DRONACHARYA
College of Engineering

LABORATORY MANUAL

B.Tech. Semester- VIII

LAB

Subject code: LC-CSE-421G

Prepared by:	**Checked by:**	**Approved by:**
Prof. Pooja Khot	Dr. Ashima Mehta	Name : Prof. (Dr.) Isha Malhotra
Sign.: ……	**Sign.: ……**	**Sign.: ……**

TABLE OF CONTENTS

VISION AND MISSION OF THE INSTITUTE

Vision:

"Empowering human values and advanced technical education to navigate and address global challenges with excellence."

Mission:

- **M1:** Seamlessly integrate human values with advanced technical education.
- **M2:** Supporting the cultivation of a new generation of innovators who are not only skilled but also ethically responsible.
- **M3:** Inspire global citizens who are equipped to create positive and sustainable impact, driving progress towards a more inclusive and harmonious world.

VISION AND MISSION OF THE DEPARTMENT

Vision:

"Steering the future of computer science through innovative advancements, fostering ethical values and principles through technical education."

Mission:

M1: Directing future innovations in computer science through revolutionary progress.

M2: Instilling a foundation of ethical values and principles in every technologist.

M3: Offering a comprehensive technical education to equip individuals for a meaningful and influential future.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Apply the technical competence in Computer Science and Engineering for solving problems in the real world.

PEO2: Carry out research and develop solutions on problems of social applications.

PEO3: Work in a corporate environment, demonstrating team skills, work morals, flexibility and lifelong learning.

PROGRAMME OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Exhibit design and programming skills to develop and mechanize business solutions using revolutionary technologies.

PSO2: Learn strong theoretical foundation leading to brilliance and enthusiasm towards research, to provide well-designed solutions to complicated problems.

PSO3: Work effectively with diverse Engineering fields as a team to design, build and develop system applications.

UNIVERSITY SYLLABUS

1. Write a java program to check whether given alphabet is vowel or not.
2. Write a java program to implement method overloading.
3. Write a java program to implement method riding.
4. Write a java program to solve Fibonacci series using recursive method.
5. Write a java program to create a class circle and initialize and display its variables center and radius.
6. Write a java program to implement parameterized constructor.
7. Write a java program to implement stack operations using array.
8. Write a java program to perform addition of two matrices.
9. Write a java program to implement exception handling.
10. Write a java program extend thread class.
11. Write a java program to implement ArrayList using collection framework.

COURSE OUTCOMES (COs)

Upon successful completion of the course, the students will:

1. Gain knowledge of the structure and model of the Big Data Analyticsprogramming language, (knowledge)
2. Use the Big Data Analytics programming language for various programming technologies (understanding)
3. Develop software in the Big Data Analytics programming language

CO-PO Mapping:

	PSO1	PSO2	PSO3	PSO4	PSO5	PSO6	PSO7	PSO8	PSO9	PSO10	PSO11	PSO12
C327.1	3	3	2	2	2	1	-	1	-	1	1	3
C327.2	3	3	3	2	3	1	-	1	-	1	1	3
C327.3	3	3	2	1	2	1	-	1	-	1	1	3

CO-PSO Mapping:

	PSO1	PSO2	PSO3
C327.1	3	2	1
C327.2	3	3	2
C327.3	3	2	3

COURSE OVERVIEW

A Big Data Analytics course is a practical component accompanying a Big Data Analytics programming course that focuses on hands-on experience with the Big Data Analytics programming language. The course covers topics such as setting up the Big Data Analytics development environment and basic syntax, Students gain practical experience by working on programming exercises and projects, applying their knowledge to solve real-world problems. The course emphasizes writing clean, efficient, and well-structured code and provides students with the skills and confidence to develop Big Data Analytics applications. By the end of the course, students should be able to design and implement Big Data Analytics programs, effectively utilize Python, handle exceptions, and work with files and data input/output. Big data involves massive amounts of information, often ranging from terabytes to petabytes. Data is generated and collected at high speed. This can be from various sources like social media, sensors, and other real-time data streams.

LIST OF EXPERIMENTS MAPPED WITH COs

	data manipulation and analysis in Python.		
7	Write Program Python and PySpark to demonstrate data ingestion into Hadoop Distributed File System (HDFS) and Apache Kafka.	C327.1, C327.2	21
8	Program Real-life applications of big data span across various industries, addressing challenges related to large-scale data processing, analysis, and extraction of valuable insight	C327.1	27
9	Python program that simulates a basic big data processing scenario using Apache Spark for querying data.Large dataset (assuming a CSV file for simplicity), performs some data preprocessing, and then executes SQL-like queries on the data using Spark SQL.	C327.3	25
10	Python program using Apache Beam, a popular open-source data processing SDK, to create a basic data pipeline.	C327.3	35
11	Python program that demonstrates basic analytical operations using Apache Spark. This example uses PySpark to perform analytics on a large dataset.	C327.3	30

DOs and DON'Ts

DOs

1. Login-on with your username and password.

2. Log off the Computer every time when you leave the Lab.

3. Arrange your chair properly when you are leaving the lab.

4. Put your bags in the designated area.

5. Ask permission to print.

DON'Ts

1. Do not share your username and password.

2. Do not remove or disconnect cables or hardware parts.

3. Do not personalize the computer setting.

4. Do not run programs that continue to execute after you log off.

5. Do not download or install any programs, games or music on computer in Lab.

6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.

7. No Internet gaming activities allowed.

8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

GENERAL SAFETY PRECAUTIONS

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.

2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.

3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.

4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.

2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.

3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.

4. Call security and emergency department immediately:

Emergency : 200 (Reception)

Security : 248 (Gate No.1)

GUIDELINES TO STUDENTS FOR REPORT PREPARATION

All students are required to maintain a record of the experiments conducted by them.

Guidelines for its preparation are as follows:-

1) All files must contain a title page followed by an index page. ***The files will not be signed by the faculty without an entry in the index page.***

2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.

3) For each experiment, the record must contain the following

(i) Aim/Objective of the experiment

(ii) Pre-experiment work (as given by the faculty)

(iii) Lab assignment questions and their solutions

(iv) Test Cases (if applicable to the course)

(v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.

2. Students must ensure that their lab record is regularly evaluated.

LAB ASSESSMENT CRITERIA

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	**Exemplary (4)**	**Competent (3)**	**Needs Improvement (2)**	**Poor (1)**
AC1: Pre-Lab written work (this may be assessed through viva)	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: Program Writing/ Modeling	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied, Program/solution written is readable	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3:	Able to identify	Able to identify	Is dependent	Unable to

Identification & Removal of errors/ bugs	errors/ bugs and remove them	errors/ bugs and remove them with little bit of guidance	totally on someone for identification of errors/ bugs and their removal	understand the reason for errors/ bugs even after they are explicitly pointed out
AC4:Execution & Demonstration	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
AC5:Lab Record Assessment	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done	Less than 40 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done

	and output	with all variants of input and output	with all variants of input and output	with all variants of input and output

LAB EXPERIMENTS

LAB EXPERIMENT 1

OBJECTIVE: Write a Big data program to implement Python program that demonstrates a basic process of extracting value from big data using a machine learning model.

PRE-EXPERIMENT QUESTIONS:

1. What is method Big data program to implement ?
2. What are the requirements for method big data using a machine learning model. ?

BRIEF DISCUSSION AND EXPLANATION:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score


# Step 1: Load and preprocess big data
# Assuming 'big_data.csv' is a large dataset with features and target variable
data = pd.read_csv('big_data.csv')


# Preprocess data as needed (handle missing values, encode categorical variables, etc.)
# For simplicity, let's assume the target variable is binary (0 or 1)


# Step 2: Split data into training and testing sets
```

```
X = data.drop('target_variable', axis=1) # Features

y = data['target_variable']  # Target variable



X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)



# Step 3: Train a machine learning model (Random Forest in this example)

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)



# Step 4: Make predictions on the test set

y_pred = model.predict(X_test)



# Step 5: Evaluate the model's performance

accuracy = accuracy_score(y_test, y_pred)

print(f'Model Accuracy: {accuracy * 100:.2f}%')



# Step 6: Extract value based on model predictions

# Depending on your business case, you might use the model predictions to make decisions or take actions.
```

Additional steps for deployment, monitoring, and continuous improvement would be necessary in a real-world scenario.

Output:

```
print(f'Model Accuracy: {accuracy * 100:.2f}%')
```

```
Model Accuracy: 85.00%
```

LAB EXPERIMENT 2

OBJECTIVE:

Program that involves working with Big Data on a distributed computing platform like Apache Spark and comparing it with a Relational Database Management System (RDBMS)

PRE-EXPERIMENT QUESTIONS:

3. What is method Big data program to implement ?
4. What are the requirements for method big data using a machine learning model. ?

BRIEF DISCUSSION AND EXPLANATION

```
from pyspark.sql import SparkSession


# Create a Spark session
spark = SparkSession.builder.appName("BigDataExample").getOrCreate()


# Read a large dataset (assuming a Parquet file for illustration)
big_data_df = spark.read.parquet("big_data.parquet")


# Perform a simple data transformation
processed_data = big_data_df.filter(big_data_df['column_name'] > 50)
```

```
# Show the result

processed_data.show()



# Stop the Spark session

spark.stop()

import sqlite3



# Create a SQLite connection

conn = sqlite3.connect('database.db')

cursor = conn.cursor()



# Assuming you have a large table named 'big_data_table' with a column 'column_name'

# Execute a SQL query to process the data

cursor.execute("SELECT * FROM big_data_table WHERE column_name > 50")



# Fetch and print the result

result = cursor.fetchall()

for row in result:

    print(row)
```

```
# Close the database connection

conn.close()
```

LAB EXPERIMENT 3

OBJECTIVE:

Program for a Big Data Data Lakes scenario involves working with a distributed storage system and possibly using a query language like SQL or tools like Apache Spark. Below is a simplified example using Python, PySpark, and the Spark SQL API to demonstrate working with data stored in a Data Lake.

PRE-EXPERIMENT QUESTIONS:

1. What Data Lakes?
2. What is PySpark, and the Spark SQL API?

BRIEF DISCUSSION AND EXPLANATION:

```
from pyspark.sql import SparkSession


# Create a Spark session
spark = SparkSession.builder.appName("DataLakeExample").getOrCreate()


# Read data from a file in a Data Lake (Assuming a parquet file for illustration)
data_lake_df = spark.read.parquet("data_lake_file.parquet")


# Perform a basic data analysis using Spark SQL
data_lake_df.createOrReplaceTempView("data_lake_table")
```

```
result = spark.sql("SELECT column1, AVG(column2) as avg_column2 FROM data_lake_table GROUP BY column1")


# Show the result

result.show()


# Write the result back to the Data Lake (Assuming a parquet file for illustration)

result.write.parquet("output_result.parquet")


# Stop the Spark session

spark.stop()
```

In this example:

- `data_lake_file.parquet` represents a file stored in a Data Lake, and the program reads the data from it.
- The program then uses Spark SQL to perform a basic analysis (calculating the average of `column2` grouped by `column1`).
- The result is shown using the `show()` method.
- The processed data is written back to the Data Lake, represented by the `output_result.parquet` file.

POST EXPERIMENT QUESTIONS:

1. What is the output of the program? Explain the Python, PySpark, and the Spark?
2. What is Big Data Data Lakes scenario?

LAB EXPERIMENT 4

OBJECTIVE

Python program using PySpark, which is the Python API for Apache Spark, to demonstrate basic data processing:

PRE-EXPERIMENT QUESTIONS:

1. What is Python API ?
2. Explain basic data Processing?

BRIEF DISCUSSION AND EXPLANATION:

```
from pyspark.sql import SparkSession


# Create a Spark session
spark = SparkSession.builder.appName("BigDataProcessingExample").getOrCreate()


# Read a large dataset (assuming a CSV file for illustration)
big_data_df = spark.read.csv("big_data.csv", header=True, inferSchema=True)


# Perform a simple data transformation (example: calculate the sum of a column)
total_sum = big_data_df.agg({"column_name": "sum"}).collect()[0][0]


# Show the result
print(f"Total sum of 'column_name': {total_sum}")
```

```
# Stop the Spark session

spark.stop()
```

Output: In this example:

- **big_data.csv** represents a large dataset, and the program reads the data using PySpark.
- The program performs a basic data transformation, calculating the sum of a column ('column_name' in this example).
- The result is then printed.

Please note that you would need to replace "big_data.csv" and "column_name" with your actual dataset and column name. Additionally, in a real-world scenario, you would likely perform more complex transformations and analyses on the data.

The output of this program would be the calculated sum of the specified column, as indicated by the **print** statement. This is a basic example, and actual processing tasks would depend on the specific requirements and nature of your big data.

POST EXPERIMENT QUESTIONS:

1, What is CSV?

2. Explain Data transformation?

LAB EXPERIMENT 5

OBJECTIVE:

Python program using SQLAlchemy, a popular SQL toolkit and Object-Relational Mapping (ORM) library, to model data and interact with a relational database.

PRE-EXPERIMENT QUESTIONS:

1. What is Sql Toolkit and object-Oriented Mapping?
2. What Object-Relational Mapping (ORM) library, to model data and interact with a relational database.?

BRIEF DISCUSSION AND EXPLANATION:

```
from sqlalchemy import create_engine, Column, Integer, String, MetaData
from sqlalchemy.orm import declarative_base, Session


# Define the database connection
DATABASE_URL = "sqlite:///example.db"
engine = create_engine(DATABASE_URL, connect_args={"check_same_thread": False})


# Define the data model using SQLAlchemy ORM
Base = declarative_base()


class BigDataModel(Base):
    __tablename__ = "big_data_table"
```

```
    id = Column(Integer, primary_key=True, index=True)
    column1 = Column(String, index=True)
    column2 = Column(Integer)

# Create the database tables
Base.metadata.create_all(bind=engine)

# Create a session to interact with the database
db_session = Session(engine)

# Example: Inserting data into the database
new_data_entry = BigDataModel(column1="example_data", column2=42)
db_session.add(new_data_entry)
db_session.commit()

# Example: Querying data from the database
queried_data = db_session.query(BigDataModel).filter(BigDataModel.column1 == "example_data").first()
print("Queried Data:", queried_data.__dict__)
```

```
# Close the database session

db_session.close()
```

POST EXPERIMENT QUESTIONS:

1. How to do new data entry into the database and query?

2. Explain in Details of `BigDataModel` class using SQLAlchemy?

LAB EXPERIMENT 6

OBJECTIVE:

Python program that demonstrates basic data operations using the pandas library. Pandas is a popular library for data manipulation and analysis in Python.

PRE-EXPERIMENT QUESTIONS:

1. What are data Operations?
2. What is the data Manipulation?

BRIEF DISCUSSION AND EXPLANATION:

```
import pandas as pd



# Create a sample DataFrame

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Emma'],

        'Age': [25, 30, 22, 35, 28],

        'Salary': [50000, 60000, 45000, 70000, 55000]}



df = pd.DataFrame(data)



# Display the original DataFrame

print("Original DataFrame:")

print(df)

print("\n")
```

Data Operations:

```
# 1. Selecting columns
selected_columns = df[['Name', 'Age']]
print("Selected Columns:")
print(selected_columns)
print("\n")

# 2. Filtering data based on a condition
filtered_data = df[df['Age'] > 25]
print("Filtered Data (Age > 25):")
print(filtered_data)
print("\n")

# 3. Sorting by a column
sorted_data = df.sort_values(by='Salary', ascending=False)
print("Sorted Data (by Salary):")
print(sorted_data)
print("\n")
```

```
# 4. Grouping data and calculating aggregates

grouped_data = df.groupby('Age').mean()

print("Grouped Data (Average Salary by Age):")

print(grouped_data)

print("\n")


# 5. Adding a new column

df['Bonus'] = df['Salary'] * 0.1

print("DataFrame with Bonus Column:")

print(df)

print("\n")


# 6. Deleting a column

df = df.drop('Bonus', axis=1)

print("DataFrame after removing the Bonus column:")

print(df)

print("\n")


# 7. Renaming columns

df = df.rename(columns={'Age': 'Years'})

print("DataFrame after renaming the Age column to Years:")
```

```
print(df)
```

Make sure to install the pandas library before running this program if you haven't already:

POST EXPERIMENT QUESTIONS:

1.What is Pandas Library?

2.What is Data Frame**?**

LAB EXPERIMENT 7

OBJECTIVE:

Write Program Python and PySpark to demonstrate data ingestion into Hadoop Distributed File System (HDFS) and Apache Kafka.

PRE-EXPERIMENT QUESTIONS:

1. What is a Data ingestion?
2. What are the common operations in data ingestion?

BRIEF DISCUSSION AND EXPLANATION:

```
from pyspark.sql import SparkSession


# Create a Spark session
spark = SparkSession.builder.appName("DataIngestionToHDFS").getOrCreate()


# Read data from a CSV file (assuming it's a large dataset)
input_file_path = "your_large_dataset.csv"
data_df = spark.read.csv(input_file_path, header=True, inferSchema=True)


# Write the data to HDFS (assuming it's running on localhost)
hdfs_output_path = "hdfs://localhost:9000/user/your_username/your_output_path"
data_df.write.mode("overwrite").parquet(hdfs_output_path)
```

```
# Stop the Spark session
spark.stop()
```

// Ensure you have a running HDFS instance, and adjust the `input_file_path` and `hdfs_output_path` accordingly.

```
from kafka import KafkaProducer


# Create a Kafka producer
producer = KafkaProducer(bootstrap_servers='localhost:9092')


# Read data from a file (assuming a text file for simplicity)
input_file_path = "your_data_file.txt"


with open(input_file_path, 'r') as file:
    for line in file:
        # Send each line as a message to the Kafka topic
        producer.send('your_kafka_topic', value=line.encode('utf-8'))


# Close the Kafka producer
```

```
producer.close()
```

// Make sure you have a running Kafka broker and adjust the **input_file_path** and **your_kafka_topic** accordingly.

These are basic examples, and in real-world scenarios, you would likely deal with more complex data formats, configurations, and error handling. Additionally, you may need to consider tools like Apache NiFi for comprehensive data ingestion pipelines.

Ensure you have the required libraries installed before running the programs:

```
pip install pyspark kafka-python
```

POST EXPERIMENT QUESTIONS:

1. What is a kafka?
2. What are the common operations performed on a stack?

LAB EXPERIMENT 8

OBJECTIVE:

Program Real-life applications of big data span across various industries, addressing challenges related to large-scale data processing, analysis, and extraction of valuable insight

PRE-EXPERIMENT QUESTIONS:

1.What is large scale data Processing?

2.Extraction of value insights big Data?

BRIEF DISCUSSION AND EXPLANATION:

```
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import linear_kernel



# Load a sample e-commerce dataset (products and user interactions)

ecommerce_data = pd.read_csv("ecommerce_data.csv")



# Perform data preprocessing (cleaning, handling missing values, etc.)



# Create a TF-IDF vectorizer to convert product descriptions into numerical features

tfidf_vectorizer = TfidfVectorizer(stop_words='english')

tfidf_matrix = tfidf_vectorizer.fit_transform(ecommerce_data['product_description'].fillna(''))
```

```
# Calculate the cosine similarity between products based on their descriptions

cosine_similarity = linear_kernel(tfidf_matrix, tfidf_matrix)



# Function to get personalized product recommendations for a given product ID

def get_recommendations(product_id, cosine_sim=cosine_similarity):

    idx = ecommerce_data.index[ecommerce_data['product_id'] == product_id].tolist()[0]

    sim_scores = list(enumerate(cosine_sim[idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)[1:6]

    product_indices = [i[0] for i in sim_scores]

    return ecommerce_data['product_name'].iloc[product_indices]



# Example: Get recommendations for a specific product (change product_id accordingly)

product_id_to_recommend_for = 12345

recommendations = get_recommendations(product_id_to_recommend_for)



# Display the recommendations

print(f"Top 5 Recommendations for Product ID {product_id_to_recommend_for}:\n")

for i, product_name in enumerate(recommendations, start=1):

    print(f"{i}. {product_name}")
```

POST EXPERIMENT QUESTIONS:

1.Cosine similarity is calculated between products based on their descriptions to identify similar products?

LAB EXPERIMENT 9

OBJECTIVE:

Python program that simulates a basic big data processing scenario using Apache Spark for querying data.Large dataset (assuming a CSV file for simplicity), performs some data preprocessing, and then executes SQL-like queries on the data using Spark SQL.

PRE-EXPERIMENT QUESTIONS:

1. Explain basic big data processing scenario?
2. How Apache Spark for querying data works?

BRIEF DISCUSSION AND EXPLANATION:

```
from pyspark.sql import SparkSession


# Create a Spark session
spark = SparkSession.builder.appName("BigDataQueryExample").getOrCreate()


# Read a large dataset (assuming a CSV file for illustration)
big_data_df = spark.read.csv("big_data.csv", header=True, inferSchema=True)


# Display the original DataFrame
print("Original DataFrame:")
big_data_df.show(truncate=False)
print("\n")
```

```
# Perform data preprocessing (select relevant columns, filter data, etc.)
processed_data = big_data_df.select("column1", "column2").filter(big_data_df["column2"] > 50)

# Display the processed DataFrame
print("Processed DataFrame:")
processed_data.show(truncate=False)
print("\n")

# Execute SQL-like queries on the data using Spark SQL
processed_data.createOrReplaceTempView("processed_data_table")
query_result = spark.sql("SELECT column1, AVG(column2) as avg_column2 FROM processed_data_table GROUP BY column1")

# Display the result of the query
print("Query Result:")
query_result.show(truncate=False)

# Stop the Spark session
spark.stop()
```

LAB EXPERIMENT 10

OBJECTIVE:

Python program using Apache Beam, a popular open-source data processing SDK, to create a basic data pipeline.

PRE-EXPERIMENT QUESTIONS:

1. What is Apache beam?

2. Popular open source data processing?

BRIEF DISCUSSION AND EXPLANATION:

```
import apache_beam as beam



# Sample data (replace this with your actual data source)

data = [

    {'user_id': 1, 'event_type': 'click'},

    {'user_id': 2, 'event_type': 'purchase'},

    {'user_id': 3, 'event_type': 'click'},

    # ... more data ...

]



# Apache Beam pipeline

with beam.Pipeline() as pipeline:

    # Step 1: Read data from a source (replace 'data' with your actual data source)
```

```
events = pipeline | 'ReadEvents' >> beam.Create(data)


# Step 2: Apply transformations (example: count events by type)
event_counts = (
    events
    | 'MapEventType' >> beam.Map(lambda event: (event['event_type'], 1))
    | 'CountByEventType' >> beam.CombinePerKey(sum)
)


# Step 3: Write the results to an output (replace 'output' with your actual output destination)
event_counts | 'WriteResults' >> beam.io.WriteToText('output')
```

The `ReadEvents` step reads data from a source. Replace `'data'` with your actual data source, such as reading from a file, a database, or a streaming source.

- The **MapEventType** step applies a transformation to convert each event into a key-value pair with the event type as the key and 1 as the value.
- The **CountByEventType** step uses **CombinePerKey** to count the occurrences of each event type.
- The **WriteResults** step writes the final results to an output destination. Replace **'output'** with your actual output destination, which could be a file, database, or another storage system.

POST EXPERIMENT QUESTIONS:

1. How to read file from database?
2. Explain database on other storage system?

LAB EXPERIMENT 11

OBJECTIVE:

Python program that demonstrates basic analytical operations using Apache Spark. This example uses PySpark to perform analytics on a large dataset.

PRE-EXPERIMENT QUESTIONS:

1.what are analytical operations?

2.Explain PySpark to Perform analytical operations?

BRIEF DISCUSSION AND EXPLANATION:

```
from pyspark.sql import SparkSession

from pyspark.sql.functions import col



# Create a Spark session

spark = SparkSession.builder.appName("BigDataAnalyticsExample").getOrCreate()



# Read a large dataset (assuming a CSV file for illustration)

big_data_df = spark.read.csv("big_data.csv", header=True, inferSchema=True)



# Display the original DataFrame

print("Original DataFrame:")
```

```
big_data_df.show(truncate=False)

print("\n")


# Analytical Operations:


# 1. Calculate the total count of rows

total_rows = big_data_df.count()

print(f"Total Rows: {total_rows}\n")


# 2. Calculate the summary statistics for numerical columns

summary_stats = big_data_df.describe().show(truncate=False)

print("Summary Statistics:")

print(summary_stats)

print("\n")


# 3. Group by a categorical column and calculate the average of a numerical column

average_by_category = big_data_df.groupBy("category").agg({"value": "avg"})

print("Average Value by Category:")

average_by_category.show(truncate=False)

print("\n")
```

```
# 4. Filter data based on a condition

filtered_data = big_data_df.filter(col("value") > 50)

print("Filtered Data (Value > 50):")

filtered_data.show(truncate=False)

print("\n")



# 5. Calculate the correlation between two numerical columns

correlation = big_data_df.stat.corr("value1", "value2")

print(f"Correlation between Value1 and Value2: {correlation}\n")



# Stop the Spark session

spark.stop()
```

Discussion:

- - *big_data.csv* represents a large dataset with various columns.
- The program uses PySpark to perform basic analytical operations, including calculating the total count of rows, summary statistics, grouping by a categorical column, filtering data based on a condition, and calculating the correlation between two numerical columns.

POST EXPERIMENT QUESTIONS:

1. filtering data based on a condition, and calculating the correlation between two numerical columns?
2. a large dataset with various columns?

This lab manual has been updated by

Prof. Pooja Khot

Crosschecked By

HOD CSE

Please spare some time to provide your valuable feedback.