**Data Warehousing and Business Intelligence**

**Project**

**Ayaan Khan | DS-D | 22I-2066**

**Building and Analysing a Near-Real-Time Data Warehouse Prototype for METRO Pakistan**

**Project Report:**

**1. Project Overview**

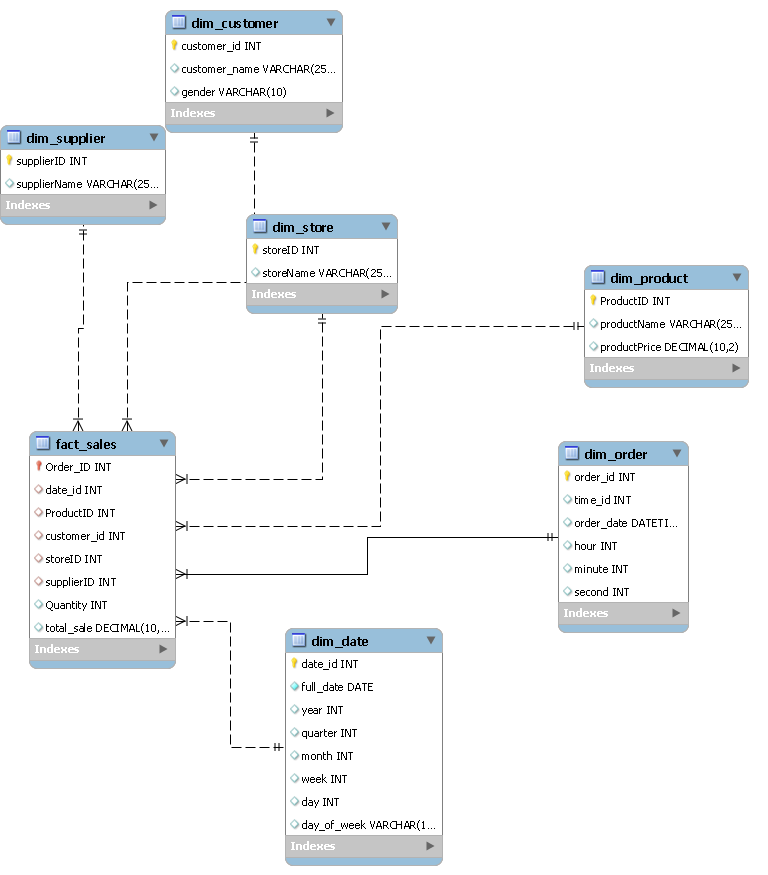
This project focuses on designing and implementing a near-real-time Data Warehouse (DW) prototype for METRO, a leading superstore chain in Pakistan. The objective is to facilitate business intelligence by analyzing customer behavior and sales patterns through real-time data integration. The star schema is used to organize data, and the MESHJOIN algorithm was implemented to enable the ETL(Extract-Transform-Load) process by enriching transactional data with master data.

The Data Warehouse allows advanced OLAP queries to be executed, providing insights such as revenue trends, seasonal product performance, and customer purchasing habits. The project will provide METRO with a platform to improve marketing strategies, manage inventory efficiently, and optimize operational processes.

**2. Schema for Data Warehouse (DW)**

The star schema implemented in this project is designed to support multidimensional analytical queries. The updated schema includes dimensions for product, customer, store, supplier, date, and time to provide granular insights:

**Schema Diagram:**



**Fact Table: fact\_sales**

Captures transactional data with attributes:

* **Order\_ID:** Unique identifier for each transaction.
* **date\_id:** Links to dim\_date.
* **ProductID:** Links to dim\_product.
* **customer\_id:** Links to dim\_customer.
* **supplier\_id:** Links to dim\_supplier
* **storeID:** Links to dim\_store.
* **Quantity:** Number of units sold.
* **total\_sale:** Calculated as Quantity × productPrice.

**Dimension Tables:**

1. **dim\_product:** Details about products, including ProductID, productName, productPrice, supplierID, and supplierName.
2. **dim\_customer:** Customer attributes like customer\_id, customer\_name, and gender.
3. **dim\_store:** Store-related data, including storeID and storeName.
4. **dim\_supplier:** Supplier information (supplierID and supplierName).
5. **dim\_date:** Date hierarchy with attributes like full\_date, year, quarter, month, week, day, and day\_of\_week.
6. **dim\_order:** Adds time granularity with attributes such as order\_id, time\_id, order\_date, hour, minute, and second.

This schema supports slicing, dicing, and drill-down queries for advanced analytical capabilities.

**3. MESHJOIN Algorithm**

The MESHJOIN algorithm was implemented as the central component of the ETL pipeline to enrich transactional data in real time. The algorithm integrates stream data (transactions) with master data (products and customers) and processes it for DW storage.

**Key Components:**

* **Stream Buffer:** Temporarily holds incoming transactional data from the TRANSACTIONS table.
* **Disk Buffer:** Cyclically loads partitions of master data (products and customers) into memory.
* **Hash Table:** Indexes transactional data for quick lookups.
* **Queue:** Tracks the order of transactions to ensure they are fully processed.

**Steps in MESHJOIN Implementation:**

1. **Load Transactions:** Stream transactions are read into the hash table and queue.
2. **Process Master Data:** Master data is divided into partitions and cyclically loaded into the disk buffer.
3. **Join and Enrich:** Transactions are matched with master data, enriched with attributes like productName and customer\_name, and calculated for total\_sale.
4. **Load into DW:** Enriched data is inserted into the DW, avoiding duplicates in dimensions.
5. **Repeat:** The process continues until all transactions are fully processed.
6. The staggered processing ensures efficient use of memory while minimizing disk I/O operations.

**4. Three Shortcomings of MESHJOIN**

1. **Memory Limitations:**

High memory dependency can lead to performance bottlenecks when processing large datasets or handling high data velocities.

1. **Latency for Late Transactions:**

Transactions must stay in memory until all master data partitions are processed, introducing delays for transactions arriving later in the cycle.

1. **Fixed Partition Sizes:**

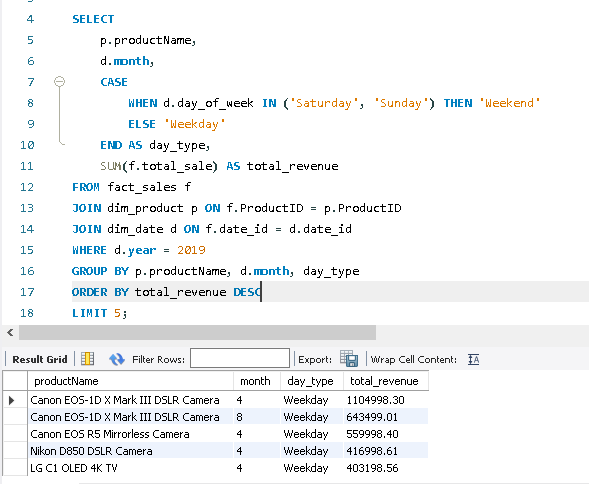
The need for fixed-size buffers makes the algorithm less adaptable to datasets with varying distributions or dynamic workloads.

**5. OLAP Queries**

Below are some key OLAP queries used for analysis.

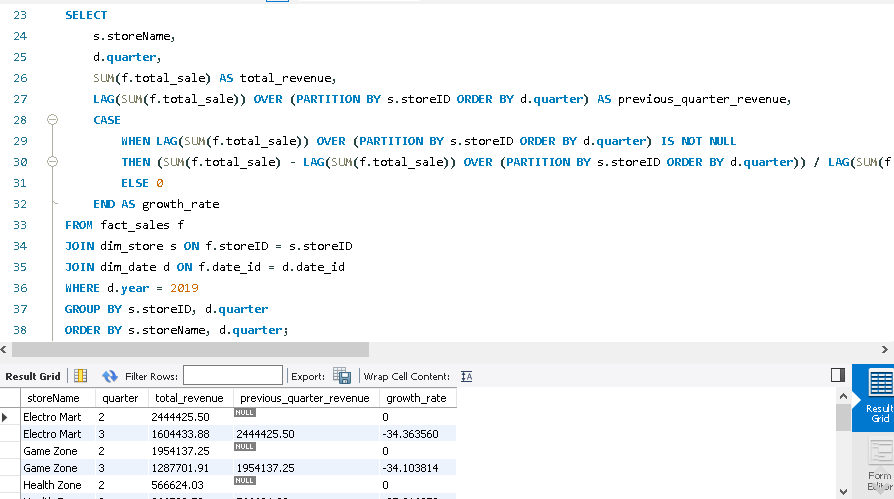
Q1. Top Revenue-Generating Products on Weekdays and Weekends with Monthly Drill-Down

Find the top 5 products that generated the highest revenue, separated by weekday and weekend sales, with results grouped by month for a specified year.



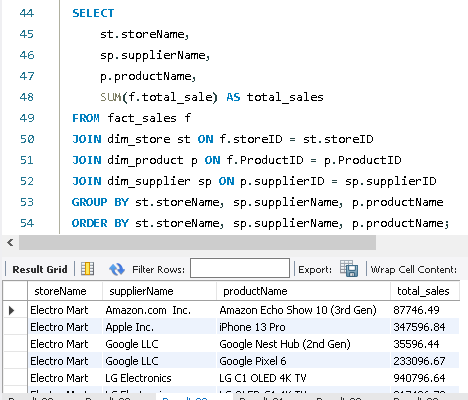
Q2. Trend Analysis of Store Revenue Growth Rate Quarterly for 2017

Calculate the revenue growth rate for each store on a quarterly basis for 2017.



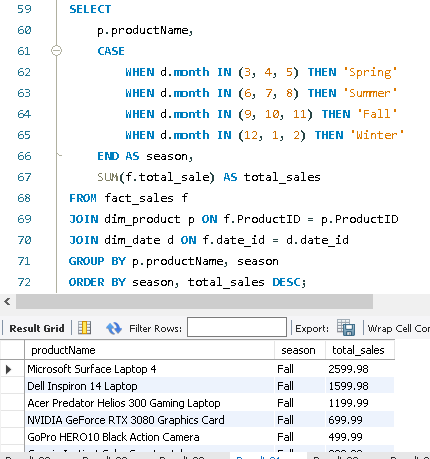
Q3. Detailed Supplier Sales Contribution by Store and Product Name

For each store, show the total sales contribution of each supplier broken down by product name. The output should group results by store, then supplier, and then product name under each supplier.



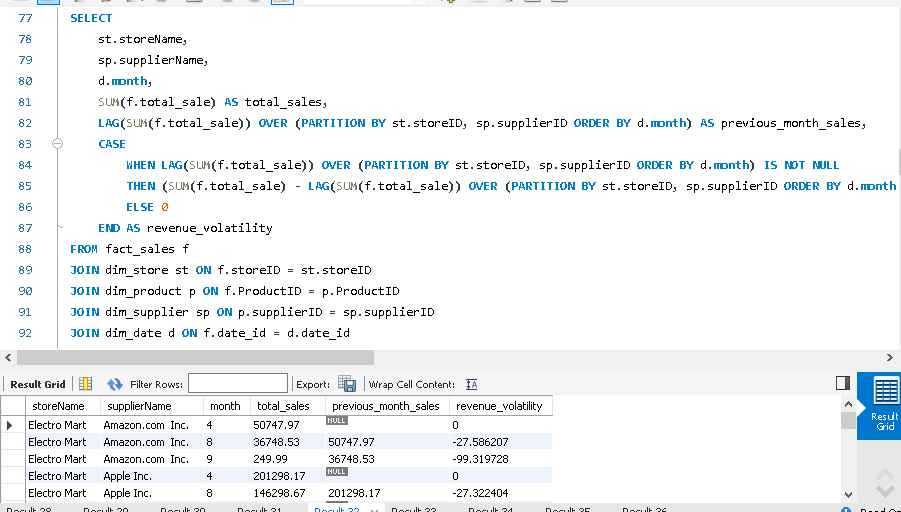
Q4. Seasonal Analysis of Product Sales Using Dynamic Drill-Down

Present total sales for each product, drilled down by seasonal periods (Spring, Summer, Fall, Winter). This can help understand product performance across seasonal periods.



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

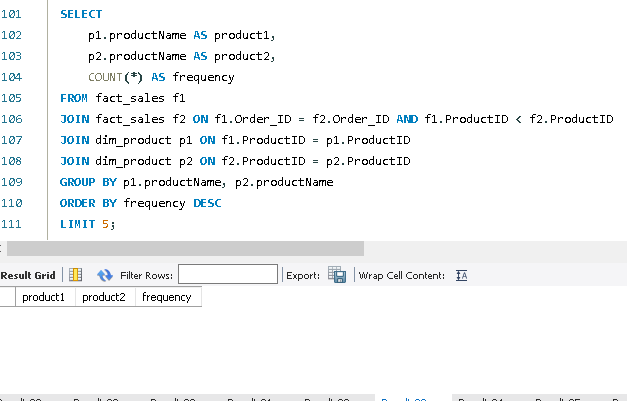
Calculate the month-to-month revenue volatility for each store and supplier pair. Volatility can be defined as the percentage change in revenue from one month to the next, helping identify stores or suppliers with highly fluctuating sales.



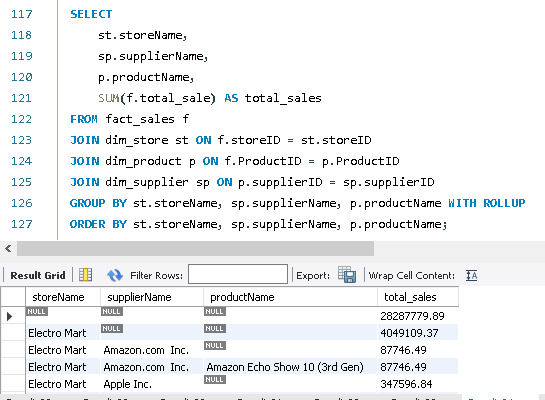
Q6. Top 5 Products Purchased Together Across Multiple Orders (Product Affinity Analysis)

Identify the top 5 products frequently bought together within a set of orders (i.e., multiple

products purchased in the same transaction). This product affinity analysis could inform potential product bundling strategies.

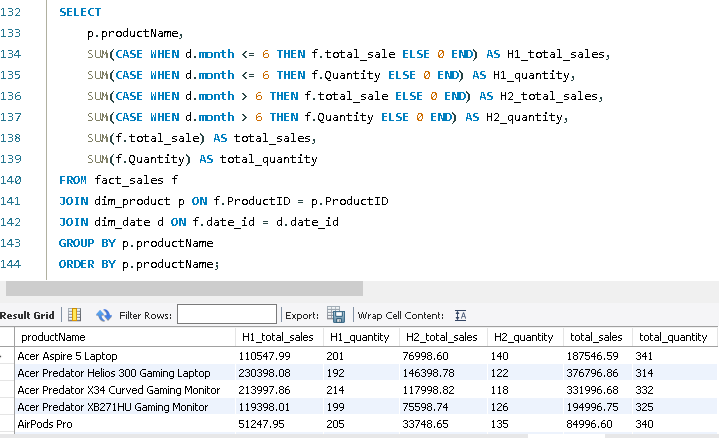


Q7. Yearly Revenue Trends by Store, Supplier, and Product with ROLLUP

Use the ROLLUP operation to aggregate yearly revenue data by store, supplier, and product, enabling a comprehensive overview from individual product-level details up to total revenue per store. This query should provide an overview of cumulative and hierarchical sales figures.

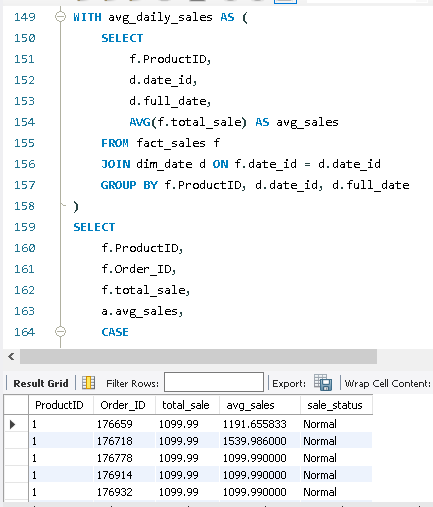
Q8. Revenue and Volume-Based Sales Analysis for Each Product for H1 and H2

For each product, calculate the total revenue and quantity sold in the first and second halves of the year, along with yearly totals. This split-by-time-period analysis can reveal changes in product popularity or demand over the year.



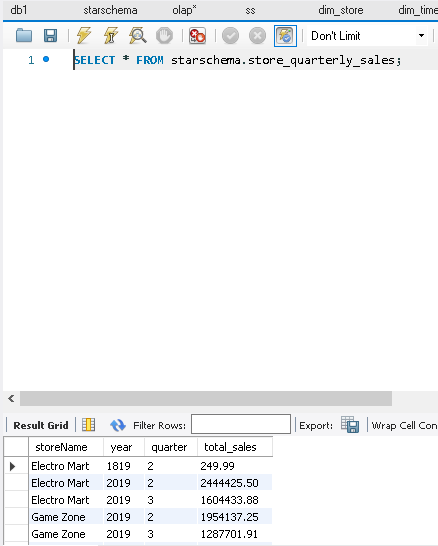
Q9. Identify High Revenue Spikes in Product Sales and Highlight Outliers

Calculate daily average sales for each product and flag days where the sales exceed twice the daily average by product as potential outliers or spikes. Explain any identified anomalies in the report, as these may indicate unusual demand events.



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

Create a view named STORE\_QUARTERLY\_SALES that aggregates total quarterly sales by store, ordered by store name. This view allows quick retrieval of store-specific trends across quarters, significantly improving query performance for regular sales analysis.



**6. What I Learned from the Project**

This project provided hands-on experience in designing, implementing, and analyzing data warehouses. Key learnings include:

* **Star Schema Design:** Creating an effective star schema for multidimensional analysis.
* **Real-Time ETL Implementation:** Applying the MESHJOIN algorithm for enriching and loading transactional data into the DW.
* **OLAP Query Development:** Writing and optimizing complex queries for business intelligence.
* **Data Cleaning and Preprocessing:** Ensuring data consistency using Python scripts for transformations.
* **Granular Time Analysis:** Including a time dimension to support detailed temporal insights.

The project bridged theoretical knowledge and practical applications which is essential for real-world challenges in data warehousing and analytics.