**Name:** Afza Anjum **Roll #:** 21I-1724 **Section:** DS-C

**WAREHOUSE PROJECT REPORT**

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

**Prototype for METRO Shopping Store in Pakistan**

**1. Project Overview**

The project focuses on the development of a data warehouse (DW) prototype for METRO Shopping Store in Pakistan, aimed at optimizing data analysis and decision-making in the retail sector. The objective is to create an effective schema for sales data, implement an efficient method for data loading using the MESHJOIN algorithm, and perform OLAP (Online Analytical Processing) operations such as slicing, dicing, drill-down, and materializing views.

In this project, I have applied key principles of Data Warehousing & Business Intelligence to:

* Design a star schema with appropriate dimension and fact tables.
* Implement the MESHJOIN algorithm in Java to load transactional data into the data warehouse efficiently.
* Perform OLAP analysis to gain insights into sales, customer behavior, and store performance.

**2. Schema for Data Warehouse**

The **star schema** for the data warehouse is designed to represent the sales scenario and facilitate efficient querying and analysis. The schema includes the following components:

* **Dimension Tables**:
  1. **Product Dimension** (ProductID, ProductName, ProductPrice)
  2. **Customer Dimension** (CustomerID, CustomerName, Gender)
  3. **Store Dimension** (StoreID, StoreName)
  4. **Supplier Dimension** (SupplierID, SupplierName)
* **Fact Table**:
  1. **Sales Fact(Transaction Data)** (OrderID, OrderDate, ProductID, CustomerID, StoreID, SupplierID, Quantity, TotalSale)



Each of the **dimension tables** stores descriptive attributes for the respective entities, while the **fact table** stores the transactional data, including sales quantity and total sales amount. Foreign keys in the fact table refer to the primary keys in the dimension tables.

**3. MESHJOIN Algorithm**

The **MESHJOIN** algorithm is a technique used for efficiently joining transactional data with metadata (such as dimension tables). The process is optimized by hashing both the transactional data and metadata into smaller chunks, improving the performance of the join operation.

Steps Involved in MESHJOIN Algorithm:

1. **Load the Dimension Tables**: Product and customer details are loaded from the SQL database into hash maps (dimensionTable and customerTable) for quick lookup during the join.
2. **Process the Fact Data**: For each transaction in the fact table (processed within the sliding window), the algorithm:

* Retrieves the corresponding product and customer details using the product ID and customer ID.
* Calculates the total\_sales for the transaction based on the product's price and the quantity ordered.
* Writes the joined data (including both dimension and fact data) to an output file and inserts it into the database.

1. **Sliding Window Update**: As new transaction data is processed, the sliding window ensures that the most recent transactions are handled, and older records are removed, optimizing memory usage and processing time.The algorithm helps speed up the process of loading data into the warehouse by reducing the amount of data that needs to be compared during the join.

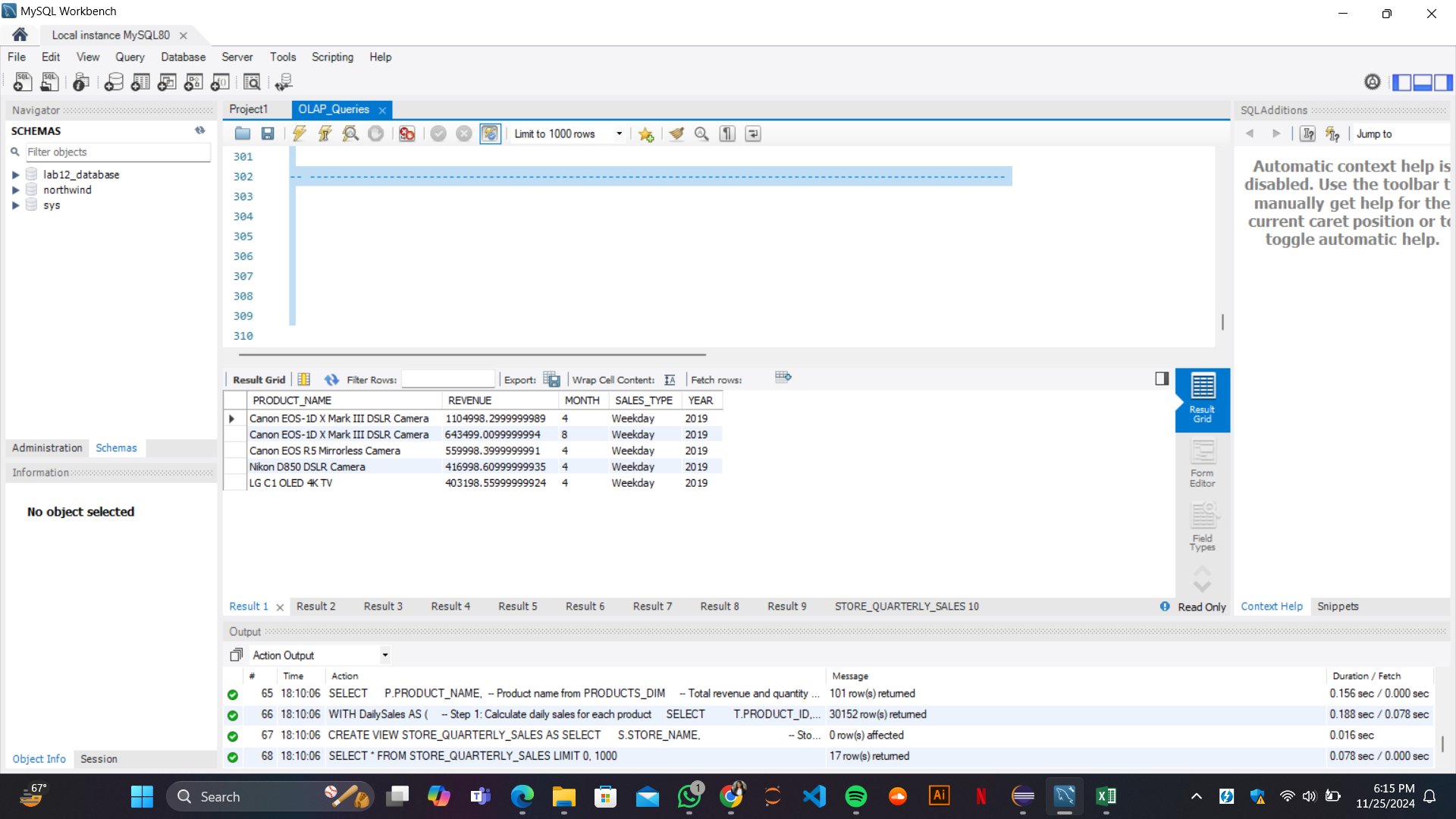
**4. Shortcomings of the MESHJOIN Algorithm**

While the MESHJOIN algorithm offers significant improvements in data loading performance, it has the following shortcomings:

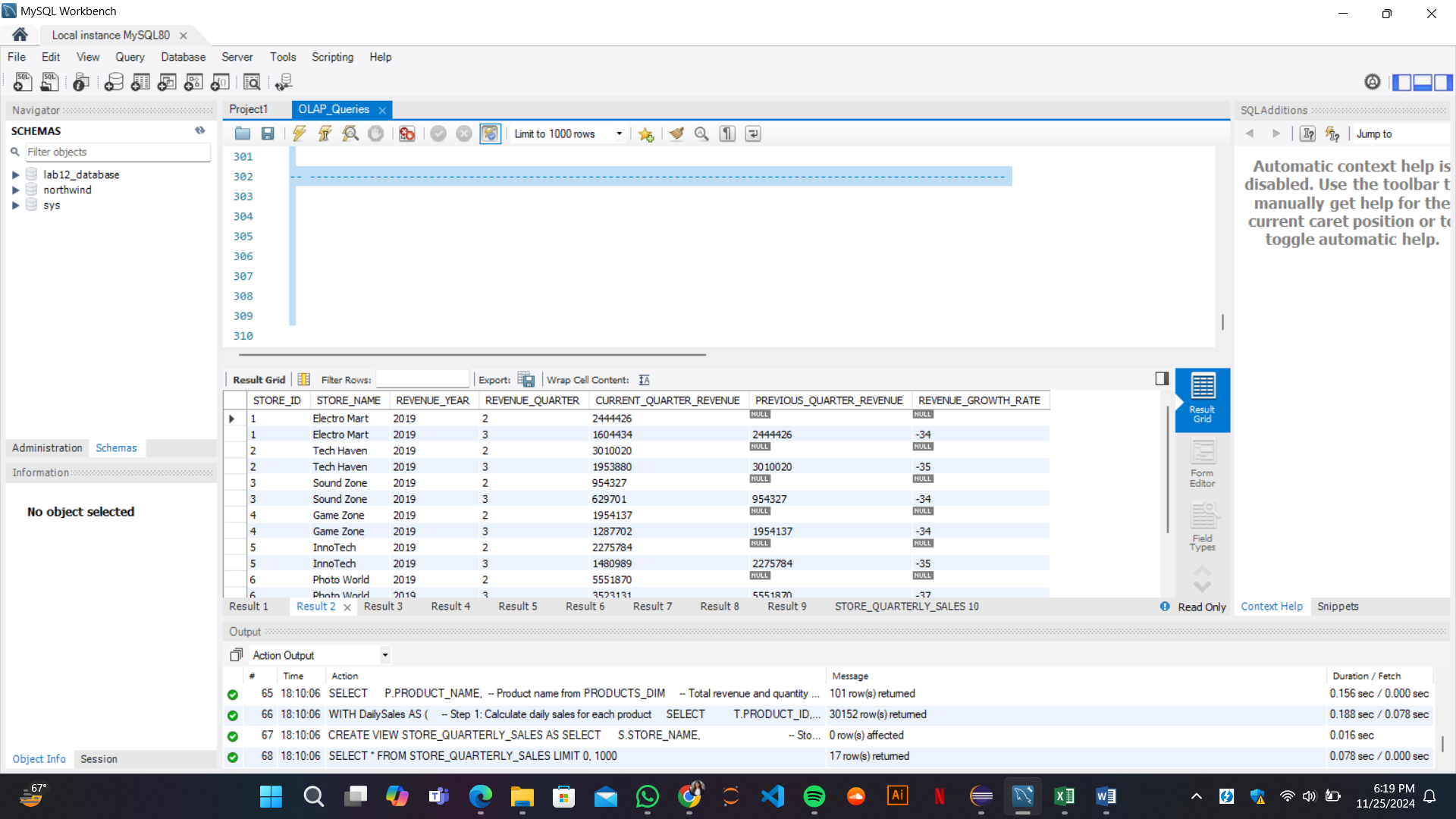
1. **Limited to Small Window Sizes**: The sliding window size should be carefully chosen to ensure that the system doesn't run out of memory or become inefficient with too large a window.
2. **Complexity of Join Operations**: For large-scale datasets, managing the dimension tables in memory and performing frequent lookups could still result in performance bottlenecks.
3. **Scalability**: While the algorithm works well for relatively small to medium-sized datasets, it may require further optimization or a distributed approach (e.g., using a big data processing framework like Apache Spark) for very large datasets.

**5. DW analysis**

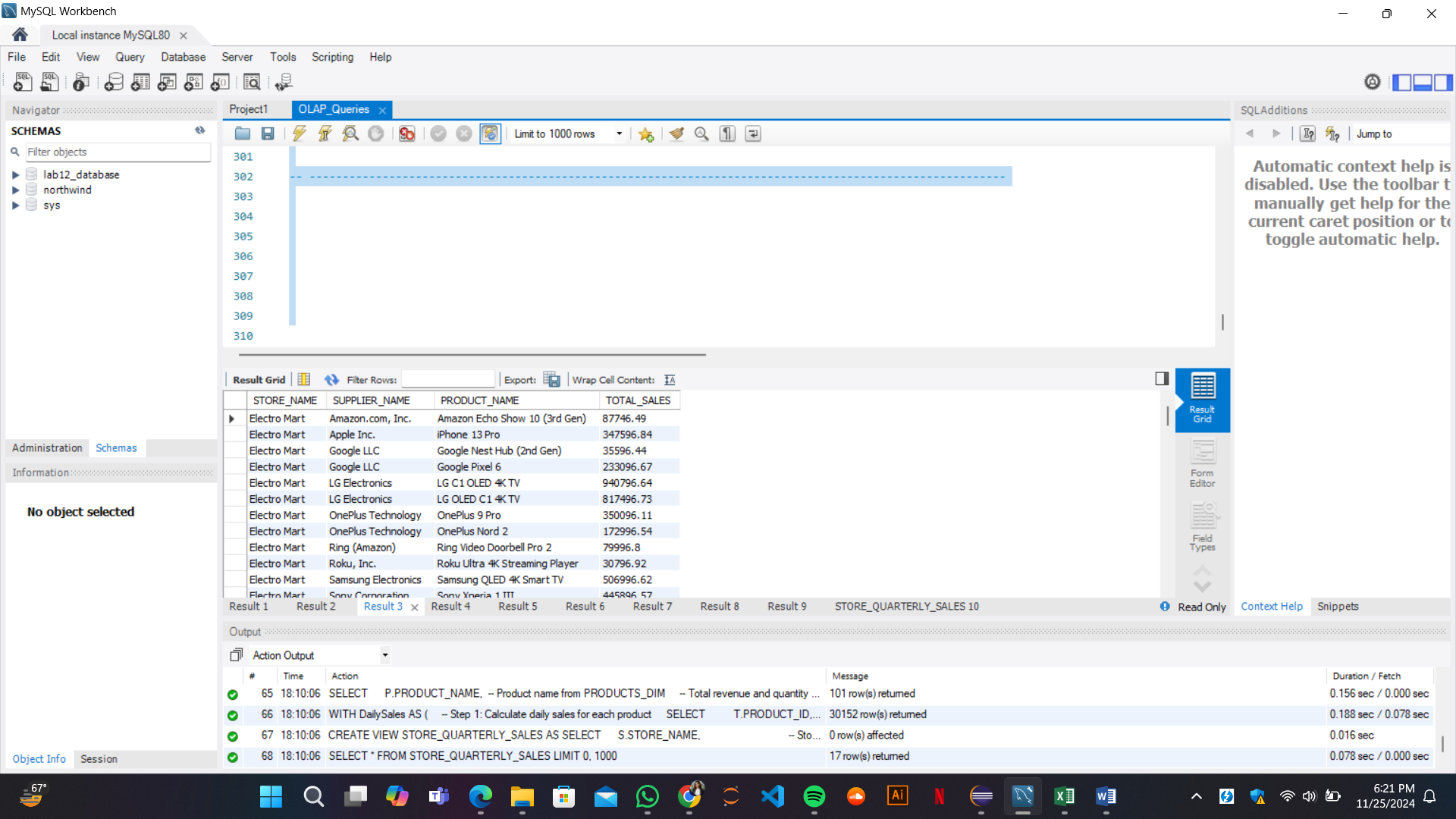
**Q1. Top Revenue-Generating Products on Weekdays and Weekends with Monthly Drill-Down:** Find the top 5 products that generated the highest revenue, separated by weekday and weekend sales, with results grouped by month for a specified year.



**Q2. Trend Analysis of Store Revenue Growth Rate Quarterly for 2019:** Calculate the revenue growth rate for each store on a quarterly basis for 2019.



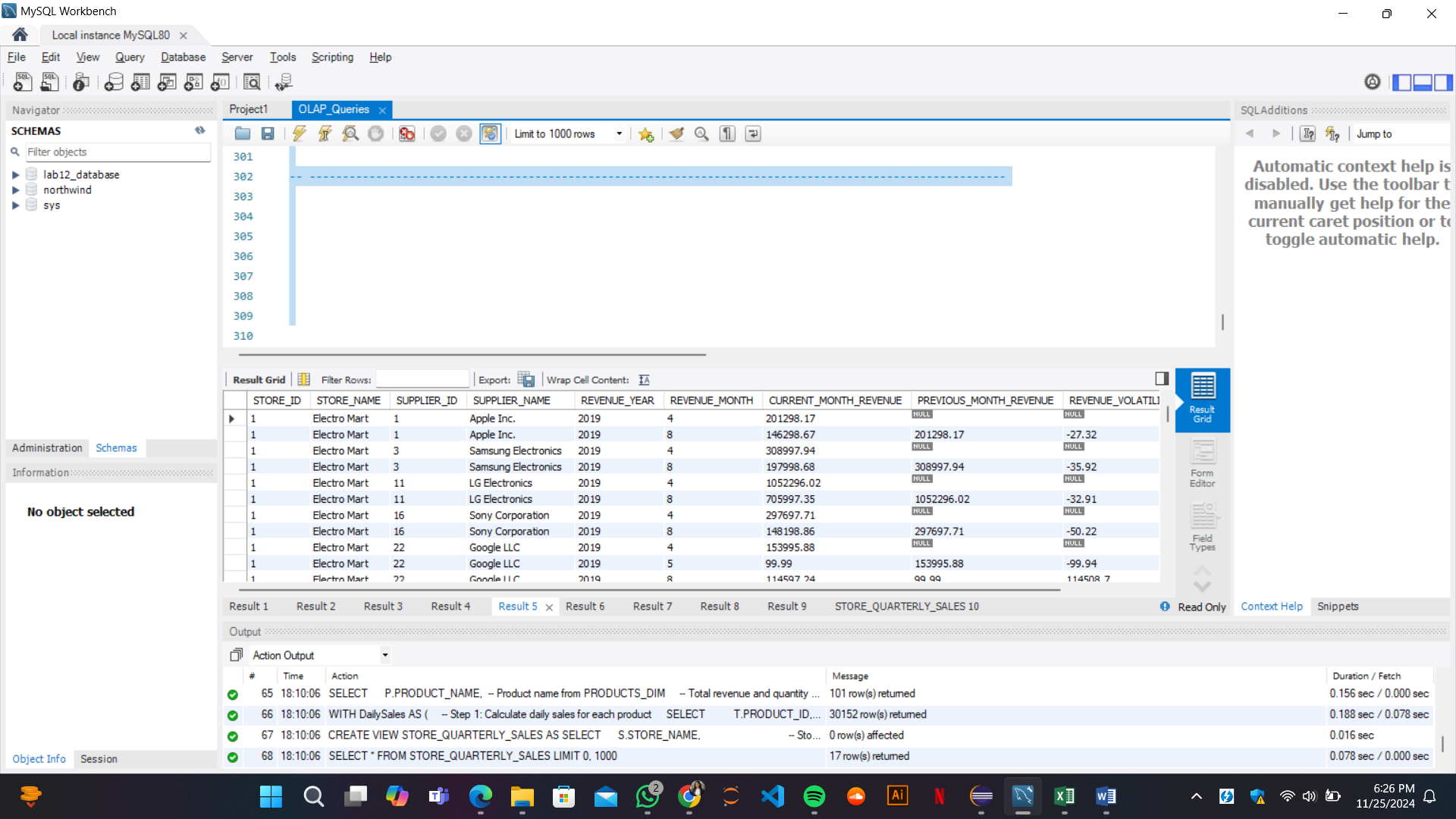
**Q3. Detailed Supplier Sales Contribution by Store and Product Name:** For each store, show the total sales contribution of each supplier broken down by product name. The output should group results by store, then supplier, and then product name under each supplier.



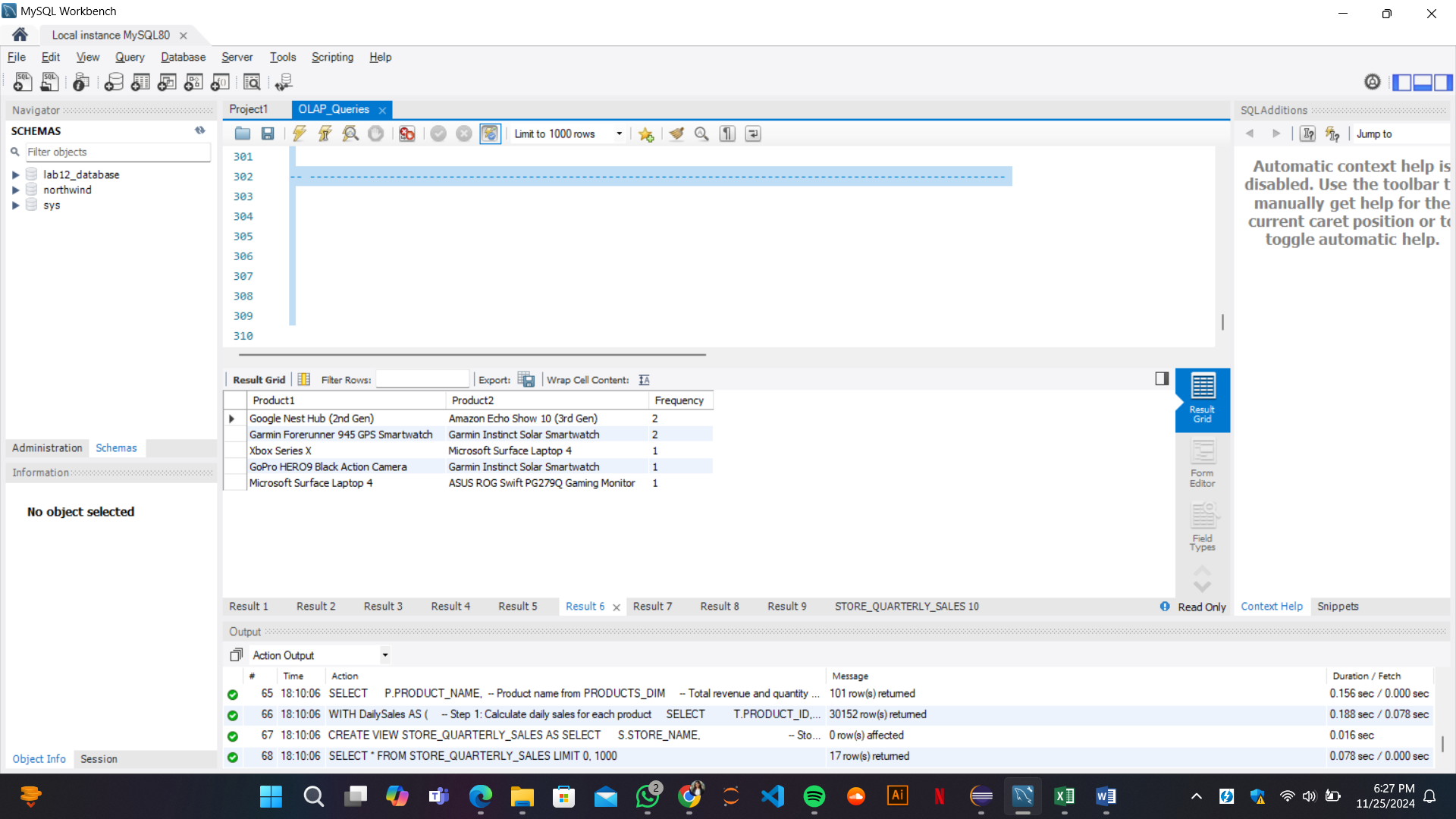
**Q4. Seasonal Analysis of Product Sales Using Dynamic Drill-Down:** Present total sales for each product, drilled down by seasonal periods (Spring, Summer, Fall, Winter). This can help understand product performance across seasonal periods.



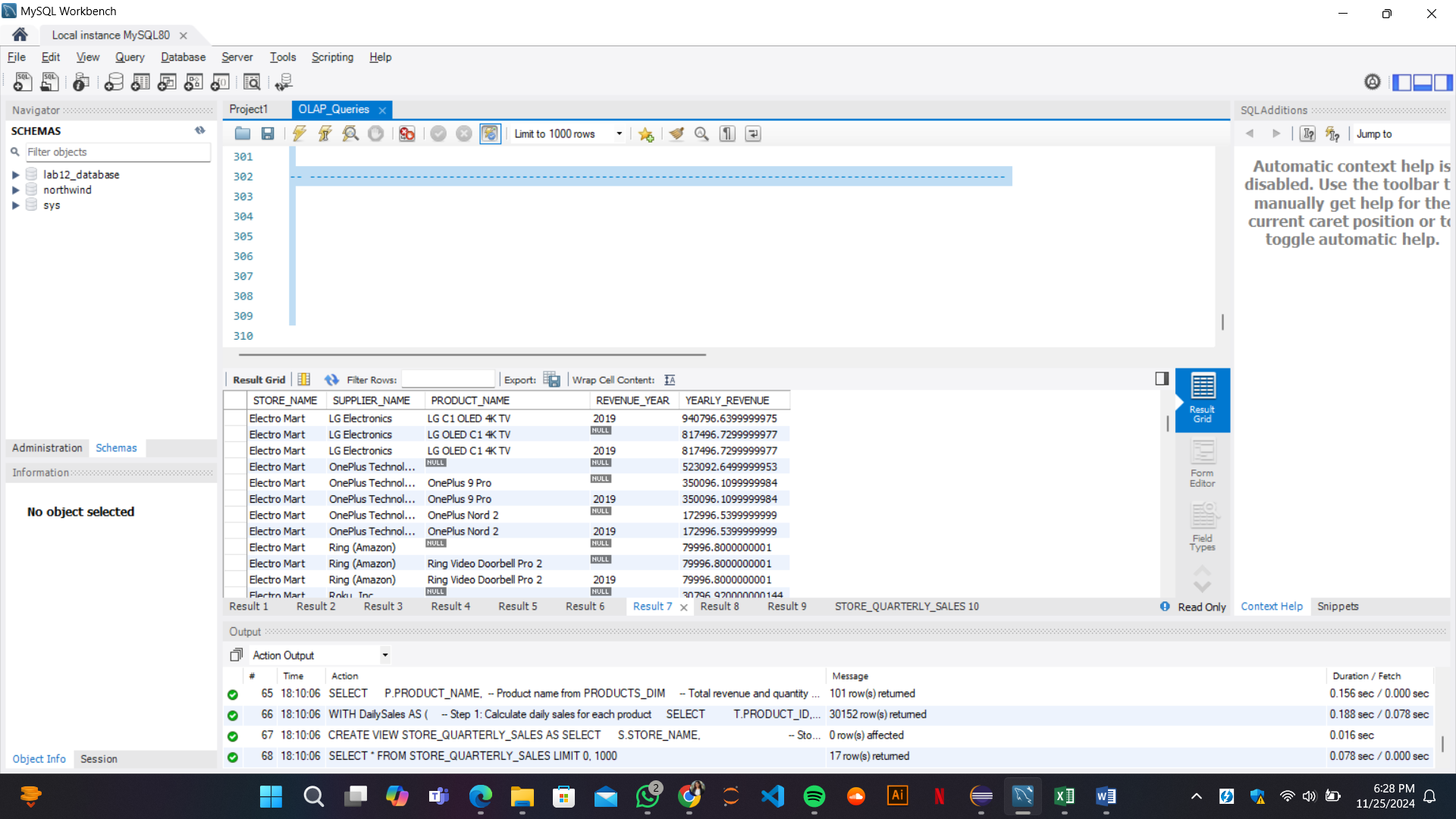
**Q5. Store-Wise and Supplier-Wise Monthly Revenue Volatility**: Calculate the month-to-month revenue volatility for each store and supplier pair. Volatility can be defined as the percentage change in revenue from one month to the next, helping identify stores or suppliers with highly fluctuating sales.



**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.



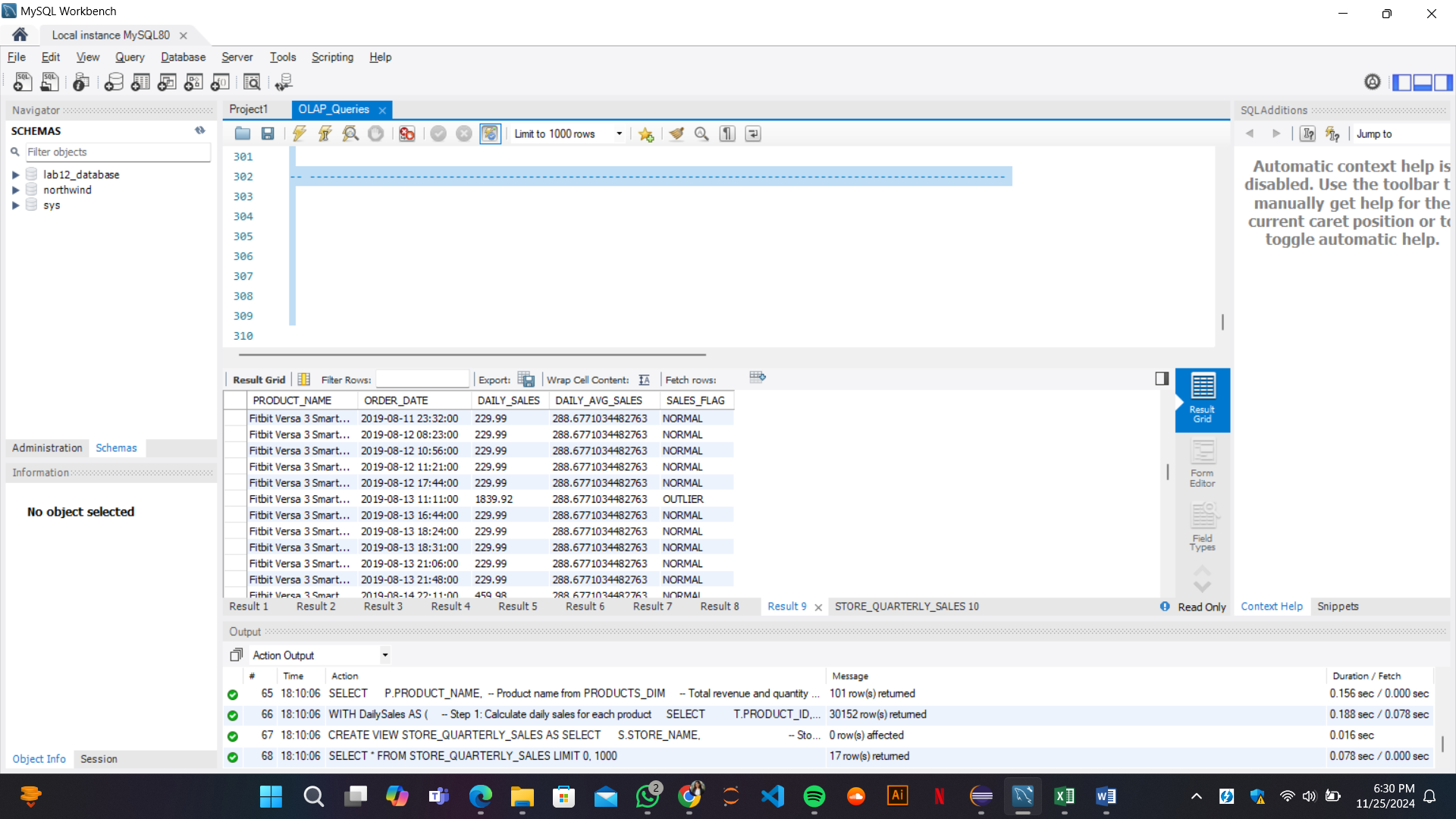
**Q7. Yearly Revenue Trends by Store, Supplier, and Product with ROLLUP:** Use the ROLLUP operation to aggregate yearly revenue data by store, supplier, and product, enabling a comprehensive overview from individual product-level details up to total revenue per store. This query should provide an overview of cumulative and hierarchical sales figures.



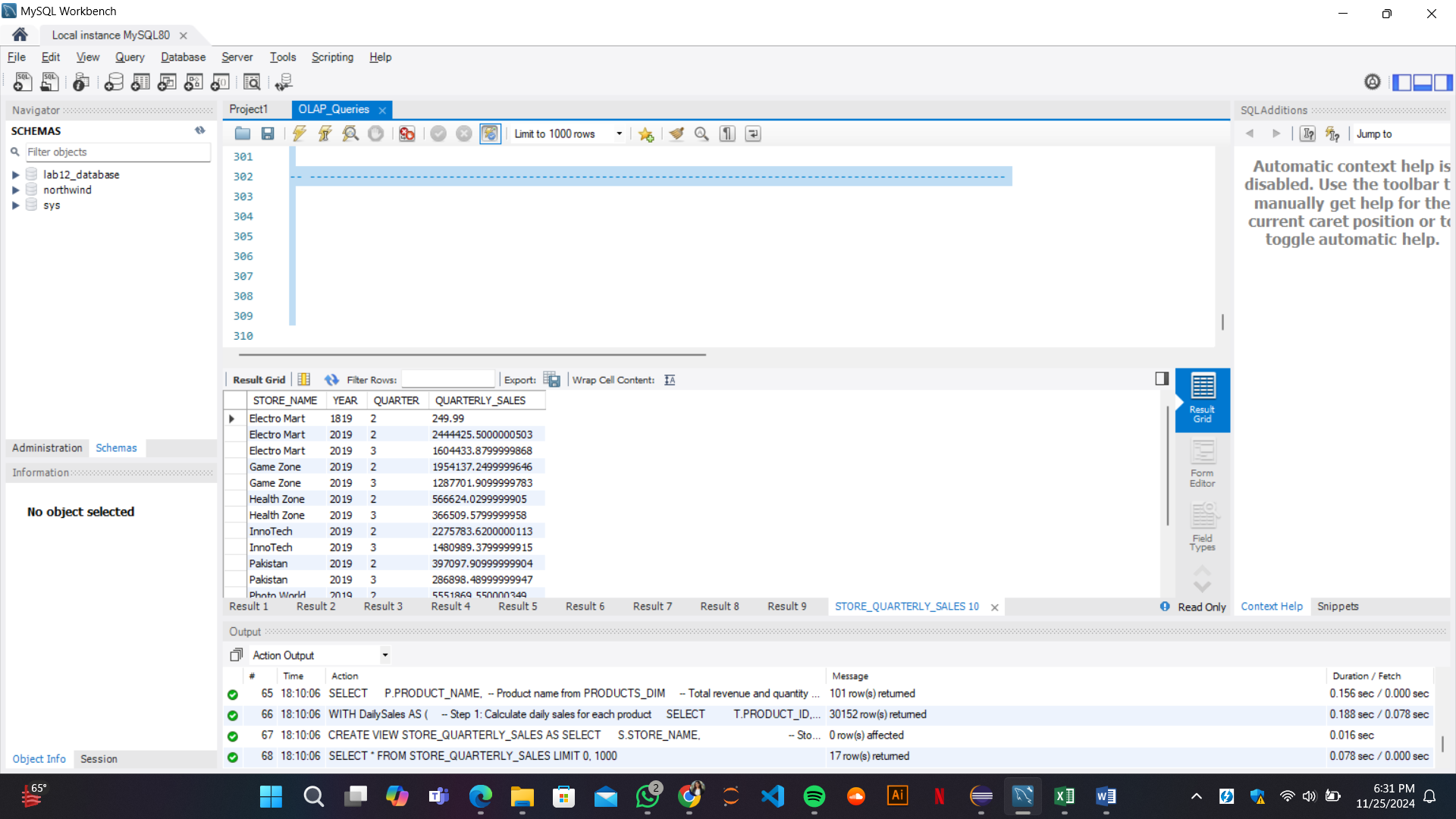
**Q8. Revenue and Volume-Based Sales Analysis for Each Product for H1 and H2:** For each product, calculate the total revenue and quantity sold in the first and second halves of the year, along with yearly totals. This split-by-time-period analysis can reveal changes in product popularity or demand over 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.



**5. Learning Outcomes from the Project**

Through this project, several valuable insights were gained:

1. **Data Warehouse Design**:

The project enhanced my understanding of how to design an efficient data warehouse schema (specifically a star schema) to support business intelligence operations. The importance of dimension tables and fact tables in structuring data for efficient analysis became evident.

1. **Efficient Data Loading**:

Implementing the MESHJOIN algorithm helped me learn how to optimize the data loading process into the data warehouse. By leveraging hashing techniques, we were able to significantly reduce the time taken for data joins, especially when dealing with large transactional datasets.

1. **OLAP Techniques**:

I gained practical experience in applying OLAP concepts such as slicing, dicing, drill-down, and materializing views. These techniques allow for deeper insights into data and provide decision-makers with the ability to analyze trends, customer behavior, and product performance across different dimensions.

1. **Challenges and Optimization**:

The project also highlighted the challenges associated with data processing at scale, such as data skew, hashing efficiency, and handling null values. These challenges have deepened my understanding of the importance of optimizing algorithms for large datasets.