BI Engineer Technical Assessment

Instructions:

Please complete the following tasks and submit your results in a shared folder or GitHub repository. You may use SQL, Python, or any other tools you are comfortable with. Include your code, a README file explaining your approach, and any necessary files or outputs.

Task 1: Data Modeling

Scenario:
You are tasked with designing a data warehouse to store information about an e-commerce platform's transactions, products, customers, and promotions. For additional context, the platform's primary product is selling clothing apparel (e.g. shirts, shoes, etc.). Your goal is to create a star schema that can be used for reporting and analytics.

Requirements:

1. Design a star schema that includes:
- A fact table for transactions.
- Dimension tables for customers, products, and promotions.

2. Create an ER diagram to visualize your schema.

3. Write SQL DDL statements to create the tables in a PostgreSQL database.

*Please include any columns related to an ecommerce clothing site in the fact and dimension tables that you feel would be necessary to have for accurate reporting and analytics.

Deliverables:

• A description of your schema, explaining why you chose this structure.
• The ER diagram.
• SQL DDL statements to create the tables.

Task 2: Advanced SQL Query and Optimization

Scenario:
You are working with a large dataset in a PostgreSQL database (version 15 and up) that contains customer orders. The sales team has noticed performance issues when running a specific query to generate their weekly sales reports. Your task is to optimize the query for better performance.

Dataset Schema:

• Orders table:
 - `order_id`: Unique identifier for the order (integer)
 - `order_date`: Date the order was placed (timestamp)
 - `customer_id`: Unique identifier for the customer (integer)
 - `product_id`: Unique identifier for the product (integer)
 - `quantity`: Number of products ordered (integer)
 - `total_amount`: Total amount of the order (float)

• Products table:
 - `product_id`: Unique identifier for the product (integer)
 - `product_name`: Name of the product (string)
 - `category`: Product category (string)

• Customers table:
 - `customer_id`: Unique identifier for the customer (integer)
 - `customer_name`: Name of the customer (string)
 - `country`: Country of the customer (string)

Original Query:

```
SELECT
   p.category,
   SUM(o.total_amount) AS total_sales,
   COUNT(o.order_id) AS total_orders,
   AVG(o.total_amount) AS avg_order_value
FROM Orders o
JOIN Products p ON o.product_id = p.product_id
GROUP BY p.category;
```

Requirements:

1. Identify the performance issues in the original query.
2. Optimize the query for faster execution. Consider using indexing, query rewriting, or other optimization techniques.
3. Write a brief explanation of your optimization strategy.

Bonus:

Create a sample dataset and test and compare the execution time of both the original and optimized queries.

Deliverables:

• Optimized SQL query.
• A brief explanation of the changes and performance improvements.
• **Bonus:** Performance comparison (before and after optimization).

Task 3: Complex Data Modeling for a Multi-Store Retail Chain

Scenario:
You are hired to design a data warehouse for a multi-store clothing retail chain that has stores globally. Each store operates independently but shares a centralized product catalog and customer base. The company wants a data warehouse that enables sales analysis at the store level, but also across all stores.

Requirements:

1. Design a data model that supports the following use cases:
- Track individual sales by store.
- Analyze sales trends across all stores and within specific regions.
- Track customer purchasing behavior across multiple stores.
- Analyze inventory at the product level for each store.

2. Your data model should include:
- A fact table for transactions.
- Dimension tables for stores, products, customers, and regions.
- Additional measures such as quantity sold, total sales, and discount applied.

3. Create an ER diagram to represent your model and write SQL DDL statements to implement it in a PostgreSQL database.

Deliverables:

• ER diagram of the schema.
• SQL DDL statements to create the tables.
• A brief explanation of how your model supports the use cases.

Submission Instructions:

• Include all necessary files in a single repository or folder.
• Provide a README file explaining how to run your code and any assumptions you made.
• Ensure your code is clean, well-commented, and follows best practices.