ZRCH Assignment

1. Answers for bullet 6

Question 6.1 : What are the top 2 best-selling products?

Answer :

SELECT product_id, SUM(quantity) as total_quantity
FROM customer_transaction
GROUP BY product_id
ORDER BY total_quantity DESC
LIMIT 2;

```
shopsmart=# SELECT product_id, SUM(quantity) as total_quantity
FROM customer_transaction
GROUP BY product_id
ORDER BY total_quantity DESC
LIMIT 2;
 product_id | total_quantity
------------+----------------
 P001       |             15
 P003       |              6
(2 rows)
```

Question 6.2 : What is the average order value per customer?

Answer :

SELECT customer_id, AVG(quantity * price_per_unit) as avg_order_value
FROM customer_transaction
GROUP BY customer_id
ORDER BY customer_id;

```
shopsmart=# SELECT customer_id, AVG(quantity * price_per_unit) as avg_order_value
FROM customer_transaction
GROUP BY customer_id
ORDER BY customer_id;
 customer_id | avg_order_value
-------------+-----------------
 C001        |             700
 C002        |             600
 C003        |             600
(3 rows)
```

Question 6.3 : What is the total revenue generated per product category?

Answer :

SELECT category, SUM(quantity * price_per_unit) as total_revenue
FROM customer_transaction
GROUP BY category
ORDER BY category;

```
shopsmart=# SELECT category, SUM(quantity * price_per_unit) as total_revenue
FROM customer_transaction
GROUP BY category
ORDER BY category;
  category  | total_revenue
------------+---------------
 Category A |          2400
 Category B |           800
(2 rows)
```

2. Approach

Data Ingestion:

- Ingest customer transactions from JSON file and product catalog from CSV file using Python's **‘pandas’** library for efficient data handling.

Data Transformation:

- Clean and transform the data to ensure consistency and accuracy.
    - Handle missing values and incorrect data types (e.g., invalid prices).
    - Drop duplicate records to avoid redundancy and ensure data integrity.
    - Convert data types to the appropriate formats (e.g., price to numerical type, timestamp to datetime type).
- Merge the datasets based on the common field (product_id) to create a unified dataset.

Data Storage:

- Store the processed data in a PostgreSQL database using the **'Upsert'** (update by ‘transaction_id’ or insert) load type to efficiently manage existing records and new entries.
- Ensure relational storage for easy querying.

ETL Pipeline Implementation:

- Implement the ETL (Extract, Transform, Load) process using Python scripts.
- Use Docker to containerize the entire solution, including the ETL script, PostgreSQL database, and any dependencies.
- Provide a Docker Compose file to orchestrate the services and ensure easy setup and deployment.

3. Design Decisions

Data Ingestion:

- **Choice of pandas for data handling:** pandas is chosen due to its powerful data manipulation capabilities and ease of use with CSV and JSON formats.
- **Error handling:** Implement error handling mechanisms to manage issues during data ingestion, such as missing files or incorrect data formats.

Data Transformation:

- **Data Cleaning:**
    - Missing values are excluded from the dataset.
    - Invalid price entries are corrected or dropped.
- **Data Standardization:** Ensure all data types are consistent across datasets to facilitate merging and querying.

Data Storage:

- **PostgreSQL:** Chosen for its robustness, support for complex queries, and familiarity. It provides a relational structure suitable for the requirements of the assignment.
- **Upsert Load Type:** Utilizes the 'upsert' method to efficiently handle both new entries and updates to existing records, ensuring data consistency and reducing redundancy.

ETL Pipeline Implementation:

- **Python Scripts:** Python is chosen for its simplicity and extensive libraries for data processing and database interaction.
- **Dockerization:** Containerizing the solution ensures that the environment is consistent across different setups, making it easy to deploy and scale.
- **Docker Compose:** Simplifies the orchestration of multiple services, ensuring that the ETL process, database, and any other dependencies are properly configured and running.

4. Justifications

Data Ingestion:

- **‘Pandas’ Library:** Efficiently handles large datasets and provides various functions to read and process data from multiple formats.
- **Error Handling:** Ensures the pipeline is robust and can handle unexpected issues without crashing.

Data Transformation:

- **Cleaning and Standardization:** Essential for data quality and integrity. Ensures that downstream processes receive accurate and consistent data.
- **Merging Datasets:** Creates a unified view of the data, which is crucial for analysis and reporting.

Data Storage:

- **PostgreSQL:** Offers powerful querying capabilities and supports ACID transactions, ensuring data integrity. Its relational structure is perfect for handling organized data and complex connections.

ETL Pipeline Implementation:

- **Python Scripts:** Widely used in data engineering for its readability and rich ecosystem of libraries.
- **Dockerization:** Ensures the solution is portable, scalable, and easy to maintain. Docker containers provide isolation, ensuring that the environment remains consistent across different stages of development and deployment.
- **Docker Compose:** Simplifies the management of multi-container applications, ensuring all components of the pipeline work seamlessly together.

By following this approach and making these design decisions, the solution addresses all the requirements of the assignment, ensuring data is accurately ingested, transformed, stored, and made available for analysis and reporting.