.count('G') + sequence.count('C')
    total_count = len(sequence)
    gc_content = (gc_count / total_count) * 100
    return gc_content

sequence="ATGCGTAGCTAGGCTAGCTA"
gc_content = calculate_gc_content(sequence)
print(f"GC Content: {gc_content:.2f}%")
```

Output:

KSMSCIT010 – Himanshu Singh

```
Requirement already satisfied: biopython in /usr/local/lib/python3.11/dist-packages (1.85)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from biopython) (2.0.2)
Genoma_CpI19_Refinada_v2
Seq('GTGTCGGAGGCTCCATCGACATGGAACGAGCGGTGGCAAGAAGTTACTAATGAG...CAC')
600
GC Content: 50.00%
```

Code:

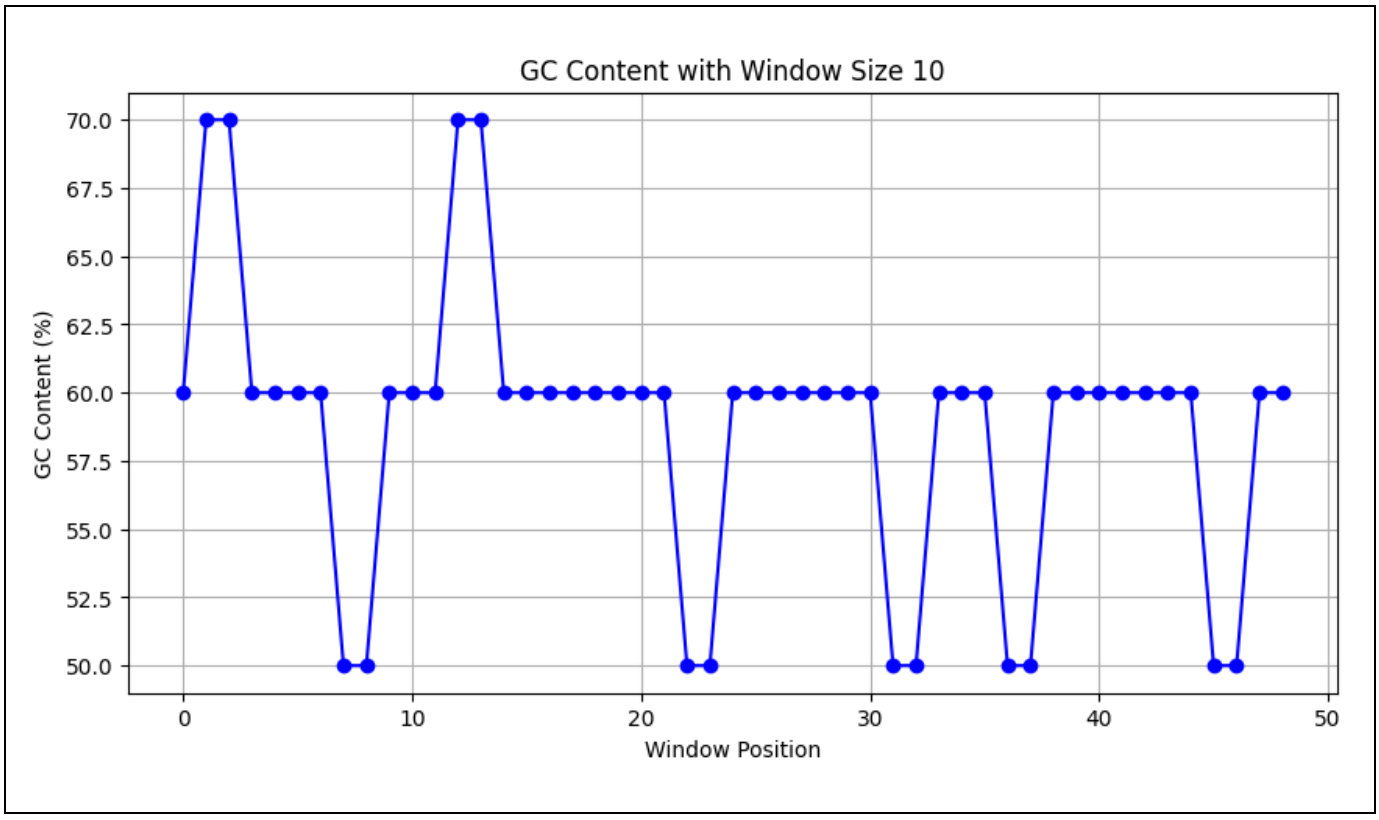
```
#Importing the required packages

from  Bio import SeqIO
import matplotlib.pyplot as plt
fp="/content/sample.fasta"
for seq_record in SeqIO.parse(fp, "fasta"):
    print(seq_record.id)
    print(repr(seq_record.seq))
    print(len(seq_record))

def calculate_gc_content(sequence, window_size):

    gc_content = []
    for i in range(0, len(sequence) - window_size + 1):
        window = sequence[i:i + window_size]
        gc_count = window.count("G") + window.count("C")
        gc_content.append(gc_count / window_size * 100)
    return gc_content


# Example DNA sequence
dna_sequence = "ATGCGCGTAGCTAGGCTACGCGTACGTAGCGTAGCGTAGCTAGGCTAGCGTACGTAGC"
window_size = 10
```

```
# Calculate GC content
gc_values = calculate_gc_content(dna_sequence, window_size)


# Plot GC content
plt.figure(figsize=(10, 5))
plt.plot(range(len(gc_values)), gc_values, marker='o', linestyle='-', color='b')
plt.title(f"GC Content with Window Size {window_size}")
plt.xlabel("Window Position")
plt.ylabel("GC Content (%)")
plt.grid()
plt.show()
```

Output:

KSMSCIT010 – Himanshu Singh

```
Genoma_CpI19_Refinada_v2
Seq('GTGTCGGAGGCTCCATCGACATGGAACGAGCGGTGGCAAGAAGTTACTAATGAG...CAC')
600
```



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 5

Sentinelhub

Aim: Sentinelhub.

Code:

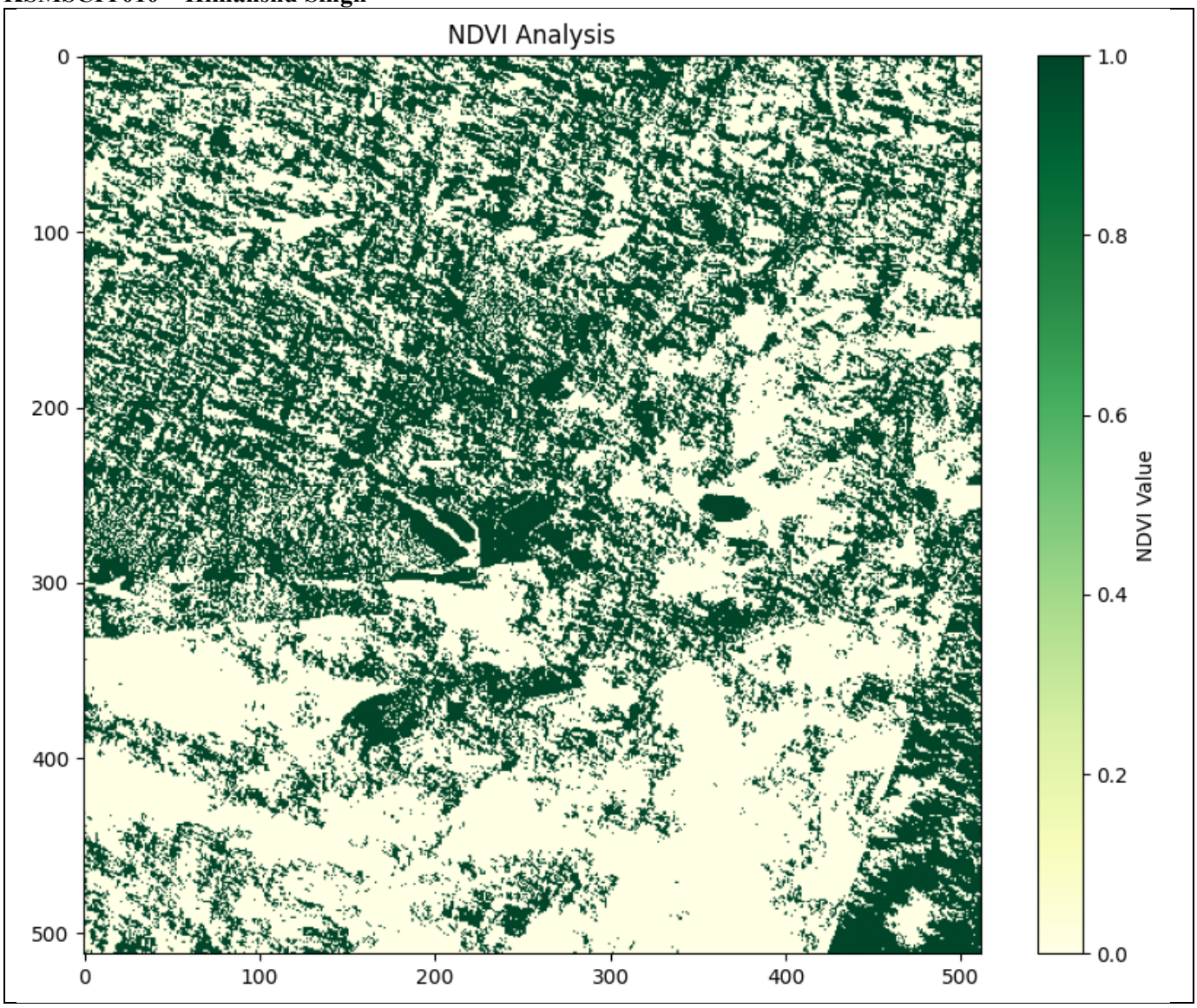
```
!pip install sentinelhub numpy matplotlib geopandas shapely> /dev/null
```

```
import numpy as np
import matplotlib.pyplot as plt

print('\033[1mSentinel Hub API Configuration:\033[0m')
print('\033[1m==================================\033[0m')
# Configure Sentinel Hub API
config = SHConfig()
config.instance_id = 'fb89a462-8e6b-4d31-8c2b-87f6fe40471b'
config.sh_client_id = '4120319c-6ecc-4a34-8903-ad3a76be7ba3'
config.sh_client_secret = 'S3fZ0K23xmxTZhNDc5Zpwe70m8Wz1gcl'

# Define the area of interest (latitude, longitude) for a specific region
area_of_interest = BBox(bbox=(-74.0, 40.5, -73.8, 40.7), crs=CRS.WGS84)  # Example: NYC

time_interval = ('2024-01-01', '2024-01-10')

evalscript = """
// NDVI calculation
function setup() {
    return {
        input: ["B04", "B08"],
        output: { bands: 1 }
    };
}

function evaluatePixel(sample) {
    let ndvi = (sample.B08 - sample.B04) / (sample.B08 + sample.B04);
    return [ndvi];
}
"""
   request = SentinelHubRequest(
       evalscript=evalscript,
       input_data=[
           SentinelHubRequest.input_data(
               data_collection=DataCollection.SENTINEL2_L2A,
               time_interval=time_interval
           )
       ],
       responses=[
           SentinelHubRequest.output_response('default', MimeType.TIFF)
       ],
       bbox=area_of_interest,
       size=(512, 512),
       config=config
   )

   # Get data (list of arrays)
   response = request.get_data()

   # Assign the first (and likely only) array to ndvi_data
   ndvi_data = response[0]

   # Convert to float and clip
   ndvi_data = ndvi_data.astype(np.float32)
   ndvi_data = np.clip(ndvi_data, -1, 1)
```

```
# If the array has extra dimensions (e.g. shape [512, 512, 1]), squeeze them
if ndvi_data.ndim == 3 and ndvi_data.shape[-1] == 1:
    ndvi_data = ndvi_data.squeeze(-1)

# Plot NDVI
plt.figure(figsize=(10, 8))
plt.title("NDVI Analysis")
plt.imshow(ndvi_data, cmap='YlGn')
plt.colorbar(label="NDVI Value")
plt.show()
```

Output:

KSMSCIT010 – Himanshu Singh



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 6

EDA on Ecom Reviews

Aim: EDA on Ecom Reviews.

Code:

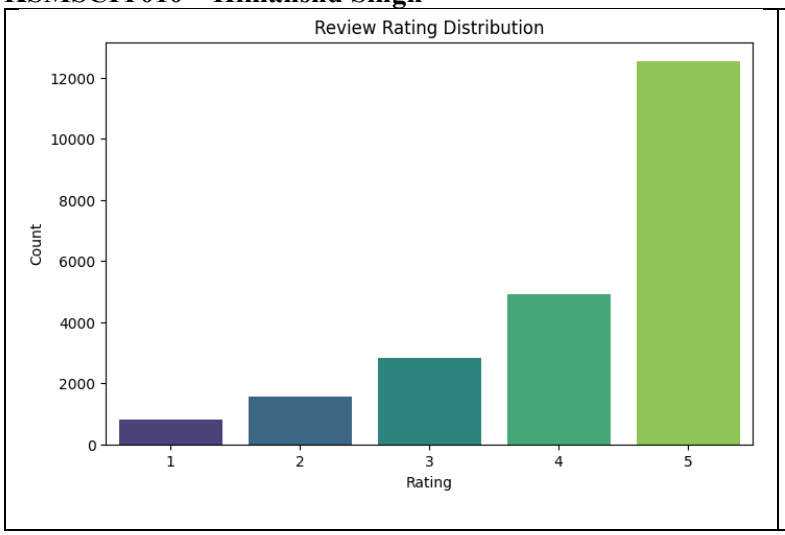
```
#Importing the required packages

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re


# Load dataset
file_path = "/content/Womens Clothing E-Commerce Reviews.csv"
df = pd.read_csv(file_path)

# Drop rows with missing reviews
df = df.dropna(subset=['Review Text'])

# Function to clean text
def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra spaces
    return text
```

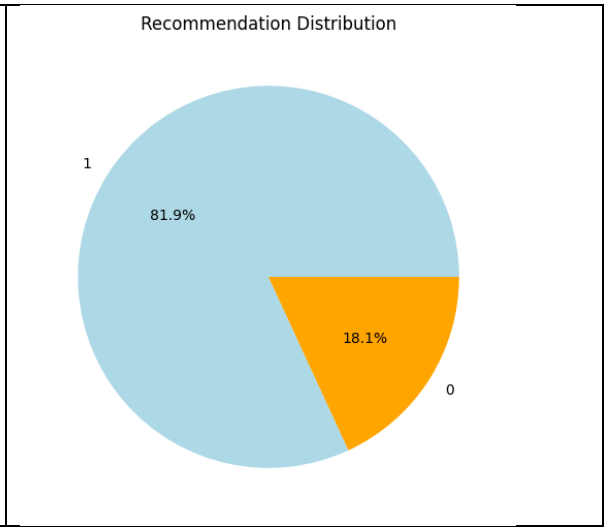
```
# Apply text cleaning
df['Cleaned Review'] = df['Review Text'].apply(clean_text)

# Plot rating distribution
plt.figure(figsize=(8,5))
sns.countplot(x=df['Rating'], palette='viridis')
plt.title('Review Rating Distribution')
plt.xlabel('Rating')
plt.ylabel('Count')

plt.show()
# Pie chart for recommended vs. not recommended
plt.figure(figsize=(6,6))
df['Recommended IND'].value_counts().plot.pie(autopct='%1.1f%%', colors=['lightblue', 'orange'])
plt.title('Recommendation Distribution')
plt.ylabel('')
plt.show()

# Boxplot for age distribution by rating
plt.figure(figsize=(10,6))
sns.boxplot(x='Rating', y='Age', data=df, palette='coolwarm')
plt.title('Age Distribution by Rating')
plt.xlabel('Rating')
plt.ylabel('Age')
plt.show()
```

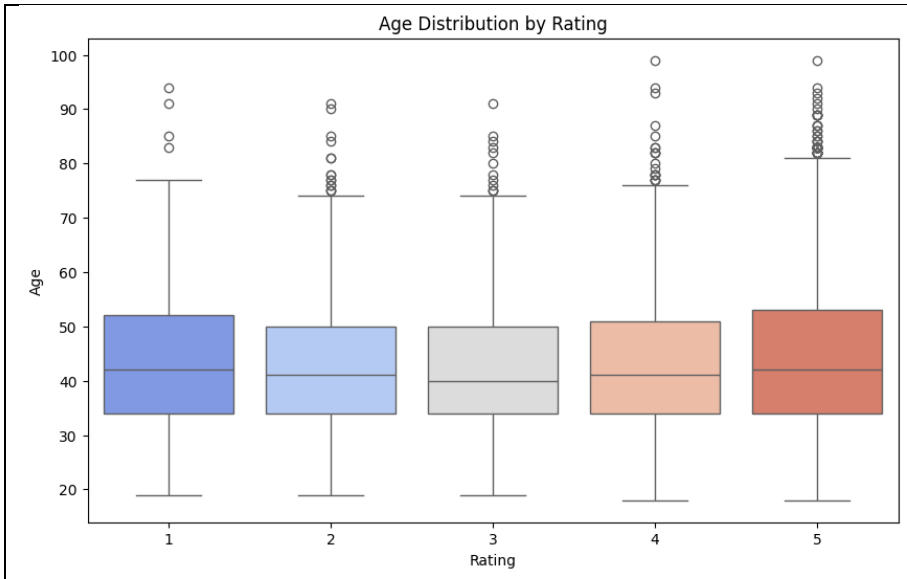
```
# Bar chart for top 10 most reviewed clothing items
plt.figure(figsize=(12,6))
top_products = df['Clothing ID'].value_counts().nlargest(10)
top_products.plot(kind='bar', color='teal')
plt.title('Top 10 Most Reviewed Clothing Items')
plt.xlabel('Clothing ID')
plt.ylabel('Number of Reviews')
plt.show()
```

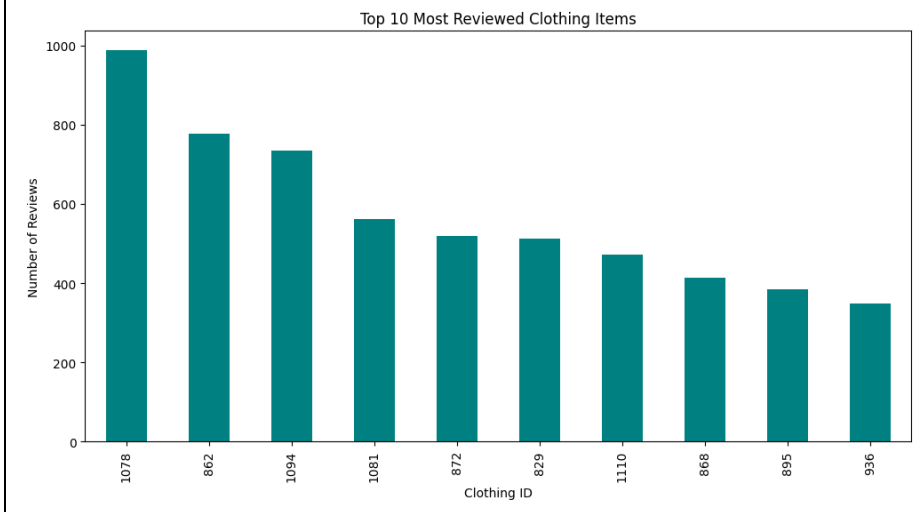
Output:

KSMSCIT010 – Himanshu Singh









KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 7

Sentiment Analysis on IMDb Dataset

Aim: Sentiment Analysis on IMDb Dataset.

Code:

```
# Importing the required packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import nltk
from nltk.corpus import stopwords
import re
from keras.datasets import imdb

# Download NLTK resources
nltk.download('stopwords')
nltk.download('punkt')

stop_words = set(stopwords.words('english'))

def load_and_clean_data(num_words=10000):
    """
    Loads the IMDb dataset from Keras, decodes integer sequences back to text,
    then cleans up duplicates and missing values.
    Returns a pandas DataFrame with columns ['review', 'sentiment'].
    """
  # 1. Load IMDb data (train/test splits)
  (train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=num_words)

  # 2. Build a reverse dictionary to decode integer sequences back to words
  word_index = imdb.get_word_index()
  reverse_word_index = {value: key for (key, value) in word_index.items()}

  # Note: The first few indices are reserved (0, 1, 2, 3), so we shift by 3 when decoding.
  def decode_review(encoded_review):
      return " ".join([reverse_word_index.get(i - 3, "?") for i in encoded_review if i >= 3])

  # 3. Decode reviews
  train_reviews = [decode_review(seq) for seq in train_data]
  test_reviews = [decode_review(seq) for seq in test_data]

  # 4. Create DataFrames for train and test
  df_train = pd.DataFrame({
      "review": train_reviews,
      "sentiment": train_labels
  })
  df_test = pd.DataFrame({
      "review": test_reviews,
      "sentiment": test_labels
  })
   # Combine train and test sets
   df = pd.concat([df_train, df_test], ignore_index=True)

   # 5. Convert sentiment from 0/1 to string labels (optional)
   df["sentiment"] = df["sentiment"].map({0: "negative", 1: "positive"})

   # 6. Drop duplicates
   df.drop_duplicates(inplace=True)

   # 7. Drop missing values (if any)
   df.dropna(inplace=True)

   # 8. Ensure 'review' and 'sentiment' columns exist
   if "review" not in df.columns or "sentiment" not in df.columns:
       raise ValueError("Dataset does not have required columns: 'review' and 'sentiment'")

   return df
```

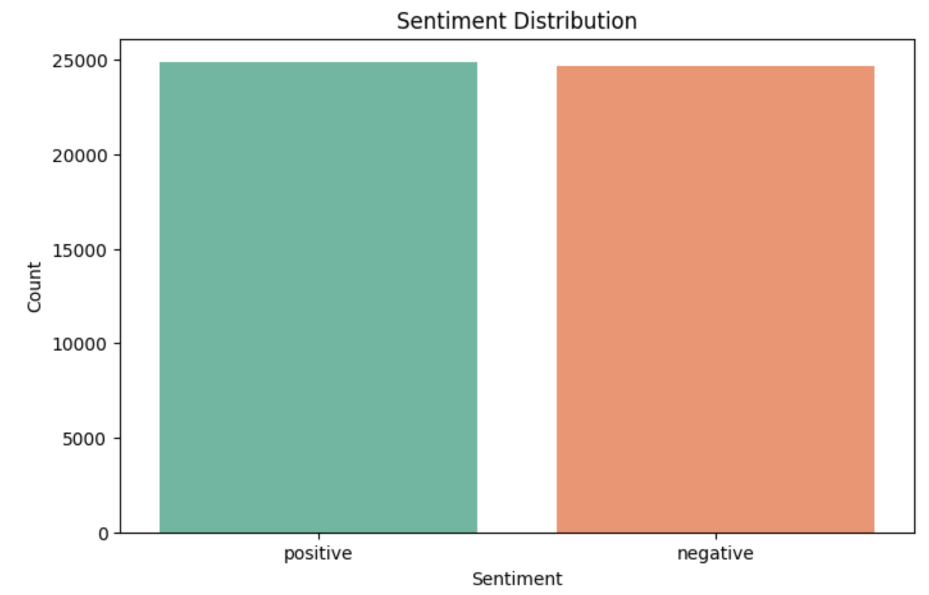
```
# ---------------------------------------------------------------------
# Main script
# ---------------------------------------------------------------------
df = load_and_clean_data(num_words=10000)

# Check the first few rows
print(df.head())

# Check for missing values
print("\nMissing Values:\n", df.isnull().sum())

# (Optional) Check distribution of sentiments
print("\nSentiment Distribution:\n", df["sentiment"].value_counts())

def clean_text(text):
    text = re.sub(r'<.*?>', '', text)  # Remove HTML tags
    text = re.sub(r'[^a-zA-Z ]', '', text)  # Remove non-alphabetic characters
    text = text.lower()
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return ' '.join(words)

# Apply cleaning
df['clean_review'] = df['review'].astype(str).apply(clean_text)
# Visualization

# 1. Sentiment Distribution Bar Plot
plt.figure(figsize=(8, 5))
sns.countplot(x='sentiment', data=df, palette='Set2')
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()

# 2. Word Cloud for Positive Reviews
positive_reviews = ' '.join(df[df['sentiment'] == 'positive']['clean_review'])
wordcloud_positive = WordCloud(width=800, height=400, background_color='white').generate(positive_reviews)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_positive, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Positive Reviews')
plt.show()
```

```
# 3. Word Cloud for Negative Reviews
negative_reviews = ' '.join(df[df['sentiment'] == 'negative']['clean_review'])
wordcloud_negative = WordCloud(width=800, height=400, background_color='black', colormap='Reds').generate(negative_reviews)

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_negative, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Negative Reviews')
plt.show()
```

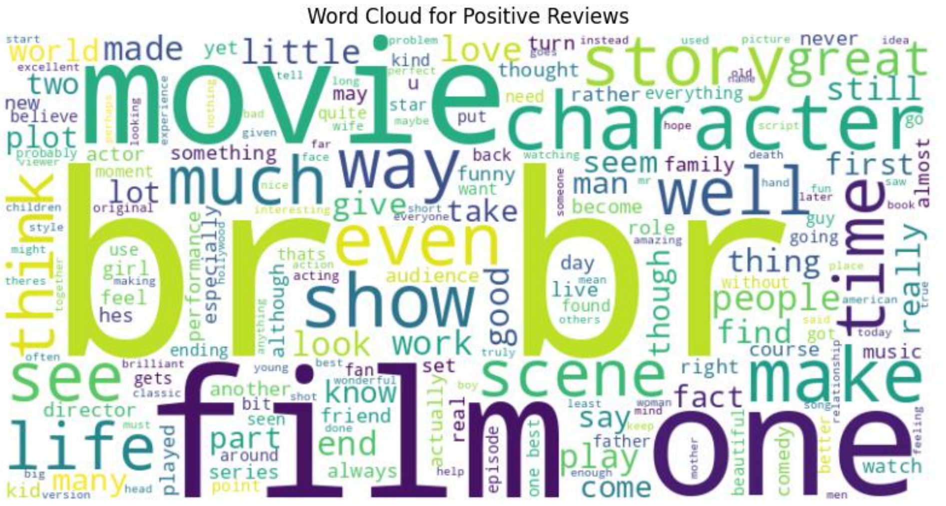
Output:

KSMSCIT010 – Himanshu Singh

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
                                              review sentiment
0  this film was just brilliant casting location ...  positive
1  big hair big boobs bad music and a giant safet...  negative
2  this has to be one of the worst films of the 1...  negative
3  the at storytelling the traditional sort many ...  positive
4  worst mistake of my life br br i picked this m...  negative

Missing Values:
 review       0
sentiment    0
dtype: int64

Sentiment Distribution:
 sentiment
positive    24881
negative    24697
Name: count, dtype: int64
<ipython-input-28-c5b0a518d2ab>:97: FutureWarning:
```



Sentiment Distribution
25000
20000
15000
10000
5000
0
Count
positive
negative
Sentiment



Word Cloud for Positive Reviews
br
movie
film
one
story
character
great
way
well
even
show
scene
make
time
much
see
life
think

Word Cloud for Negative Reviews
br
movie
film
one
character
even
look
well
show
scene
story
make
way
seem
good
bad

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 8

Python/R Program for Selecting Billboard Content from Given Data

Aim: Python/R Program for Selecting Billboard Content from Given Data

Code:

```python
#Pyhton
# Importing the required packages

import pandas as pd

data = {
    'content_id': [1, 2, 3, 4, 5, 6],
    'title': ['Tiger 3', 'Arijit Singh Live in Concert', 'Super Bowl Ad', 'Bajrangi Bhaijaan', 'The Voice Finale', 'Coca-Cola Ad'],
    'category': ['Entertainment', 'Music', 'Advertisement', 'Entertainment', 'TV Show', 'Advertisement'],
    'views': [95000, 80000, 70000, 85000, 45000, 90000]
}

df = pd.DataFrame(data)

billboard_data = df[(df['category'] == 'Entertainment') & (df['views'] > 50000)]

# Display the filtered content
print("Billboard Content:")
print(billboard_data)
```

Output:


```
Billboard Content:
   content_id              title       category  views
0           1            Tiger 3  Entertainment  95000
3           4  Bajrangi Bhaijaan  Entertainment  85000
```

Code:

```r
# Importing the required packages

#R
pick_bill_song <- function(songs,num_songs){

  shuffle_songs <-sample(songs)
  bill_songs <-head(shuffle_songs,num_songs)
  return(bill_songs)
}
all_songs <-c("song 1","song 2","song 3","song 4","song5","song6","song 7","song8","song 9","song 10"
)
num_bill_song <-4

bill_songs <- pick_bill_song(all_songs,num_bill_song)
cat("BILLBOARD SONGS ARE :\n")
for (song in bill_songs){
  cat(song,"\n")
}
```

Output:

```
BILLBOARD SONGS ARE :
song 7
song 10
song 1
song8
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 9

Data Visualization using PYGAL

Aim: Data Visualization using PYGAL.

Code:

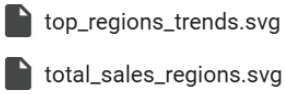
```
!pip install pygal
```

```
# Importing the required packages

import pygal
import random

regions = [f'Region {i}' for i in range(1, 11)]
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
          'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

# Generate random sales data for 10 regions over 12 months
sales_data = {region: [random.randint(1000, 10000) for _ in months] for region in regions}

bar_chart = pygal.Bar(title='Total Sales Across 10 Regions (Yearly)')
for region, sales in sales_data.items():
    bar_chart.add(region, sum(sales))  # Summing up monthly sales per region

# Save as SVG
bar_chart.render_to_file('total_sales_regions.svg')

top_regions = sorted(sales_data.items(), key=lambda x: sum(x[1]), reverse=True)[:3]

line_chart = pygal.Line(title='Monthly Sales Trends for Top 3 Regions')
line_chart.x_labels = months

# Add monthly sales data for top 3 regions
for region, sales in top_regions:
    line_chart.add(region, sales)

line_chart.render_to_file('top_regions_trends.svg')
```

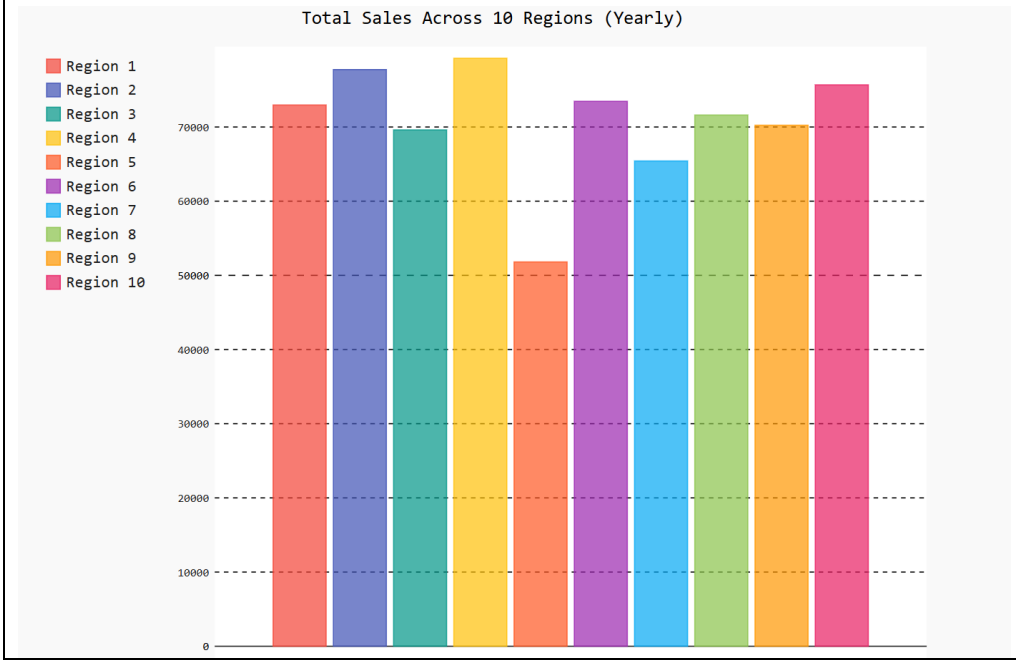
Output:

KSMSCIT010 – Himanshu Singh

This 2 Files will be downloaded!




Monthly Sales Trends for Top 3 Regions
Region 4
Region 2
Region 10
9000
8000
7000
6000
5000
4000
3000
2000
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Total Sales Across 10 Regions (Yearly)
Region 1
Region 2
Region 3
Region 4
Region 5
Region 6
Region 7
Region 8
Region 9
Region 10
70000
60000
50000
40000
30000
20000
10000
0

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 10

Processing Balance Sheet Data to Ensure Quality Filtering

Aim: Processing Balance Sheet Data to Ensure Quality Filtering.

Code:

```
# Importing the required packages

balance_data = pd.DataFrame({
    'Date': ['2024-01-01', '2024-01-05', '2024-01-10', '2024-01-12',
             '2024-01-15', '2024-01-20', '2024-01-25'],
    'Account': ['Sales', 'Purchase', 'Sales', 'Expense', 'Sales', 'Purchase', 'Salary'],
    'Amount': [15000, -12000, 20000, -5000, 0, -8000, 10000],
    'Status': ['Valid', 'Valid', 'Invalid', 'Valid', 'Invalid', 'Valid', 'Valid']
})

valid_data = balance_data[(balance_data['Status'] == 'Valid') & (balance_data['Amount'] != 0)]


valid_data.to_csv('cleaned_balance_sheet.csv', index=False)

print("Valid balance sheet data processed and saved to 'cleaned_balance_sheet.csv'.")
```

Output:

KSMSCIT010 – Himanshu Singh

```
Valid balance sheet data processed and saved to 'cleaned_balance_sheet.csv'.
```

cleaned_balance_sheet.csv

cleaned_balance_sheet.csv ×

1 to 5 of 5 entries Filter

Date	Account	Amount	Status
2024-01-01	Sales	15000	Valid
2024-01-05	Purchase	-12000	Valid
2024-01-12	Expense	-5000	Valid
2024-01-20	Purchase	-8000	Valid
2024-01-25	Salary	10000	Valid

Show 10 per page

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Practical 11

Working with MongoDB

MongoDB is an open-source, non-relational database management system (DBMS) that stores and processes data as documents instead of tables and rows. MongoDB is known for its flexibility and scalability, and is used by over **47,000** customers across 118 regions.

Features

Document model

MongoDB's document model is designed to be simple for developers to learn and use. Documents are formatted as Binary JSON (BSON) and can store various types of data.

Scalability

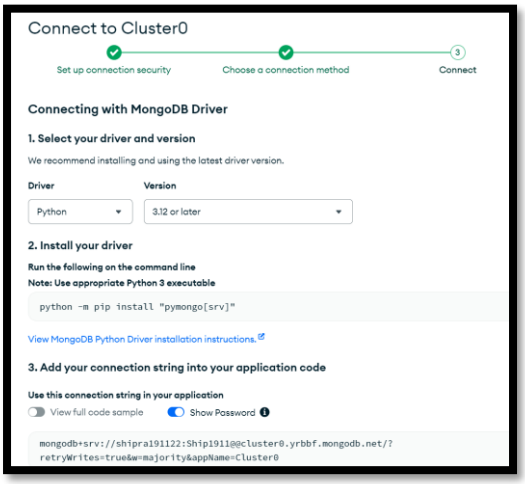
MongoDB is a distributed database that's built for high availability, horizontal scaling, and geographic distribution.

Query API

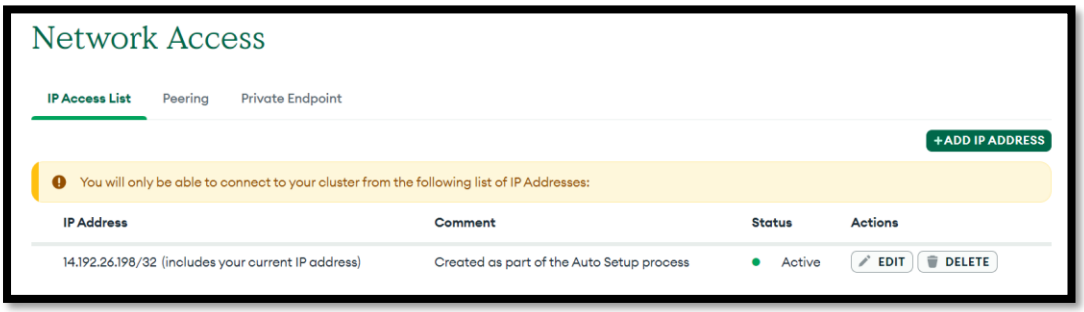
MongoDB offers a developer-native query API for working with data.

Aim: A) Python

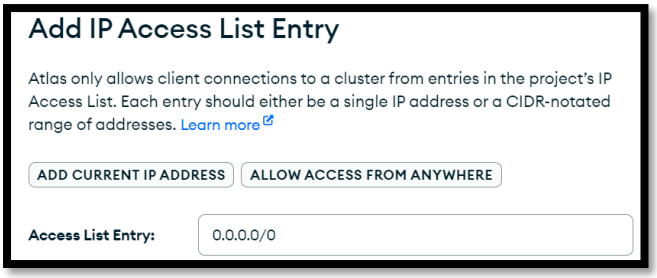
Step 1 – Signup or Login MongoDB (Email).
Step 2 – In Database go to Data Services create a "CLUSTER".
Step 3 – Deploy a cluster by choosing "M0 FREE" Name – Cluster0.
Step 4 – Connect to Cluster → Username – "Shipra191122" | Password – "sXdAQft7csi7mfy".



- python -m pip install "pymongo[srv]"
- mongodb+srv://shipra191122:<password>@cluster0.yrbbf.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0



- Here it using the local IP Address that we have to delete.
- We have to Add Ip Address because “Google Colab” is Cloud.



- **Allow Access from Anywhere and Confirm.**
- (It will allow to connect with cloud platform like “Google Colab”).

CODE:

```
from pymongo import MongoClient

client =
MongoClient('mongodb+srv://shipra191122:sXdAQft7csi7mfy@cluster0.yrbbf.mongodb.net/?retryWrites=true&w=majority')

db = client['mydatabase']
collection = db['pymongo2']

document = [
    {"_Id": "Moin1911","name": "Moin", "city": "Delhi"},
    {"_Id": "Sam123","name": "Samira", "city": "Kolkata"},
    {"_Id":"Iqra789","name": "Iqra", "city": "Ahemdabad"},
]

insert_doc = collection.insert_many(document)

for doc_id in insert_doc.inserted_ids:
  print(f"Inserted Document IDs: {doc_id}")

client.close()
```

OUTPUT:
Cluster → Browse Collection

```
Inserted Document IDs: 67015b9f899a921bffd697b1
Inserted Document IDs: 67015b9f899a921bffd697b2
Inserted Document IDs: 67015b9f899a921bffd697b3
```

QUERY RESULTS: 1-3 OF 3

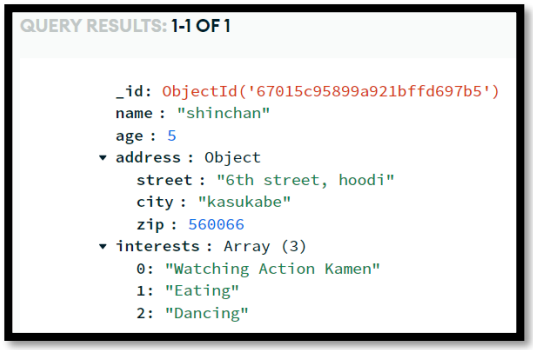
```
_id: ObjectId('67015b9f899a921bffd697b1')
_Id : "Moin1911"
name : "Moin"
city : "Delhi"

_id: ObjectId('67015b9f899a921bffd697b2')
_Id : "Sam123"
name : "Samira"
city : "Kolkata"

_id: ObjectId('67015b9f899a921bffd697b3')
_Id : "Iqra789"
name : "Iqra"
city : "Ahemdabad"
```

CODE:

```
from pymongo import MongoClient
# from pymongo.errors import DuplicateKeyError

client =
MongoClient('mongodb+srv://shipra191122:sXdAQft7csi7mfy@cluster0.yrbbf.mongodb.net/?retryWrites=true&w=majority')

db = client['mydatabase']
collection = db['pymongo3']

#inserting the embaded document..

document_with_embaded = {
   "name" : "shinchan",
   "age" : 5,
   "address" : {
      "street" : "6th street, hoodi",
      "city" : "kasukabe",
      "zip" : 560066
   },
   "interests" : ["Watching Action Kamen", "Eating", "Dancing"]
}

insterted = collection.insert_one(document_with_embaded)

print(f"inserted document ID: {insterted.inserted_id}")
```

OUTPUT:

```
inserted document ID: 67015c95899a921bffd697b5
```



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

M.Sc. (I.T.) Part-2 Semester IV

Working with MongoDB

Aim: B) R

CODE:

Step 1 - Set Up MongoDB in R

```
install.packages("mongolite")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

Step 2 - Connect to MongoDB

```
library(mongolite)

# Replace with your MongoDB URI
mongo_uri <-
"mongodb+srv://shipra191122:sXdAQft7csi7mfy@cluster0.yrbbf.mongodb.net/?retryWrites=true&w=
majority"

# Connect to the collection
mongo_conn <- mongo(collection = "mongoDB_in_R", url = mongo_uri)

# Print the first few records
print(mongo_conn$find(limit = 5))
```

```
data frame with 0 columns and 0 rows
```

Step 3 - Insert Data into MongoDB

```
data <- data.frame(name = c("Shipra", "Moin", "Samira"),
                   age = c(22, 21, 22))

# Insert the data
mongo_conn$insert(data)
```

```
List of 5
 $ nInserted  : num 3
 $ nMatched   : num 0
 $ nRemoved   : num 0
 $ nUpserted  : num 0
 $ writeErrors: list()
```

Step 4 - Query Data from MongoDB

```
# Query all records
results <- mongo_conn$find()

# View the queried data
print(results)
```

```
    name age
1 Shipra  22
2   Moin  21
3 Samira  22
```

Step 5 - Close the Connection

```
mongo_conn$disconnect()
```