|  |  |  |
| --- | --- | --- |
| Project Report | November 26  2024 | |
|  | | Prepared By  22I-1953  Vanya Shafiq |

**Project Overview**

The project involves building and analyzing a near-real-time Data Warehouse (DW) prototype for the METRO Shopping Store in Pakistan. The goal is to facilitate the store in analyzing customer shopping behaviors to optimize selling strategies, such as offering promotions on products. This requires implementing a near-real-time ETL (Extraction, Transformation, and Loading) process using the MESHJOIN algorithm and a star schema to manage the data. The project includes tasks such as implementing ETL, creating a DW schema, and conducting advanced OLAP analysis.

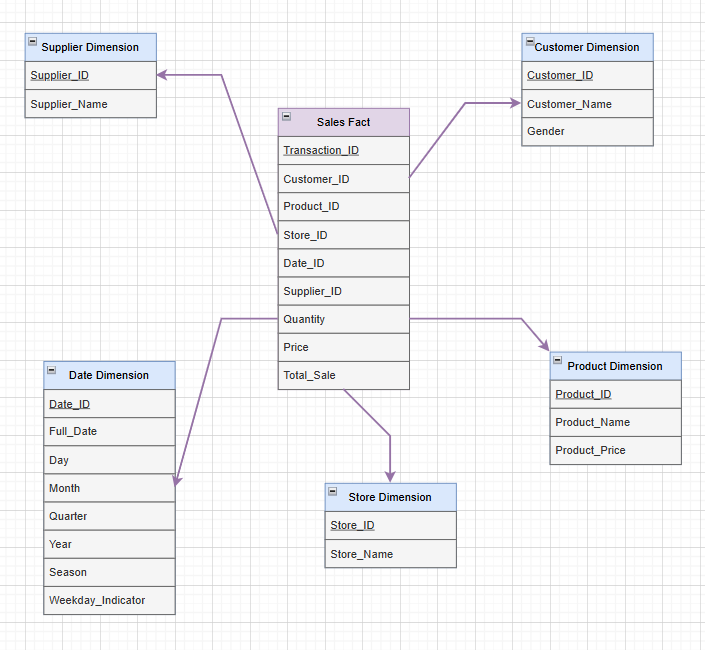
Key tasks include which I have implemented are:

1. **Schema Design**

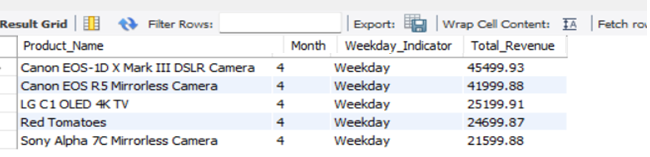
The dimension tables are as follows:

* 1. Customers
* Customer\_ID
* Customer\_Name
* Gender
  1. Products
* Product\_ID
* Product\_Name
* Product\_Price
  1. Date
* Date\_ID
* Full\_Date
* Day
* Month
* Quarter
* Year
* Season
* Weekday\_Indicator
  1. Suppliers
* Supplier\_ID
* Supplier\_Name
  1. Stores
* Store\_ID
* Store\_Name

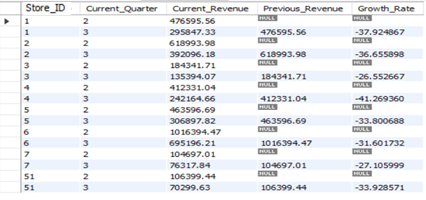
The fact table **Sales** is as follows:  
 Transaction\_ID   
 Customer\_ID  
 Product\_ID  
 Store\_ID  
 Date\_ID  
 Supplier\_ID  
 Quantity  
 Price  
 Total\_Sale



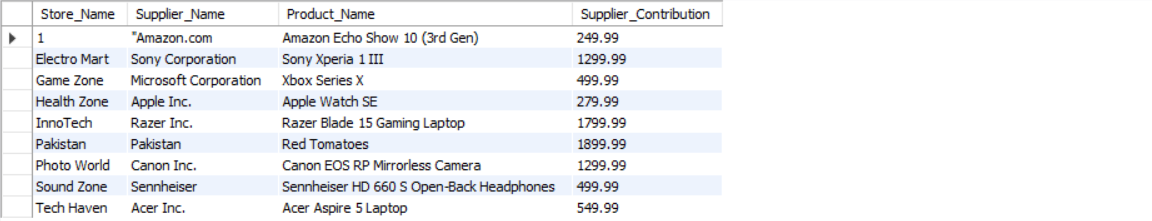
1. **ETL Pipeline with Extended MESHJOIN**
2. Extraction: Reading transactional data from the source table.
3. Transformation: Joining transactional data with master data (e.g., customer and product) using the MESHJOIN algorithm to enrich it with additional details.
4. Loading: Populating the DW while ensuring no duplication in dimension tables.
5. **OLAP Queries**
6. Top Revenue-Generating Products on Weekdays and Weekends with Monthly Drill-Down



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



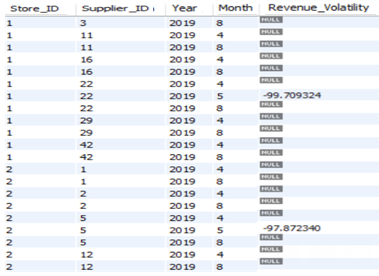
1. Detailed Supplier Sales Contribution by Store and Product Name



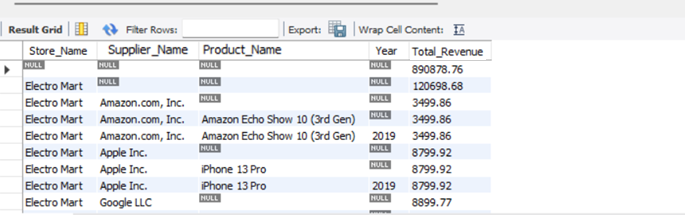
1. Seasonal Analysis of Product Sales Using Dynamic Drill-Down



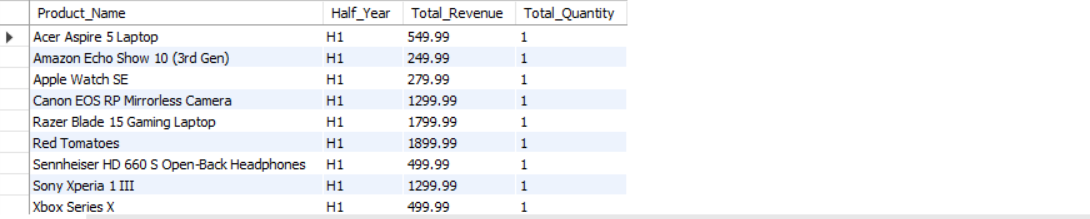
1. Store-Wise and Supplier-Wise Monthly Revenue Volatility



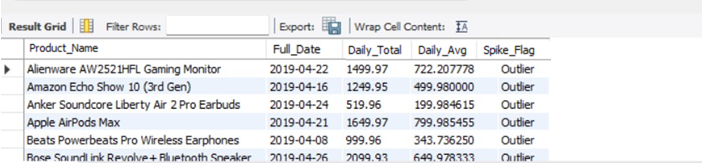
1. Top 5 Products Purchased Together Across Multiple Orders (Product Affinity Analysis)
2. Yearly Revenue Trends by Store, Supplier, and Product with ROLLUP



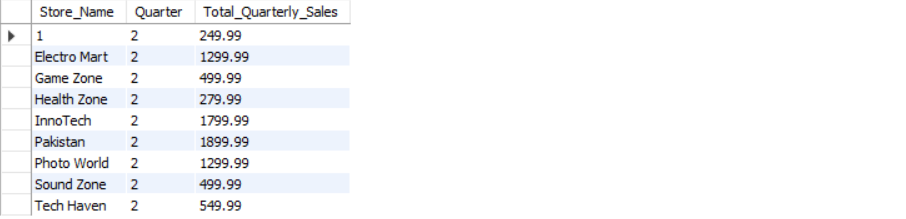
1. Revenue and Volume-Based Sales Analysis for Each Product for H1 and H2



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



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



1. **MESH JOIN**

**Implementation:**

Implementation of *MeshJoin* was done using the following steps:  
- Loading master data partitions cyclically into memory.  
- Joining streaming transactional data with master data using a hash table and a queue for efficient matching.  
- Generating enriched tuples with calculated fields like TOTAL\_SALE (quantity × price).

**Shortcomings:**

1. High memory usage

MESHJOIN needs to store many things in memory, like data from the stream and master data tables. If the data is too big, it can slow down the computer or even crash the system.

1. Struggles with fast data

If new transactions come in too quickly, MESHJOIN might not keep up. This can cause delays in processing or even some data being missed.

1. Problems with unbalanced data

The algorithm works best when data is evenly distributed. If some items (like popular products) appear much more often, it can slow down the process and make it less efficient.

**Learning Outcomes from this Project:**

1. I learnt how to create a system that stores and organizes large amounts of data so it can be easily analyzed.
2. I practiced the ETL process on how to take raw data, clean it up, add missing details, and load it into the warehouse.
3. Better understanding of business questions, like finding top-selling products or comparing sales across stores.
4. Improved my skills in using Java for data processing and SQL for managing and analyzing data.