Activity 1: Writing Basic SQL Queries with Microsoft Copilot

Step 1: Review the scenario

To begin, review the following scenario related to building the "SmartShop Inventory System." Imagine you are a database engineer tasked with developing the SmartShop Inventory System for a fictional retail company, SmartShop. This system must manage inventory data across multiple stores, providing real-time insights into stock levels, sales trends, and supplier information. The company requires:

1. A database to store and retrieve inventory data efficiently.
2. Complex queries to analyze trends and relationships between products, sales, and suppliers.
3. Optimized database operations to ensure high performance and scalability.

Your goal is to leverage Microsoft Copilot to create, debug, and optimize SQL queries, ensuring the system meets performance and accuracy requirements. This project will span three activities and culminate in a comprehensive inventory management database.

SmartShop's initial requirements include:

- Retrieving product details such as name, price, and stock levels.
- Filtering products based on categories and availability.
- Sorting data for better readability.

Step 2: Generate basic SELECT queries with Copilot

To get started, you'll use Copilot to generate basic queries to meet these needs. Use Copilot to write a query to retrieve product details, including:

1. ProductName, Category, Price, and StockLevel.

Step 3: Implement filtering and sorting

Next, you'll work on filtering and sorting capabilities.

- Write queries with Copilot to filter:
    - Products in a specific category
    - Products with low stock levels
- Add sorting to display products by Price in ascending order.

Step 4: Save your work

By the end of this activity, you will have:

- Written basic SQL queries using Copilot to retrieve and organize inventory data.
- Queries prepared for extension and refinement in Activity 2.

Save all queries in your sandbox environment. These basic queries will be expanded and used in Activity 2.

Activity 2: Creating Complex SQL Queries with Microsoft Copilot

Step 1: Review the recap and new requirements

In Activity 1, you wrote basic SQL queries to retrieve and filter inventory data. Now, SmartShop has additional requirements:

- Analyzing sales trends by joining product and sales data.
- Generating reports on supplier performance using aggregate functions.
- Combining data from multiple tables to track inventory levels across stores.

SmartShop's new needs include:

- Tracking product sales by date and store.
- Identifying top-performing suppliers based on delivered stock.
- Combining inventory data across stores for consolidated reporting.

For this activity you will not need the actual data in the table. However, you should apply your understanding of queries and the structure of the table in our scenario to execute the queries.

Step 2: Write multi-table JOIN queries with Copitot

To get started, you'll use Copilot to generate multi-table JOIN queries.

1. Use Copilot to write the query to join the Products, Sales, and Suppliers tables.
2. Write a query to display:
    a. ProductName, SaleDate, StoreLocation, and UnitsSold.

Step 3: Implement nested queries and aggregation

Next, you'll implement nested queries and aggregation.

- Write subqueries with Copilot to:
    - Calculate total sales for each product.
    - Identify suppliers with the most delayed deliveries.
- Use aggregate functions (e.g., SUM, AVG, MAX) to analyze trends and summarize data.

Step 4: Save your work

By the end of this activity, you will have:

- Generated complex SQL queries to meet SmartShop's advanced requirements.
- A set of queries ready for debugging and optimization.

Save all complex queries in your sandbox environment. These will be debugged and optimized in Activity 3.

Activity 3: Debugging and Optimizing SQL Queries with Microsoft Copilot

Step 1: Review the recap

In Activity 2, you wrote complex SQL queries for SmartShop's inventory and sales data. Some queries may have inefficiencies or errors, including:

- Slow execution times for large datasets.
- Incorrect JOIN or WHERE clauses causing errors.
- Inefficient use of aggregate functions**.**

Step 2: Debug errors in SQL queries

Use Copilot to identify and correct errors in:

- JOIN statements causing mismatched results.
- Nested queries with incorrect syntax.

Step 3: Optimize query performance with Copilot

Use Copilot to suggest and implement optimizations such as:

- Adding appropriate indexes to frequently queried columns.
- Restructuring queries for improved execution plans.
- Reducing unnecessary computations.

Step 4: Test and validate the optimized queries

Finally, use Copilot to test and validate.

- Run the optimized queries and compare their performance with the original versions.
- Ensure the results are accurate and the execution time is reduced.

Step 5: Save your work

By the end of this activity, you will have:

— Debugged and optimized complex SQL queries;
— A final set of high-performing queries for SmartShop's database system.

Save the debugged and optimized queries in your sandbox environment. These queries will serve as the final deliverables for the SmartShop Inventory System.