Data Engineering Take-Home Assignment

Scenario:

You are working as a Data Engineer at a company that processes large-scale streaming and batch data for e-commerce analytics. The company has a data lake and a data warehouse, and they need you to create a robust data pipeline to process, transform, and integrate data from various sources. Your solution must demonstrate proficiency in data pipeline creation, big data processing, and optimization techniques.

Assignment Goals:

1. **Data Ingestion**: Process raw datasets from multiple CSV files and load them into a staging area.
2. **Data Transformation**: Implement complex transformations, including cleaning, aggregation, and deduplication using Spark or Pandas.
3. **Data Storage**: Design a schema for a data warehouse and load the transformed data.
4. **SQL Analysis**: Write SQL queries for business insights.
5. **Optimization**: Use big data techniques to optimize query performance and data processing.
6. **Documentation**: Provide clear documentation for your pipeline and decisions.

Tasks

1. Data Ingestion

You are given the following raw data sources, all in CSV format:

- **transactions.csv**: Contains transactional data with details about customer purchases.
- **users.csv**: Contains user data with demographic information.
- **products.csv**: Contains product metadata.

Your task:

- Build an ingestion pipeline to load all the data into a staging area (use Pandas or PySpark).
- Handle missing values, data type inconsistencies, and invalid records.

2. Data Transformation

Transform the ingested data to create two cleaned tables:

1. **CustomerTransactionSummary**:
   - **Columns**: CustomerID, TotalSpent, TotalTransactions, LastTransactionDate
2. **ProductPerformance**:
   - **Columns**: ProductID, TotalSales, AveragePrice, UnitsSold

Bonus: Use PySpark's window functions or Pandas groupby to perform aggregations.

3. Data Storage

- Create a **star schema** for a data warehouse to store the transformed data.
- Design the following tables:
    1. **FactTransaction**: Includes transactional details.
    2. **DimCustomer**: Includes customer information.
    3. **DimProduct**: Includes product details.

Load the transformed data into these tables using an SQL-based database like SQLite, Postgres, or any data warehouse technology (e.g., Snowflake or BigQuery).

4. SQL Analysis

Write SQL queries to provide the following business insights:

1. Find the top 5 customers based on total spending.
2. Identify the best-selling product in each category.
3. Calculate the daily sales trend for the last 7 days.
4. Find the category with the highest sales in the last month.

5. Optimization

Optimize your transformations and queries:

- Use caching or persisting in PySpark.
- Partition data by date or category.
- Implement bucketing in PySpark for faster joins (optional).

6. Documentation

Provide a comprehensive README file that includes:

- An overview of your solution.
- Steps to set up and run the pipeline.
- Assumptions and challenges.
- Examples of how your solution can scale to larger datasets.

Dataset

Dataset 1: transactions.csv

Contains transaction details.

TransactionID	CustomerID	ProductID	Category	Quantity	Price	TransactionDate
T001	C001	P001	Electronics	1	499.99	2023-12-20T12:00:00Z
T002	C002	P002	Books	2	12.99	2023-12-21T14:30:00Z
T003	C001	P003	Home Appliances	1	299.99	2023-12-21T16:00:00Z

Dataset 2: users.csv

Contains user demographic information.

CustomerID	Name	Email	Age	Country
C001	Alice Doe	alice@example.com	32	USA
C002	Bob Smith	bob@example.com	45	UK
C003	Charlie Wu	charlie@example.com	29	Canada

Dataset 3: products.csv

Contains product metadata.

ProductID	ProductName	Category	Brand	Price
P001	Smartphone	Electronics	BrandX	499.99
P002	Novel	Books	BrandY	12.99
P003	Blender	Home Appliances	BrandZ	299.99

Deliverables

1. **Codebase**:
   - Python scripts or notebooks for the pipeline.
   - SQL scripts for queries.
2. **Output**:
   - Cleaned CSV files or parquet files for the transformed data.
   - Database dump or schema with example data.
3. **Documentation**:
   - README with instructions and explanation.
4. **Bonus**:
   - Optional visualizations or insights.

Evaluation Criteria

1. **Technical Skills**:
   - Effective use of Python and PySpark.
   - Efficient and correct SQL queries.
2. **Scalability**:
   - Solutions for handling large-scale data.
   - Partitioning and caching techniques.
3. **Code Quality**:
   - Modular, well-structured, and readable code.
4. **Documentation**:
   - Clarity, completeness, and professionalism.