Introduction

An e-commerce company, "ShopEase," wants to analyze its sales data to improve business decisions. The store has captured some data from their operations but have no idea how to use their data. This is your chance as a data engineer to help make sense of the data they have.

Learning Goals

In the following exercises, you will

- Learn to use advanced python scripts to interact with a database, manipulate data and conduct data analysis
- Write advanced SQL queries to create and modify tables, as well as analyze the resulting data.
- Perform database optimization to improve the efficiency of queries

Data

You have been provided with data in csv format corresponding to the database tables you need to create to perform the exercises that follow. The data can be found in the folder labelled ShopEase Data.

Due Date: Wednesday, 13th November 2024 before 8:00 AM

Lab Exercises

Lab Exercise 1: Advanced Python with Pandas and NumPy

Objective: Enhance data manipulation and analysis skills using Pandas and NumPy.

Tasks:

1. **Load Data:** Load the sales data from the ShopEase database into a Pandas DataFrame.
2. **Data Cleaning:** Handle missing values, remove duplicates, and convert data types appropriately.
3. **NumPy Operations:** Use NumPy to calculate the total revenue for each order and add it as a new column in the DataFrame.
4. **Data Transformation:** Create new columns for year, month, and day extracted from the order_date column using Pandas.
5. **Visualization:** Plot the monthly sales trend using Pandas' plotting capabilities.
6. **Database Connection:** Connect to the ShopEase database using Python's mysql.connector (for MySQL) or psycopg2(for Postgres database)
7. **Loading Data:** Populate the data into your database. Do not forget to create the schema in your database before populating the data with python.

Deliverable:
Jupyter notebook script that does all the task above. Provide images of a few rows of data in your database (show the columns as well)

Lab Exercise 2: Complex SQL Queries

Objective: Develop proficiency in writing and optimizing complex SQL queries.

Tasks:

1. **JOIN Operations:** Write SQL queries to join the orders, products, and customers tables to get a comprehensive view of each order.
2. **Subqueries:** Use subqueries to find the top 5 products with the highest sales in the last month.
3. **CASE Statements:** Implement CASE statements to categorize orders into 'High', 'Medium', and 'Low' revenue based on total order value.
4. **Query Optimization:** Analyze and optimize the queries using execution plans and indexes.

Deliverable:
Submit SQL queries for each task in Exercise 2.

Lab Exercise 3: SQL Window Functions

Objective: Understand and apply SQL window functions for advanced data analysis.

Tasks:

1. **Window Functions:** Write SQL queries using window functions like ROW_NUMBER(), RANK(), and DENSE_RANK() to rank products based on sales.
2. **Running Totals:** Calculate running totals of sales for each product category using SUM() window function.
3. **Partitioning:** Use the PARTITION BY clause to calculate the average order value for each customer.
4. **LAG and LEAD:** Use LAG() and LEAD() functions to compare each month's sales with the previous and next months.
5. **Complex Analysis:** Combine multiple window functions to analyze sales trends and customer behavior.

Deliverable:
Submit SQL queries for each task in Exercise 3.

Lab Exercise 4: Database Optimization

Objective: Learn techniques to optimize database performance.

Tasks:

1. **Performance Analysis:** Analyze the performance of existing queries on the ShopEase database using execution plans.
2. **Indexing:** Identify and create indexes on columns frequently used in WHERE clauses and JOIN operations.
3. **Schema Optimization:** Normalize the database schema to reduce redundancy and improve query performance (**hint: inventory and suppliers_data tables**).
4. **Partitioning:** Implement table partitioning strategies to manage large datasets efficiently.
5. **Profiling Tools:** Use database profiling tools to monitor query performance and identify bottlenecks.

Deliverables:

1. **SQL Queries**: Submit SQL scripts for creating indexes, making schema changes, and implementing partitioning.
2. **Screenshots**: Provide screenshots of execution plans (using `EXPLAIN`) before and after optimizations, as well as screenshots from profiling tools showing query performance improvements.

Lab Exercise 5: Integrating Python with SQL for Data Engineering (This can replace LAB5**)**

Objective: Combine Python and SQL skills to perform comprehensive data engineering tasks.

Tasks:

1. Create a **trigger** named `update_inventory` that activates after an **insert** operation on the **Order_Items** table. This trigger should:
   - Decrease the product inventory count in the **Inventories** table based on the ordered quantity.
   - Display a message if there is insufficient stock for a product.
2. Create a Stored Procedure to Update Customer Status that:
   - Accepts a **Customer ID** as a parameter.
   - Updates the **Customer Status** in the **Customers** table based on total order value: If total orders exceed $10,000, set the status to **"VIP"**. Otherwise, set the status to **"Regular"**.

Deliverable:
Submit your SQL scripts.