PRACTICAL JOURNAL

in

APPLIED BIG DATA ANALYTICS

Submitted by

KSMSCIT033 ASHISH ANIL SHENDE

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. **ASHISH ANIL SHENDE** for M.Sc. (IT) Part- II Semester- IV, Seat No: **KSMSCIT033** 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 III

INDEX

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

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

PRACTICAL 1

Aim: Recommendation System

Code:

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

Output:

	title
0	Reservoir Dogs (1992)
1	Truman Show, The (1998)
2	Matrix, The (1999)
3	Trainspotting (1996)
4	Godfather, The (1972)
5	The Butterfly Effect (2004)
6	Clockwork Orange, A (1971)
7	Godfather: Part II, The (1974)
8	Shining, The (1980)
9	Lord of the Rings: The Return of the King, The...

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

PRACTICAL 2

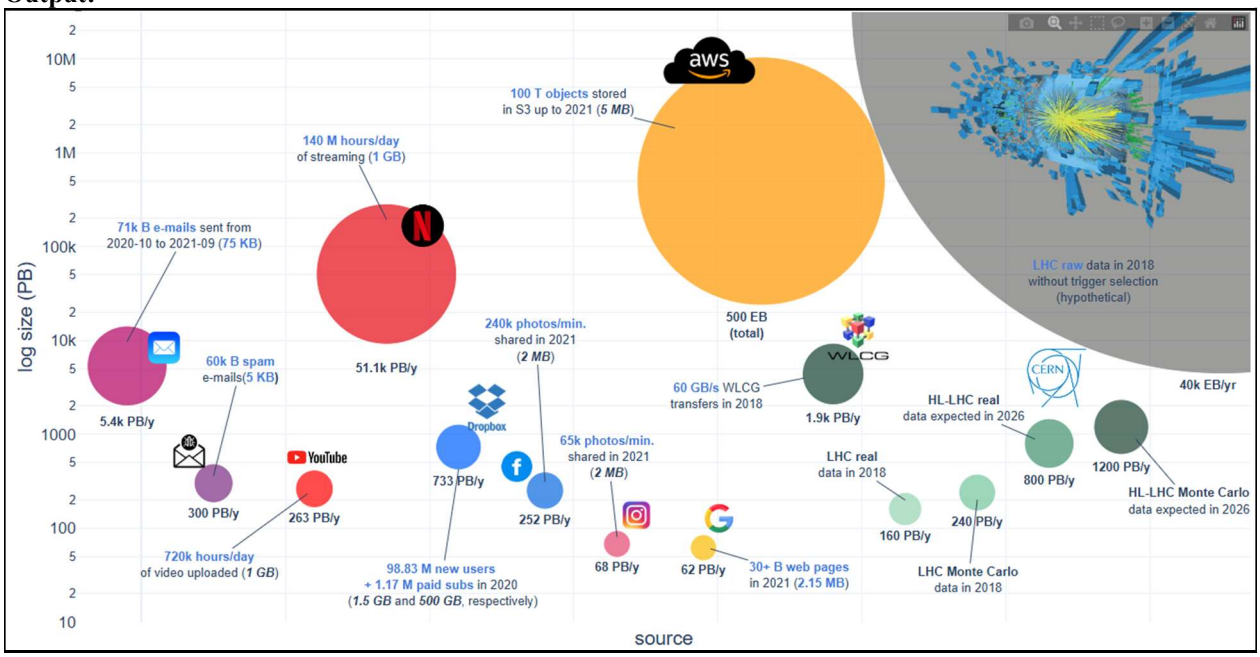
Aim: Processing Data from Social Media Platforms (Raw Data Fetching)

Code:

```
import os
import sys
from pathlib import Path
import pandas as pd
import numpy as np

pd.options.display.max_columns = 50
pd.options.display.max_rows = 30
pd.options.display.float_format = '{:,.4f}'.format
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   })
sort_idx = df.sort_values('size_PB').index
df.loc[sort_idx]
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),}
fig = go.Figure(data=[    go.Scatter( x=df.x, y=df.size_PB,   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),  )])
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])))

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"    )  )
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),  )

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:



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

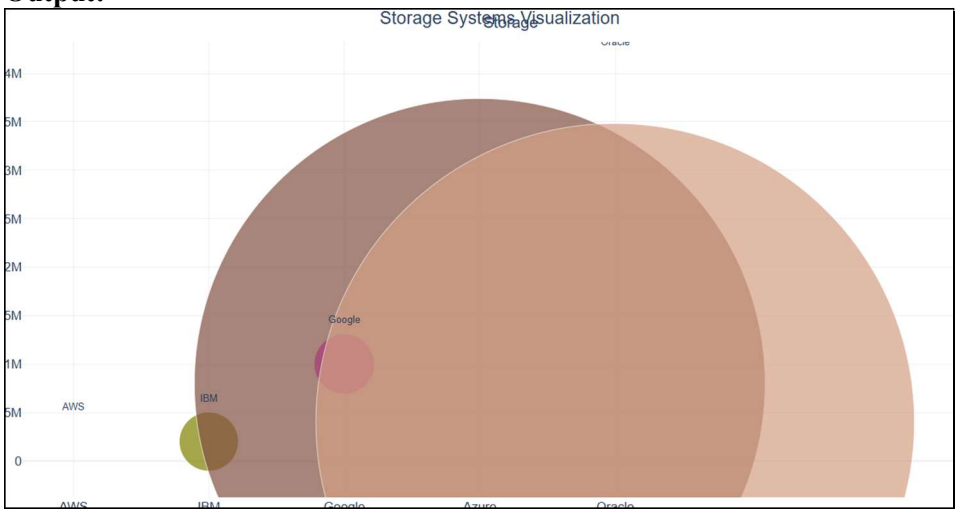
PRACTICAL 3

Aim: Collecting and Ingesting Data into Big Data Storage using Data Access Connectors

Code:

```
import pandas as pd
import plotly.graph_objects as go
file_path = '/content/my_experiment.csv'
data = pd.read_csv(file_path)
data['area_size'] = pd.to_numeric(data['area_size'], errors='coerce')
data['area_size'] = data['area_size'].fillna(0)  # Replace NaN with 0 or use .dropna()

sizeref = 2. * data['area_size'].max() / (840 ** 2)

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': [ {  'source': '/content/google_icon.png',  # Replace with the URL or path to the logo
         'xref': 'paper',  'yref': 'paper',  'x': 0.05,  'y': 0.95,   'sizex': 0.1, 'sizey': 0.1,   'opacity': 0.7,
         'layer': 'above'   }, { 'source': '/content/IBM.png', 'xref': 'paper', 'yref': 'paper', 'x': 0.9, 'y': 0.05,
         'sizex': 0.1, 'sizey': 0.1,  'opacity': 0.7,   'layer': 'above' } ] }

fig = go.Figure(data=[go.Scatter(x=data['source'], y=data['size_PB'],  mode='markers+text',
      marker=dict( size=data['area_size'],  color=data['color'],  opacity=0.7,  sizemode='area',
         sizeref=sizeref ),  text=data['source'],  textposition='top center', textfont=dict(size=14), )])
fig.update_layout(yaxis=dict( type="linear", title="Size", visible=True, ),  xaxis=dict( title="Storage",
      visible=True, ))
fig.update_layout(layout)
fig.show()
```

Output:



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

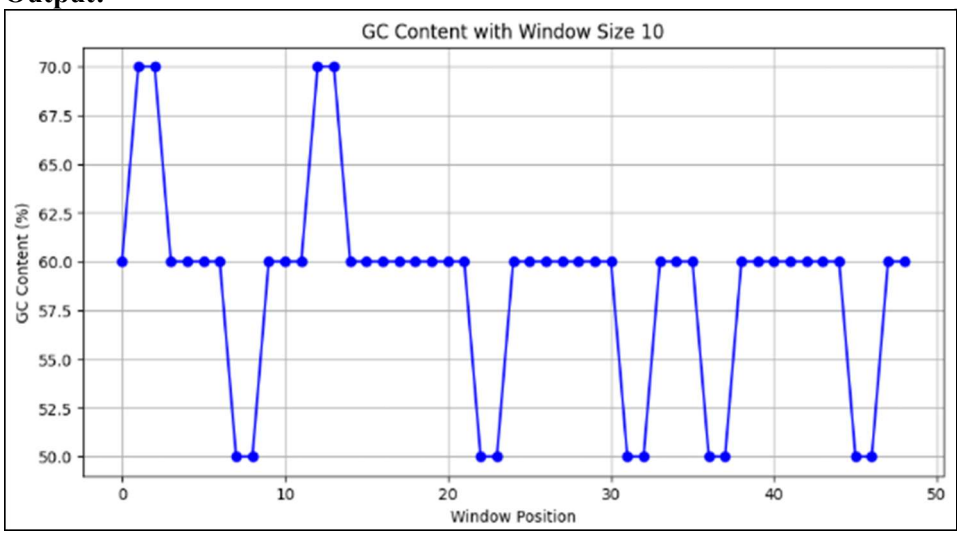
PRACTICAL 4

Aim: Genome Analysis: Calculating GC Content and Its Significance

Code:

```
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
dna_sequence = "ATGCGCGTAGCTAGGCTACGCGTACGTAGCGTAGCGTAGCT"
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:



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

PRACTICAL 5

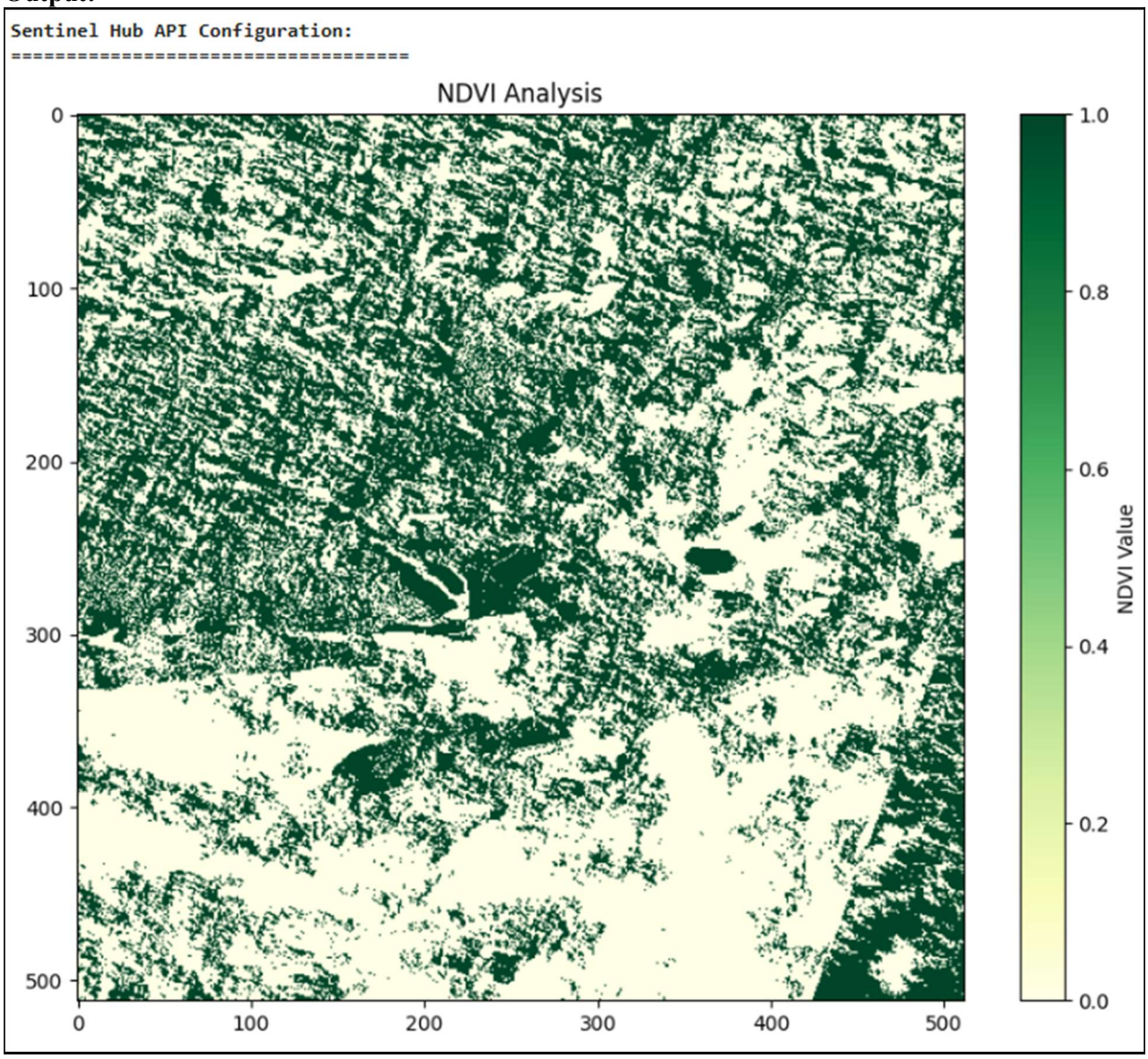
Aim: MVDI and Sentinel Hub: Remote Sensing Data Processing

Code:

```
from sentinelhub import SHConfig, BBox, CRS, DataCollection, SentinelHubRequest, MimeType
import numpy as np
import matplotlib.pyplot as plt

config = SHConfig()
config.instance_id = 'fb89a462-8e6b-4d31-8c2b-87f6fe40471b'
config.sh_client_id = '4120319c-6ecc-4a34-8903-ad3a76be7ba3'
config.sh_client_secret = 'S3fZ0K23xmxTZhNDc5Zpwe70m8Wz1gcl'
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 = """
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:



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

PRACTICAL 6

Aim: Exploratory Data Analysis (EDA) on E-Commerce Reviews

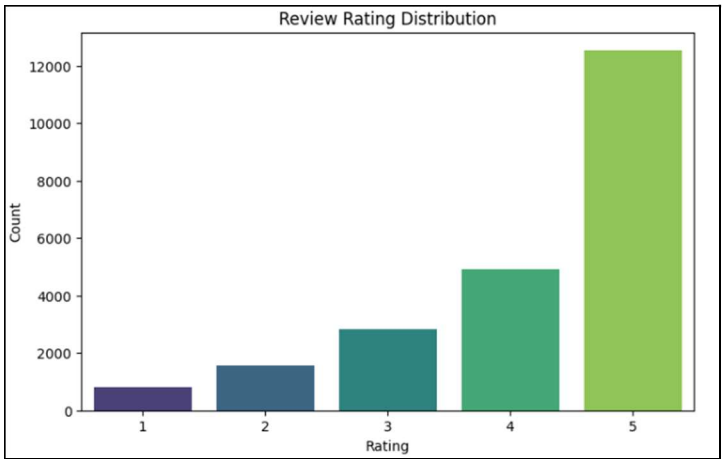
Code:

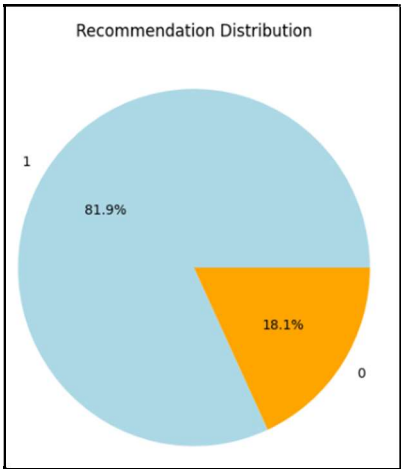
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
df = pd.read_csv("/content/Womens Clothing E-Commerce Reviews.csv")
df = df.dropna(subset=['Review Text'])
def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra spaces
    return text
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()
```

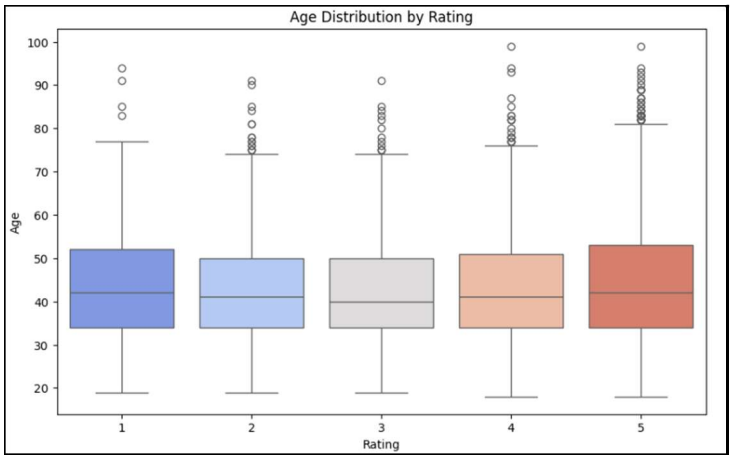
KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

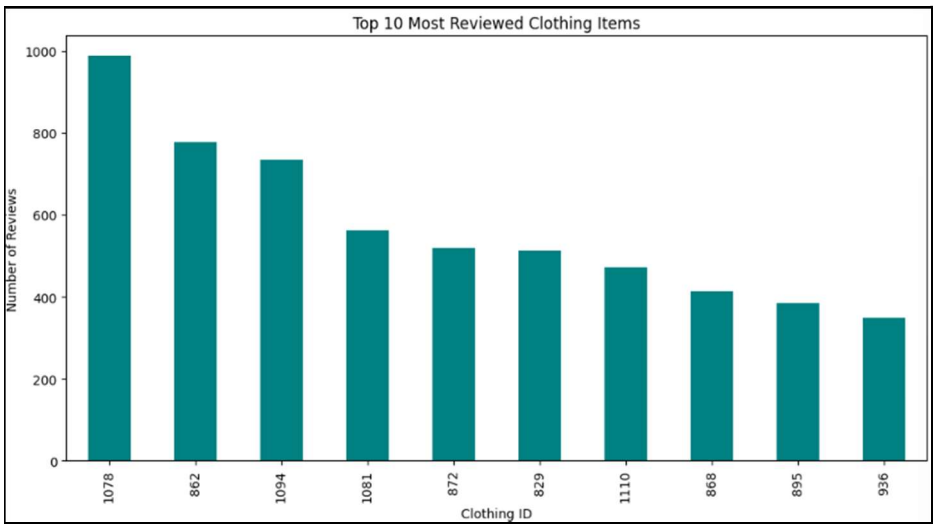
M.Sc (I.T.) Part-2 Semester IV

Output:









KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

PRACTICAL 7

Aim: Sentiment Analysis on IMDb Dataset

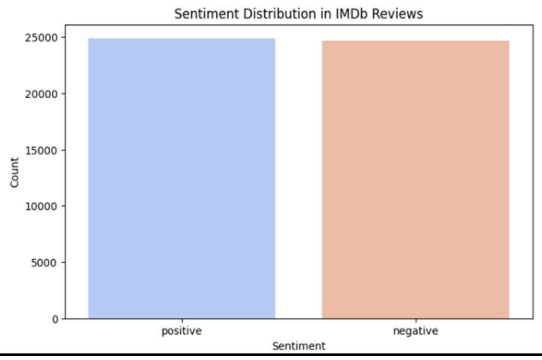
Code:

```
!pip install wordcloud seaborn nltk > /dev/null
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
from collections import Counter
import re
from keras.datasets import imdb

nltk.download('stopwords')
nltk.download('punkt')
stop_words = set(stopwords.words('english'))
def load_and_clean_data(num_words=10000):
    (train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=num_words)
    word_index = imdb.get_word_index()
    reverse_word_index = {value: key for (key, value) in word_index.items()}
    def decode_review(encoded_review):
        return " ".join([reverse_word_index.get(i - 3, "?") for i in encoded_review if i >= 3])
    train_reviews = [decode_review(seq) for seq in train_data]
    test_reviews = [decode_review(seq) for seq in test_data]

    df_train = pd.DataFrame({  "review": train_reviews,    "sentiment": train_labels    })
    df_test = pd.DataFrame({    "review": test_reviews,      "sentiment": test_labels    })

    df = pd.concat([df_train, df_test], ignore_index=True)
    df["sentiment"] = df["sentiment"].map({0: "negative", 1: "positive"})
    df.drop_duplicates(inplace=True)
    df.dropna(inplace=True)
    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

df = load_and_clean_data(num_words=10000)
print(df.head())
print("\nMissing Values:\n", df.isnull().sum())
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)
df['clean_review'] = df['review'].astype(str).apply(clean_text)
```

```
plt.figure(figsize=(8, 5))
        sns.barplot(x=df['sentiment'].value_counts().index, y=df['sentiment'].value_counts().values, palette='coolwarm')
plt.title("Sentiment Distribution in IMDb Reviews")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.show()

# Word Frequency in Positive vs Negative Reviews
from collections import Counter
positive_words = ' '.join(df[df['sentiment'] == 'positive']['clean_review']).split()
negative_words = ' '.join(df[df['sentiment'] == 'negative']['clean_review']).split()
positive_word_counts = Counter(positive_words).most_common(20)
negative_word_counts = Counter(negative_words).most_common(20)
positive_df = pd.DataFrame(positive_word_counts, columns=['Word', 'Count'])
negative_df = pd.DataFrame(negative_word_counts, columns=['Word', 'Count'])

# Plot most common words in positive reviews
plt.figure(figsize=(8, 5))
sns.barplot(x='Count', y='Word', data=positive_df, palette='coolwarm')
plt.title("Top 20 Words in Positive Reviews")
plt.show()

# Plot most common words in negative reviews
plt.figure(figsize=(8, 5))
sns.barplot(x='Count', y='Word', data=negative_df, palette='coolwarm')
plt.title("Top 20 Words negative in reviews")
plt.show()
```

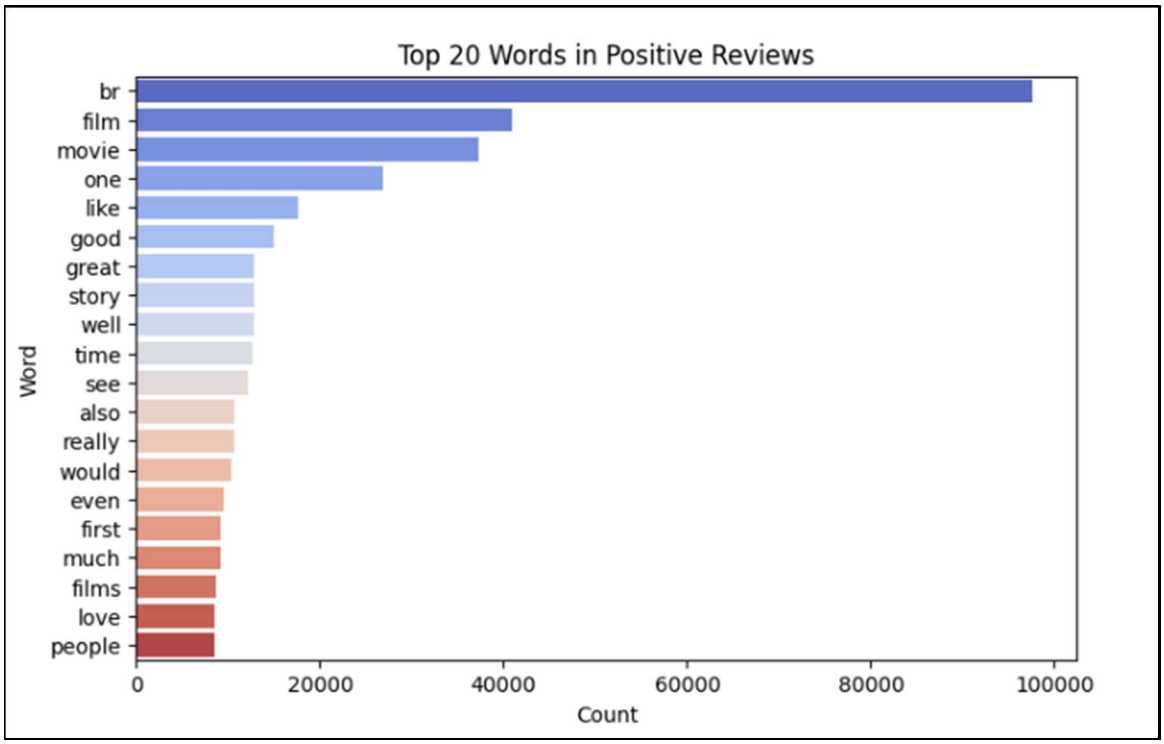
Output:

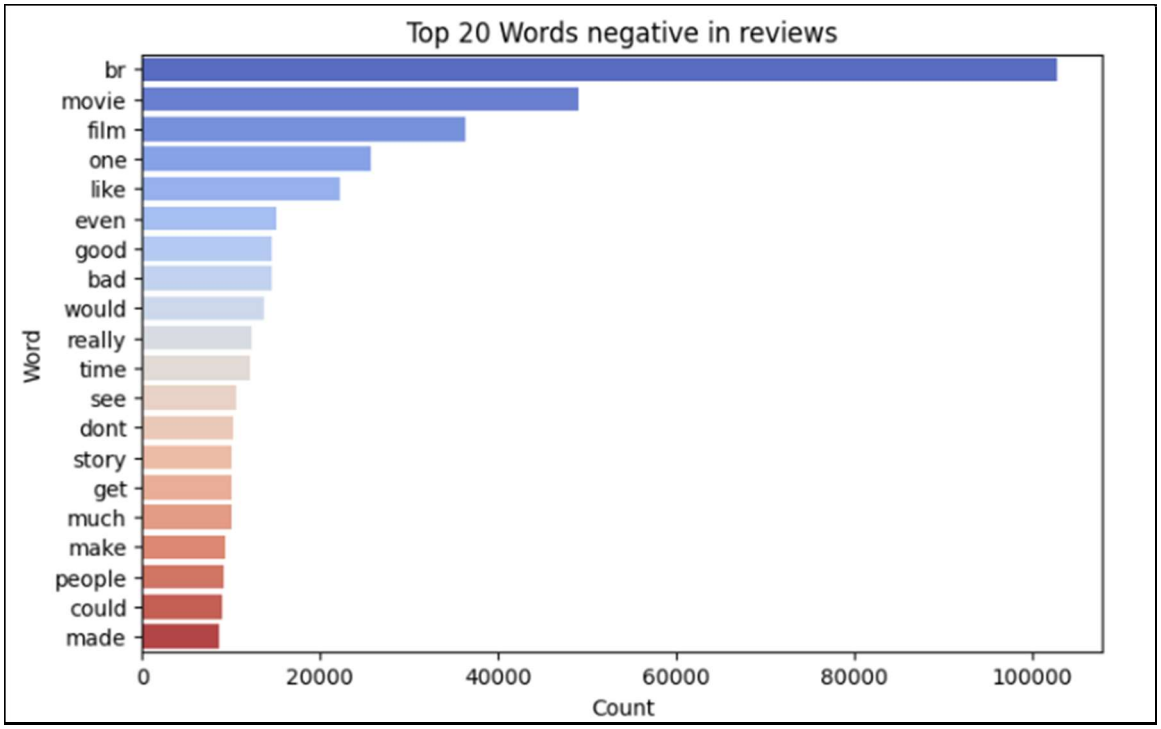
```
                                              review sentiment
0  this film was just brilliant casting location ...  positive
1  big hair big boobs bad music and a giant safet...  negative
2  this has to be one of the worst films of the 1...  negative
3  the at storytelling the traditional sort many ...  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
```



KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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





KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

PRACTICAL 8

Aim: Write a Python / R program to pick the content for Bill Boards from the given data

Code:

#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 :
> for (song in bill_songs){
+   cat(song,"\n")
+ }
song 3
song6
song 1
song 9
```

#Python

```
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)]
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
```

KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

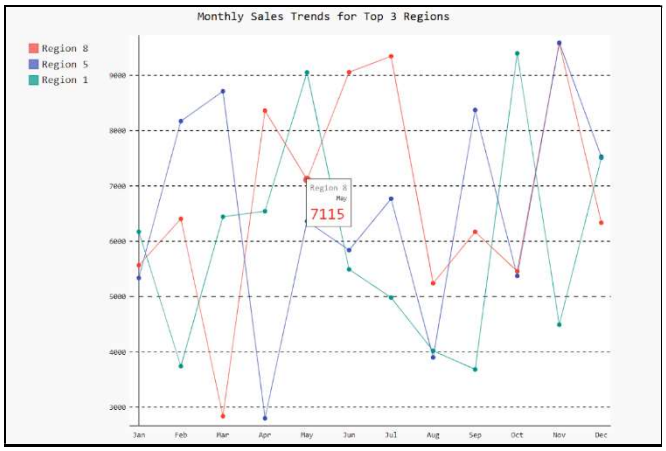
PRACTICAL 9

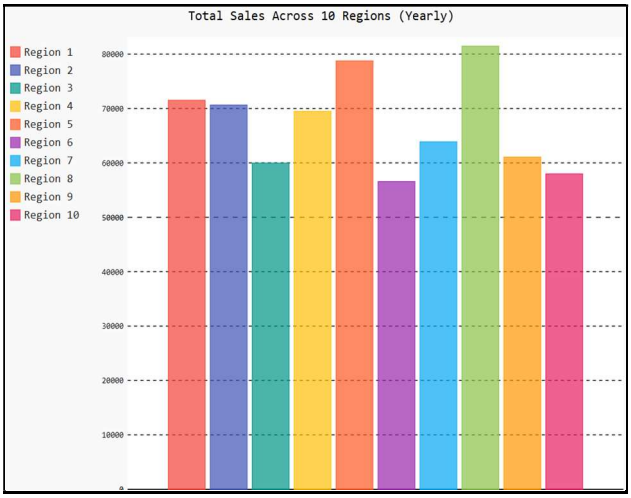
Aim: Data visualization using pygal

Code:

```
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']

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
for region, sales in top_regions:
line_chart.add(region, sales)

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

Output:





KISHINCHAND CHELLARAM COLLEGE, MUMBAI - 20

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

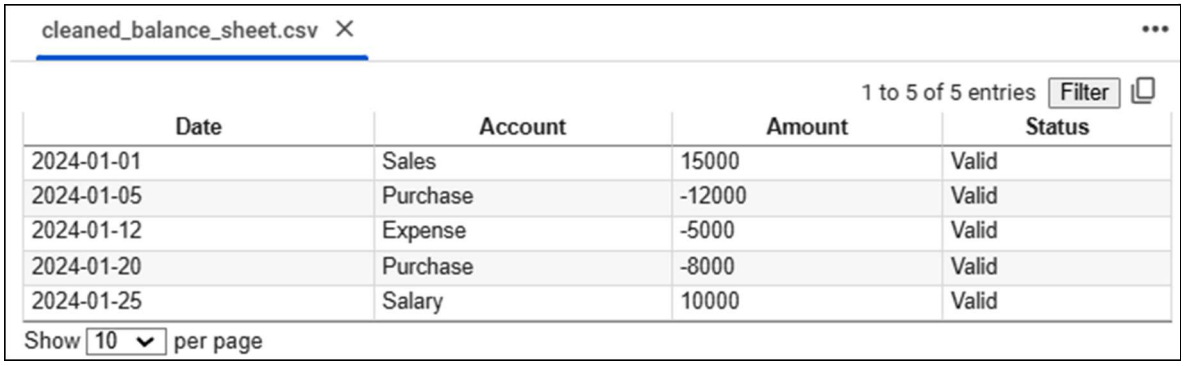
PRACTICAL 10

Aim: Write a Python program to process the balance sheet to ensure that only gooddata is processing

Code:

```
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:



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

Aim: Working with MongoDB

Code:

```
import matplotlib.pyplot as plt
from pymongo import MongoClient
class MongoDBClient:
    def __init__(self, uri, db_name):
        self.client = MongoClient(uri)
        self.db = self.client[db_name]
        self.status_message = f"Connected to {db_name}"
    def retrieve_documents(self, collection_name, limit=0):
        """Retrieve documents from the specified collection."""
        return list(self.db[collection_name].find(limit=limit))
ATLAS_URI =
"mongodb+srv://surabhisalunke02:C3g8b2m8#@cluster0.w1xhx.mongodb.net/?retryWrites=true&w=majority"
DB_NAME = 'sample_mflix'
COLLECTION_NAME = 'embedded_movies'
client = MongoDBClient(ATLAS_URI, DB_NAME)
docs = client.retrieve_documents(COLLECTION_NAME, limit=5)
total_movies = client.db[COLLECTION_NAME].count_documents({})
output_message = (
    f"{client.status_message}\n"
    f"Total Movies: {total_movies}\n"
    f"Retrieved: {len(docs)} Movies\n\n"
    + "\n\n".join(
        [f'Title: {doc.get("title", "N/A")}\n'
         f'Year: {doc.get("year", "N/A")}\n'
         f'Plot: {doc.get("plot", "N/A")}\n'
         for doc in docs] ))
plt.figure(figsize=(10, 6))
plt.text(0.1, 0.5, output_message, fontsize=12, ha='left', va='center', wrap=True)
plt.axis('off')
plt.title("Movie Information", fontsize=16)
plt.show()
```

Output: