

**Semester Project**

**Data Warehouse and Business Intelligence**

**Name: Muhammad Haroon Wajid**

**Roll no: 21i-1763**

**Section: DS-C**

**Building and Analysing a Near-Real-Time Data Warehouse**

**Prototype for METRO Shopping Store in Pakistan**

**Table of Contents**

1. **Project Overview**
2. **Key Components  
   2.1 Data Sources  
   2.2 Data Warehouse  
   2.3 MeshJoin Algorithm**
3. **Data Warehouse Schema  
   3.1 Dimension Tables  
   3.2 Fact Table**
4. **SQL Script: Create-DW.sql**
5. **MeshJoin Algorithm Overview  
   5.1 Detailed Steps of the MeshJoin Algorithm  
   5.1.1 Initialization  
   5.1.2 Buffer Management  
   5.1.3 Transaction Processing  
   5.1.4 Buffer Filling  
   5.1.5 Buffer Processing  
   5.1.6 Batch Insertion**
6. **Output of OLAP queries**
7. **Shortcomings of MESHJOIN**
8. **Learning Outcome from Project**

**1. Project Overview**

This project involves the integration of transactional data into a data warehouse using a custom implementation of the MeshJoin algorithm. The primary goal is to efficiently load and process large volumes of data from CSV files into a structured data warehouse, enabling advanced analytics and reporting.

**2. Key Components**

**2.1 Data Sources**

The project utilizes CSV files as the primary data source, including customer, product, and transaction data.

**2.2 Data Warehouse**

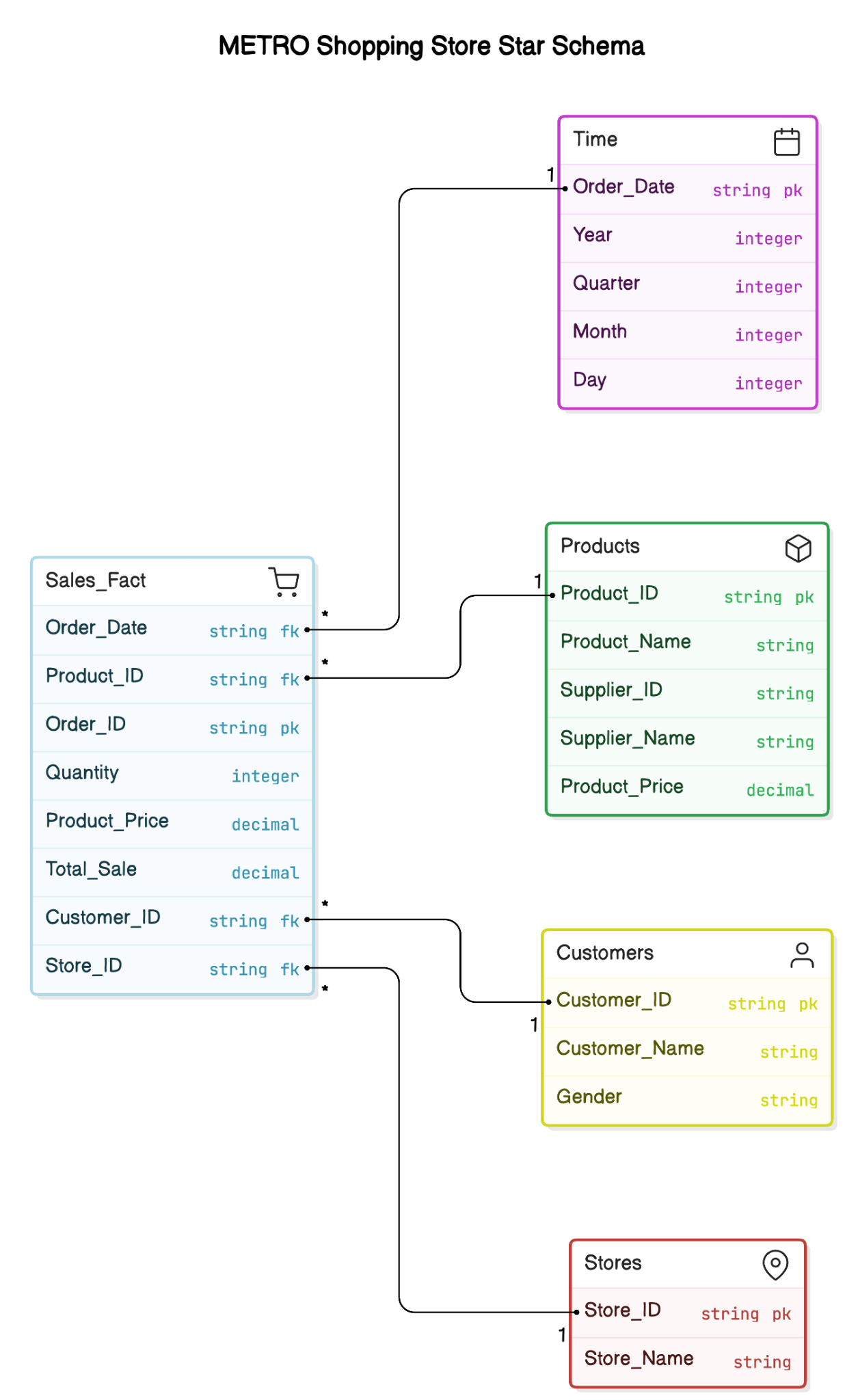
A MySQL-based data warehouse is used to store and manage the integrated data.

**2.3 MeshJoin Algorithm**

A custom implementation of the MeshJoin algorithm is used to process and load data into the data warehouse efficiently.

**3. Data Warehouse Schema**

The data warehouse schema is designed to support analytical queries and consists of dimension and fact tables.



**3.1 Dimension Tables**

**Customers\_Dim**

* Customer\_ID: Integer, Primary Key
* Customer\_Name: String
* Gender: String

**Products\_Dim**

* Product\_ID: Integer, Primary Key
* Product\_Name: String
* Supplier\_ID: Integer
* Supplier\_Name: String
* Product\_Price: Decimal

**3.2 Fact Table**

**Sales\_Fact**

* Order\_ID: Integer, Primary Key
* Order\_Date: Date
* Order\_Time: Time
* Product\_ID: Integer, Foreign Key
* Customer\_ID: Integer, Foreign Key
* Quantity: Integer
* Product\_Price: Decimal
* Total\_Sale: Decimal

**4. SQL Script: Create-DW.sql**

The Create-DW.sql file contains the SQL statements necessary to create the above tables and establish relationships between them.

**5. MeshJoin Algorithm Overview**

The MeshJoin algorithm is designed to efficiently process and integrate large volumes of data from multiple sources into a data warehouse. It is particularly useful for handling streaming data or large batch loads, where traditional join operations may be inefficient due to memory constraints.

**5.1 Detailed Steps of the MeshJoin Algorithm**

**5.1.1 Initialization**

* Establish a connection to the data warehouse using JDBC.
* Load master data from CSV files into memory (e.g., customer and product data).

**5.1.2 Buffer Management**

* Define a buffer size (BUFFER\_SIZE) to control the number of transactions processed in each batch, optimizing memory usage.

**5.1.3 Transaction Processing**

* Read transaction data from a CSV file line by line.
* Parse fields into a map with keys corresponding to column names.

**5.1.4 Buffer Filling**

* Add transaction maps to a queue (transactionQueue).
* Process the buffer once the queue reaches the defined size or all transactions are read.

**5.1.5 Buffer Processing**

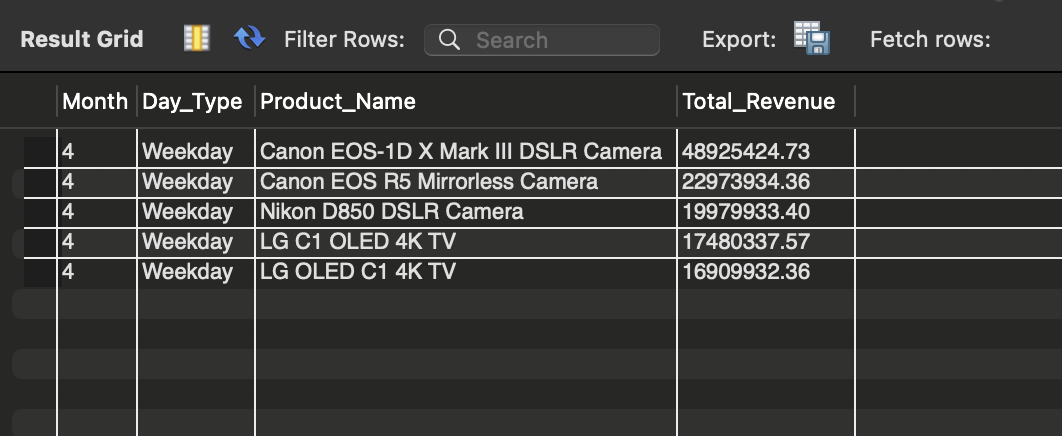
* Perform customer and product lookup for each transaction.
* Retrieve product prices and calculate the total sale amount.
* Skip transactions with invalid data and log issues.

**5.1.6 Batch Insertion**

* Prepare SQL INSERT statements for the Sales\_Fact table.
* Execute batch insert operations to minimize database writes.

**6. Output of OLAP queries**

**Q1. Top Revenue-Generating Products on Weekdays and Weekends with Monthly Drill-Down**

****

**Q2. Trend Analysis of Store Revenue Growth Rate Quarterly for 2017**

****

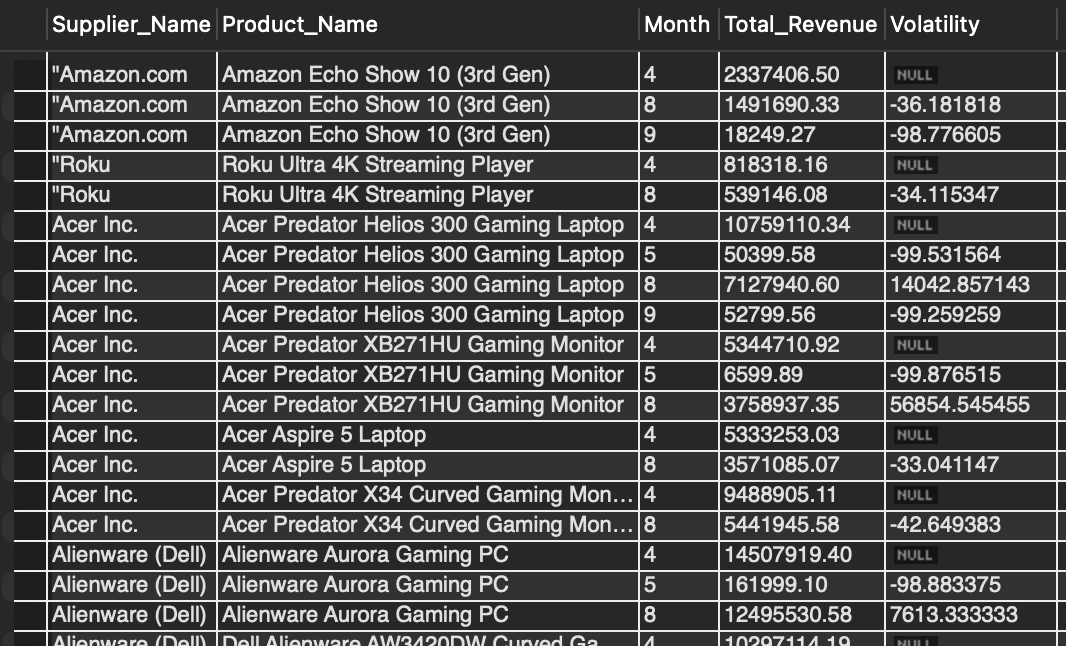
**Q3. Detailed Supplier Sales Contribution by Store and Product Name**

****

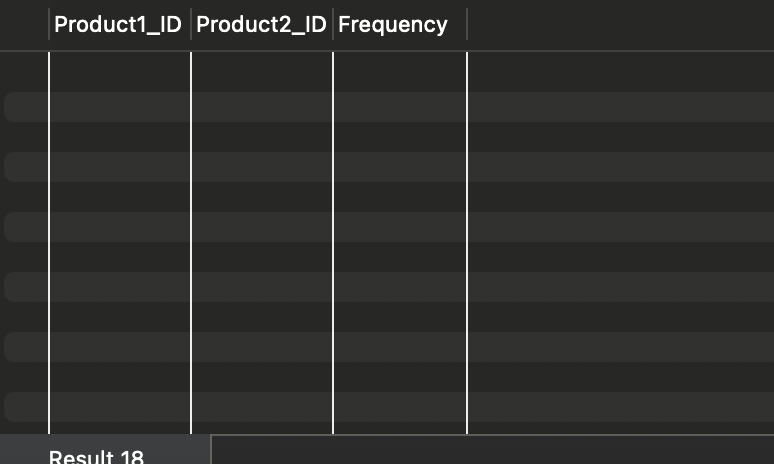
**Q4. Seasonal Analysis of Product Sales Using Dynamic Drill-Down**

****

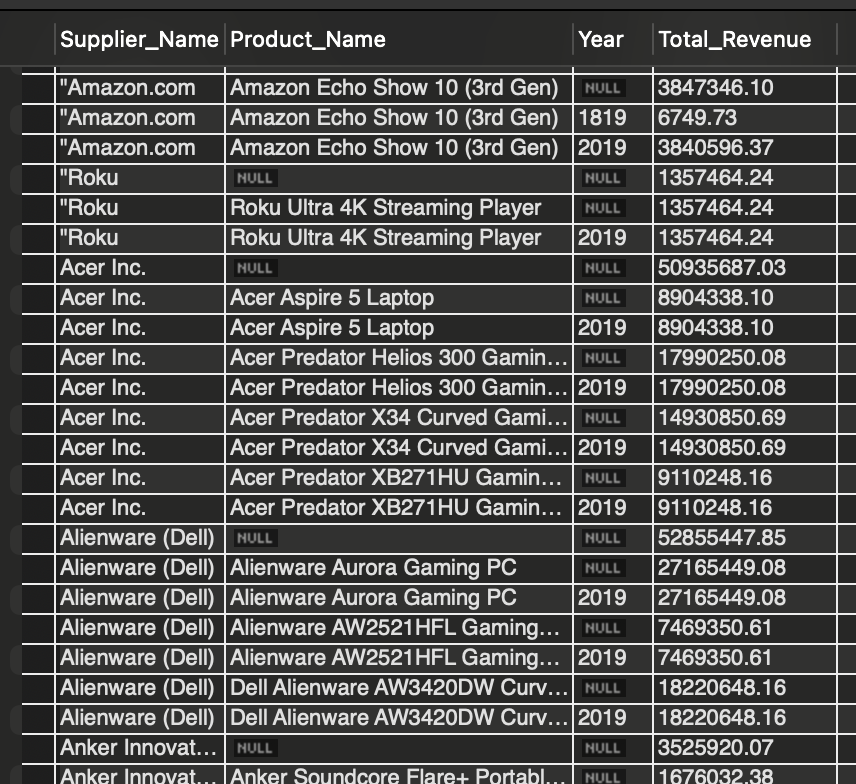
**Q5. Store-Wise and Supplier-Wise Monthly Revenue Volatility**

****

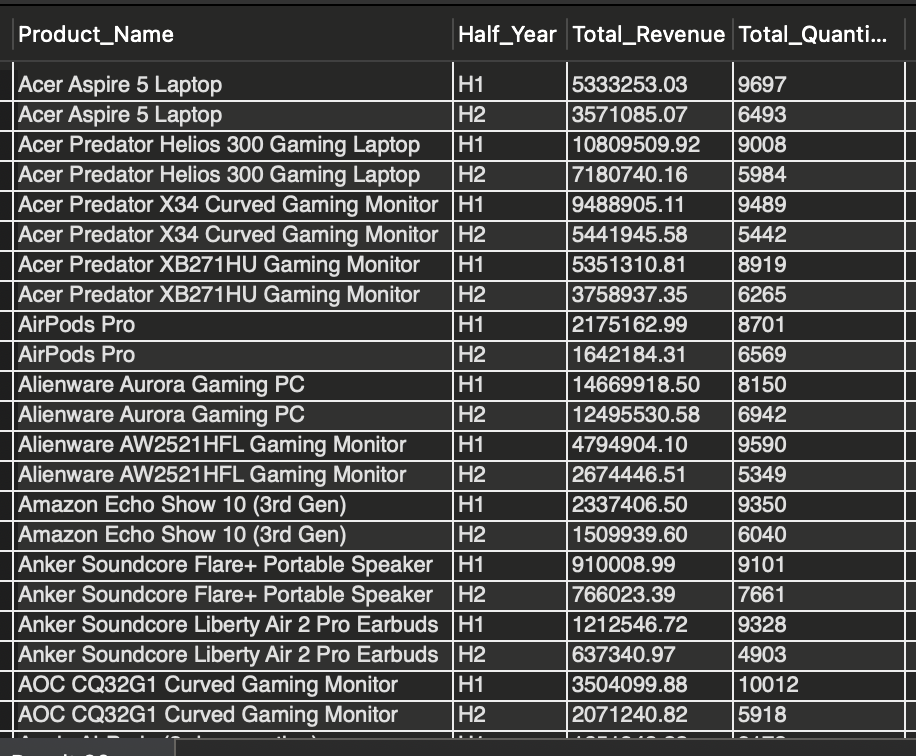
**Q6. Top 5 Products Purchased Together Across Multiple Orders (Product Affinity Analysis)**

****

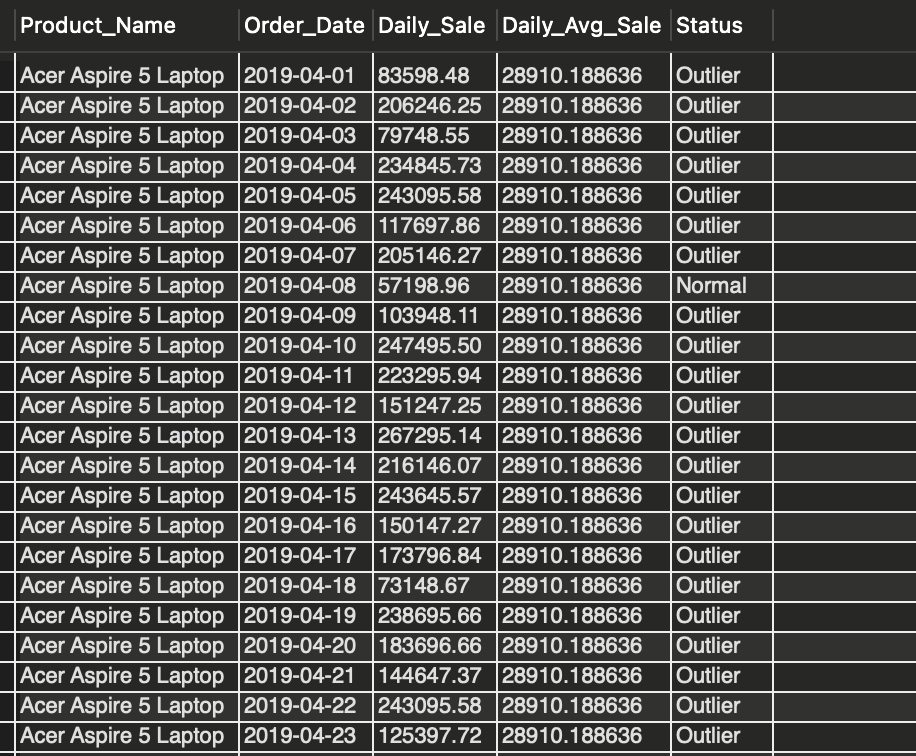
**Q7. Yearly Revenue Trends by Store, Supplier, and Product with ROLLUP**

****

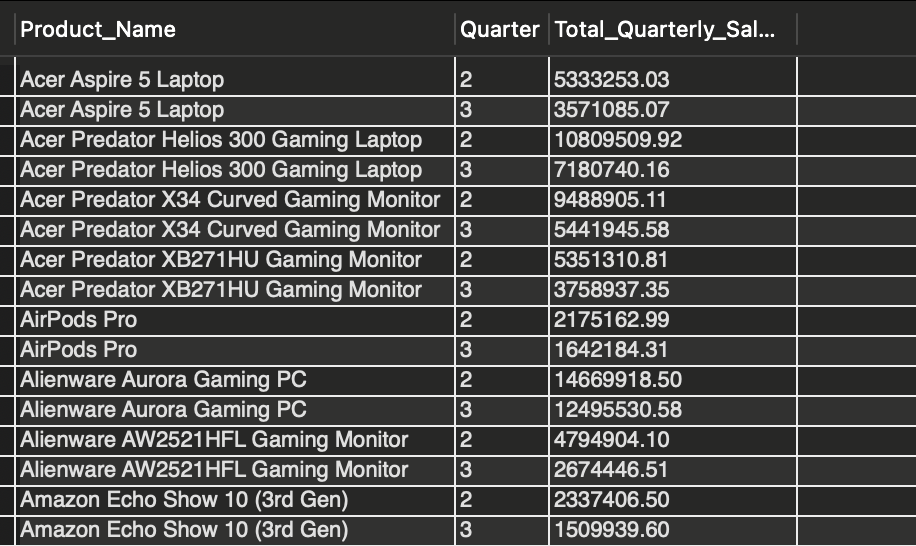
**Q8. Revenue and Volume-Based Sales Analysis for Each Product for H1 and H2**

****

**Q9. Identify High Revenue Spikes in Product Sales and Highlight Outliers**

****

**Q10. Create a View STORE\_QUARTERLY\_SALES for Optimized Sales Analysis**

****

**7. Shortcomings of the MeshJoin Algorithm**

**1. Memory Usage:**

Issue: When the MeshJoin algorithm processes data such, as customer and product information by storing it in memory storage system could potentially create a bottleneck if the size of the data surpasses the memory capacity available which could result in decreased performance or memory related issues.

**2. Latency:**

Issue: The algorithm handles transactions, in groups which causes a delay between when a transaction's received and when it is added to the data repository.This might not be ideal, for fulfilling real time data processing requirements.

**3. Error Handling and Logging:**

Issue: The problem lies in the algorithms error logging process which lacks automation, for rectifying or addressing data errors effectively.The system logs transactions, with missing prices. Fails to rectify them.

**8. Lessons Learned from the Project**

Through this project, I learned the importance of adopting efficient methods for handling datasets in data processing tasks. The implementation of the MeshJoin algorithm taught me how batch processing and buffer management can significantly enhance performance and resource utilization.

I also discovered how critical it is to maintain data integrity throughout the workflow. This experience emphasized the need for validating data before processing, managing missing values effectively, and documenting errors for future review—practices that ensure reliable analysis.

Working with SQL and JDBC in real-world scenarios was another valuable aspect of this project. I gained hands-on experience in managing database interactions, including establishing connections, using prepared statements, and applying batch processing techniques. These are essential skills for anyone pursuing data engineering or backend development roles.

Overall, this project was an eye-opening opportunity to understand the challenges of data engineering and to explore practical methods for addressing them effectively. It provided a solid foundation for building expertise in streamlining data workflows.