**Customer Segmentation and Analysis Report**

**Introduction**

Customer segmentation is a strategic approach utilized by organizations to classify their clientele based on various attributes such as demographics, purchasing behavior, and interaction history. By implementing effective segmentation strategies, businesses can tailor their marketing efforts, improve customer satisfaction, and optimize overall performance. This report details the use of SQL to analyze customer data, aiming to segment customers, identify purchasing patterns, and gain insights into customer behavior.

**Dataset Overview**

The dataset provided for this analysis includes the following variables:

* **InvoiceNo:** The invoice number for each transaction.
* **StockCode:** The unique code for each product sold.
* **Description:** The description of each product sold.
* **Quantity:** The quantity of each product sold in each transaction.
* **InvoiceDate:** The date and time of each transaction.
* **UnitPrice:** The price of each product sold.
* **CustomerID:** The unique identifier for each customer.
* **Country:** The country where each transaction occurred.

The analysis focuses on answering both beginner and advanced queries using SQL to derive insights that will support customer segmentation and improve strategic decision-making.

**Beginner Queries**

1. **Define Meta Data**

Meta data provides information about the dataset structure, including tables, fields, and data types. In MySQL Workbench or any SQL tool, meta data can be defined and queried using the SHOW TABLES and DESCRIBE table\_name commands.

**Example:**

SHOW TABLES;

DESCRIBE customer\_data;

1. **Distribution of Order Values Across All Customers**

To understand the distribution of order values, calculate the total order value for each customer and then analyze the distribution.

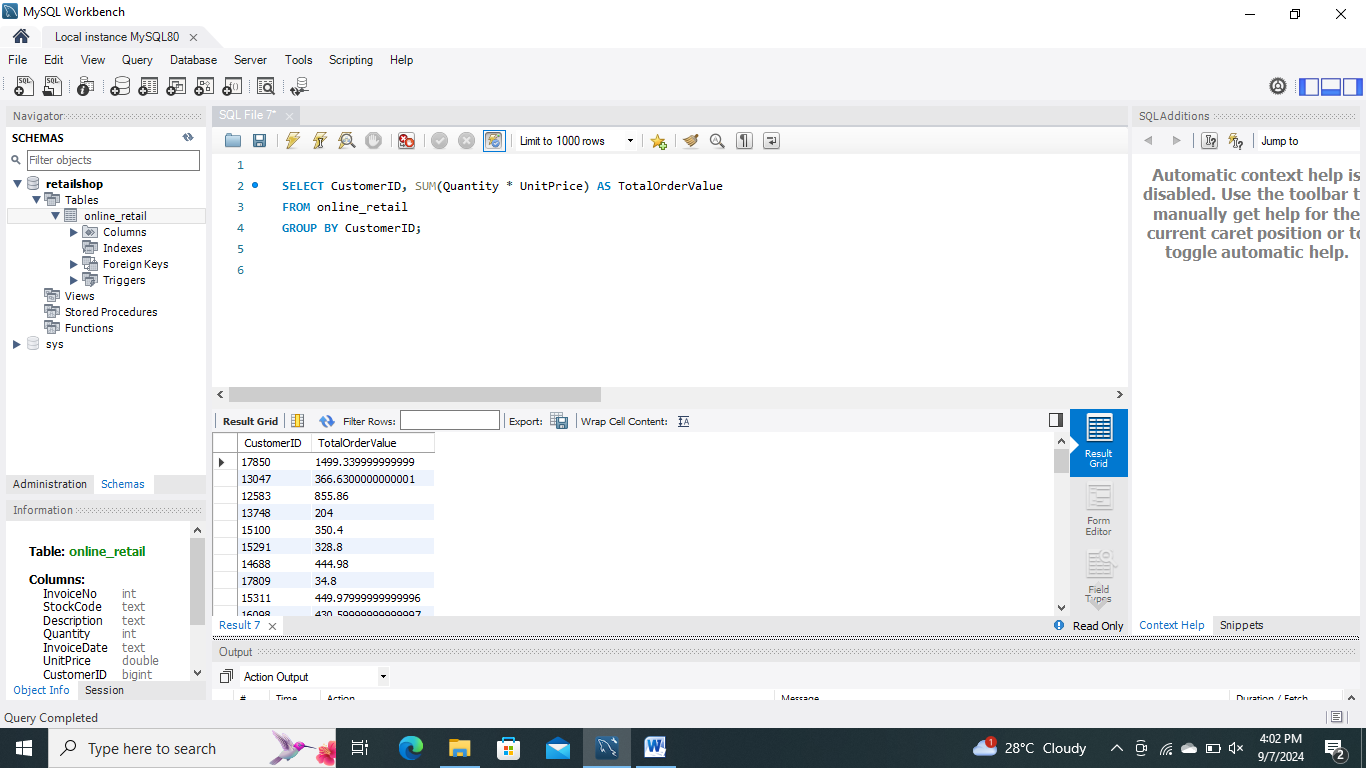
**Query:**

SELECT CustomerID, SUM(Quantity \* UnitPrice) AS TotalOrderValue

FROM online\_retail

GROUP BY CustomerID;

This query calculates the total order value for each customer by multiplying quantity and unit price and then summing it up for each CustomerID.



1. **Unique Products Purchased by Each Customer**

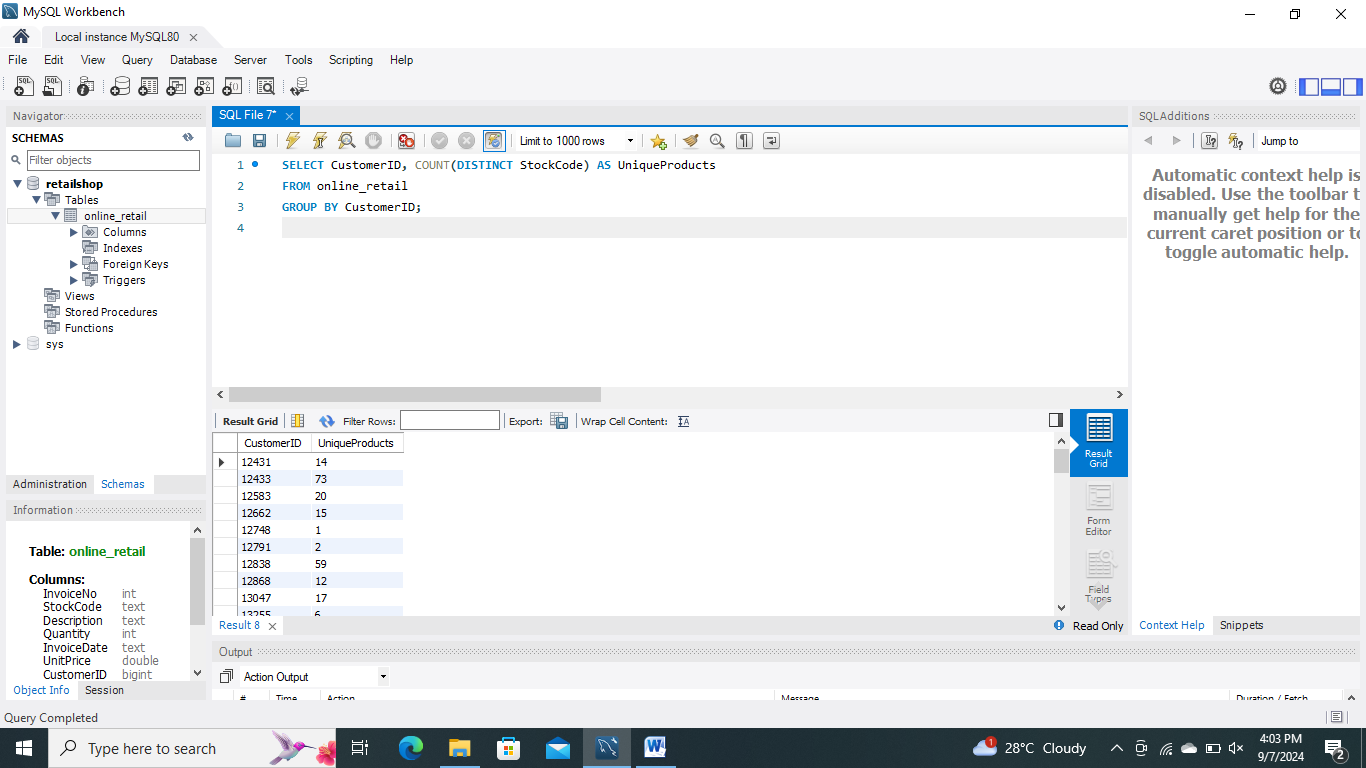
Identify the number of unique products purchased by each customer.

**Query:**

SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProducts

FROM online\_retail GROUP BY CustomerID;

This query counts the distinct StockCode values for each customer, giving the number of unique products purchased.



1. **Customers with a Single Purchase**

Determine which customers have only made one purchase.

**Query:**

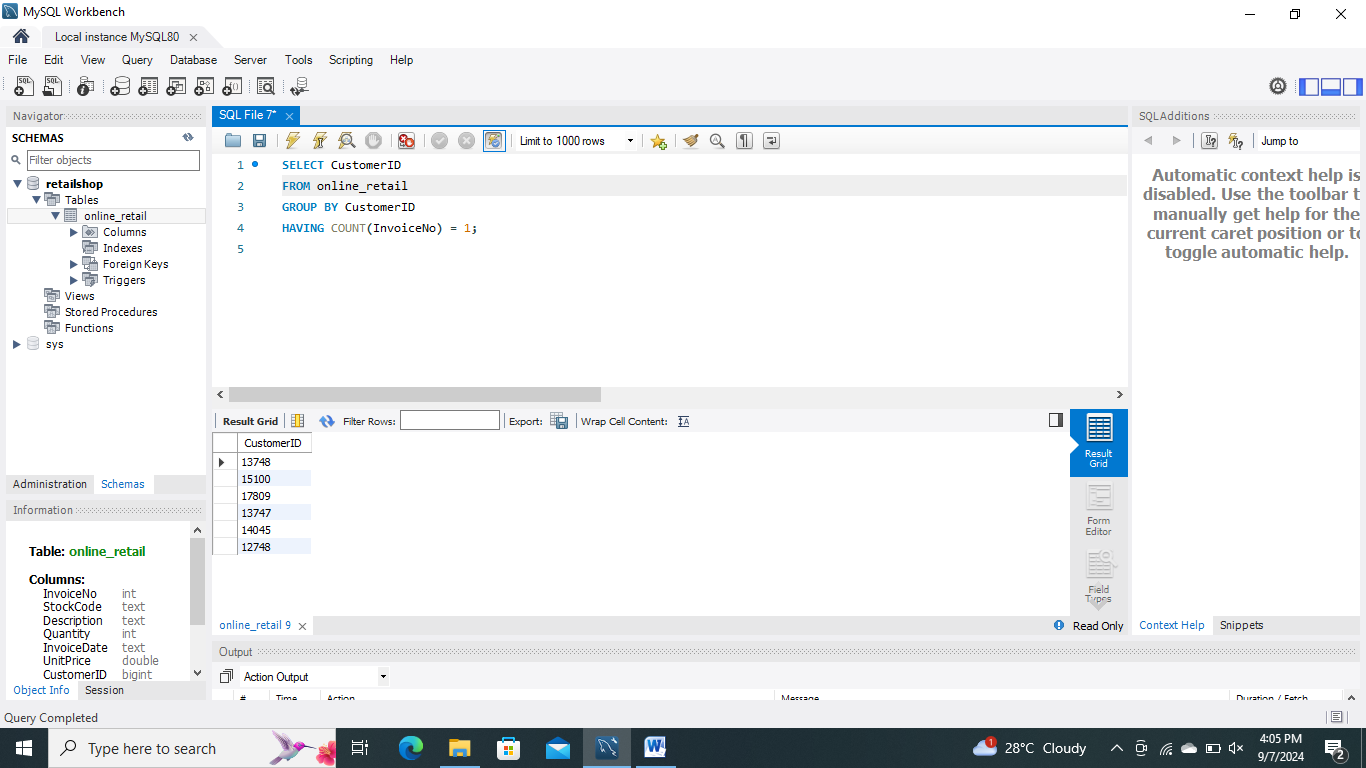
SELECT CustomerID

FROM online\_retail

GROUP BY CustomerID

HAVING COUNT(InvoiceNo) = 1;

This query groups transactions by CustomerID and filters those with only one invoice.



1. **Most Commonly Purchased Products Together**

Identify pairs of products that are frequently purchased together.

**Query:**

SELECT A.StockCode AS Product1, B.StockCode AS Product2, COUNT(\*) AS PurchaseCount

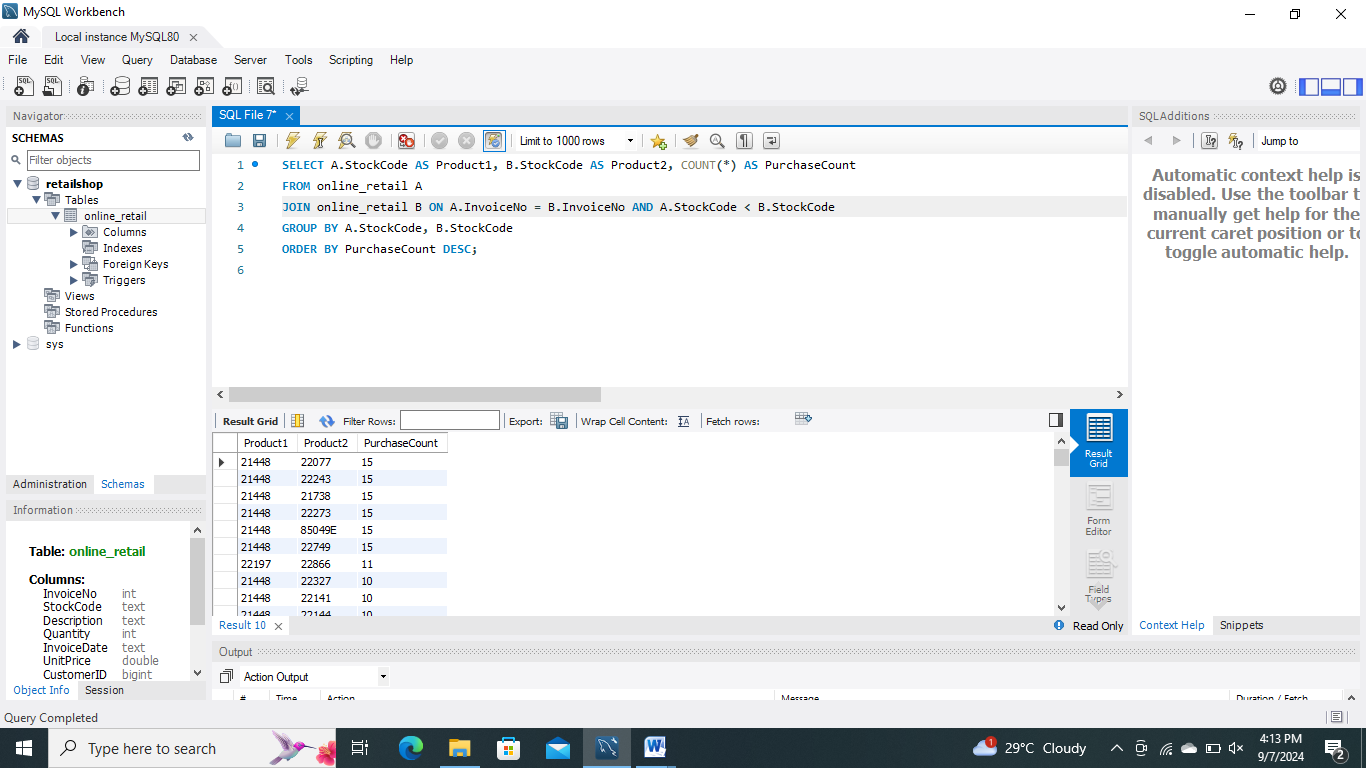
FROM online\_retail A

JOIN online\_retail B ON A.InvoiceNo = B.InvoiceNo AND A.StockCode < B.StockCode

GROUP BY A.StockCode, B.StockCode

ORDER BY PurchaseCount DESC;

This query joins the table with itself to find pairs of products purchased in the same transaction and counts their occurrences.



**Advanced Queries**

1. **Customer Segmentation by Purchase Frequency**

Segment customers based on their purchase frequency.

**Query:**

SELECT CustomerID,

CASE

WHEN COUNT(InvoiceNo) > 10 THEN 'High Frequency'

WHEN COUNT(InvoiceNo) BETWEEN 5 AND 10 THEN 'Medium Frequency'

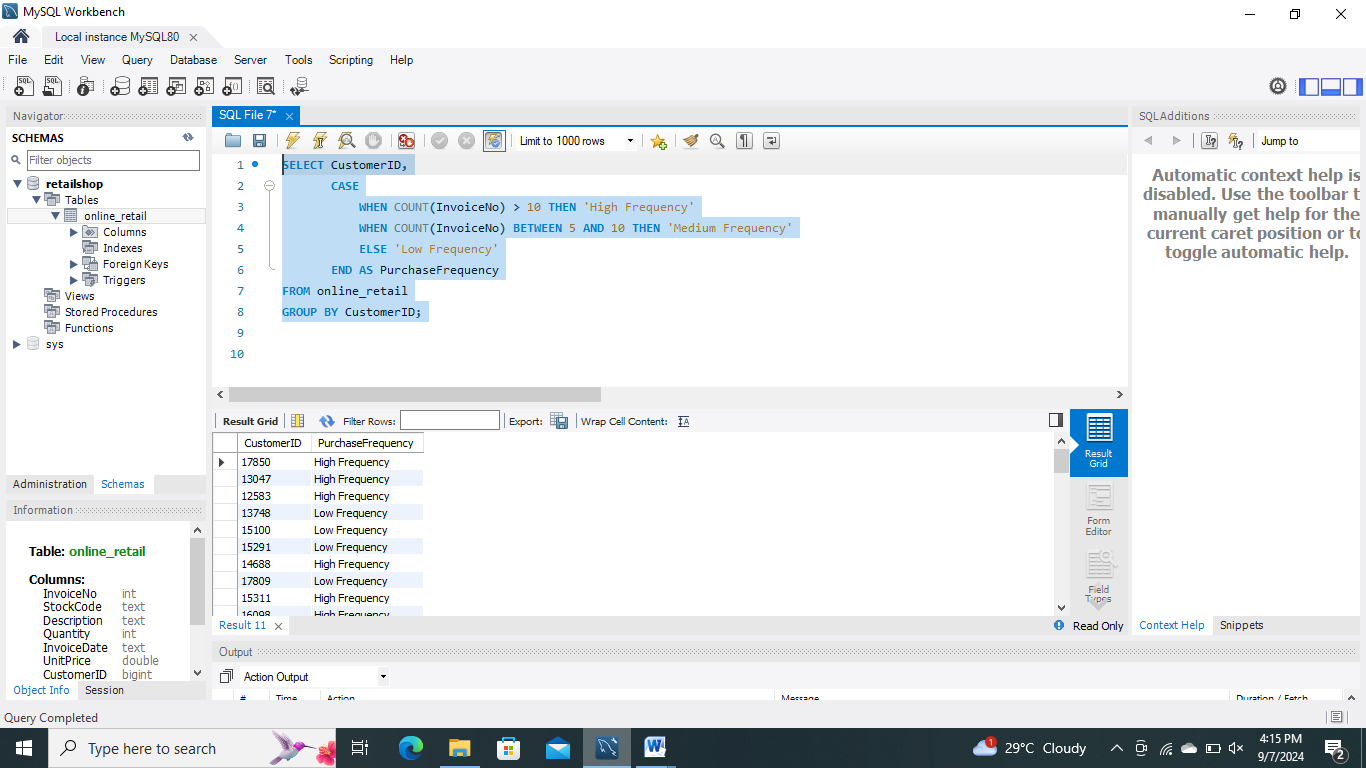
ELSE 'Low Frequency'

END AS PurchaseFrequency

FROM online\_retail

GROUP BY CustomerID;

This query segments customers into high, medium, and low frequency based on the number of transactions.



1. **Average Order Value by Country**

Calculate the average order value for each country.

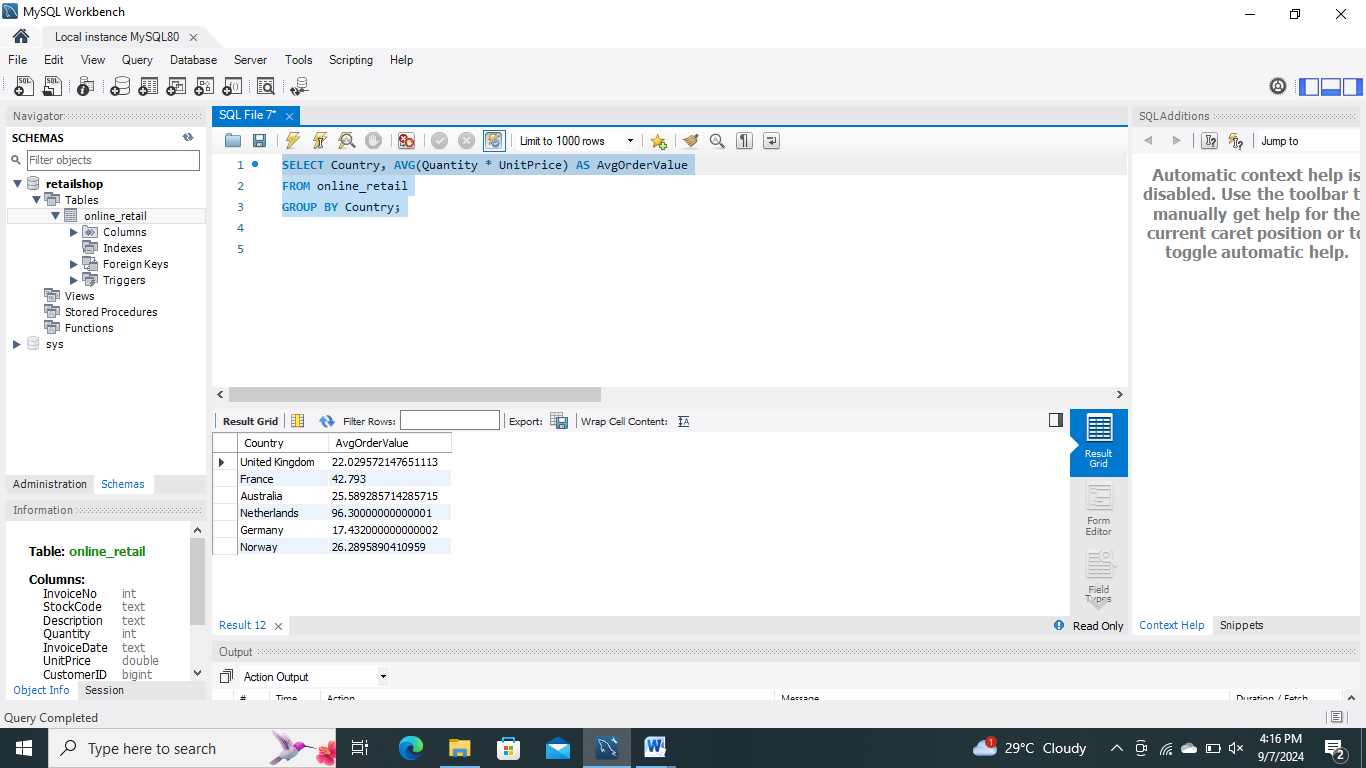
**Query:**

SELECT Country, AVG(Quantity \* UnitPrice) AS AvgOrderValue

FROM online\_retail

GROUP BY Country;

This query calculates the average value of orders for each country by grouping transactions and averaging the total order value.



1. **Customer Churn Analysis**

Identify customers who haven't made a purchase in the last 6 months.

**Query:**

SELECT CustomerID

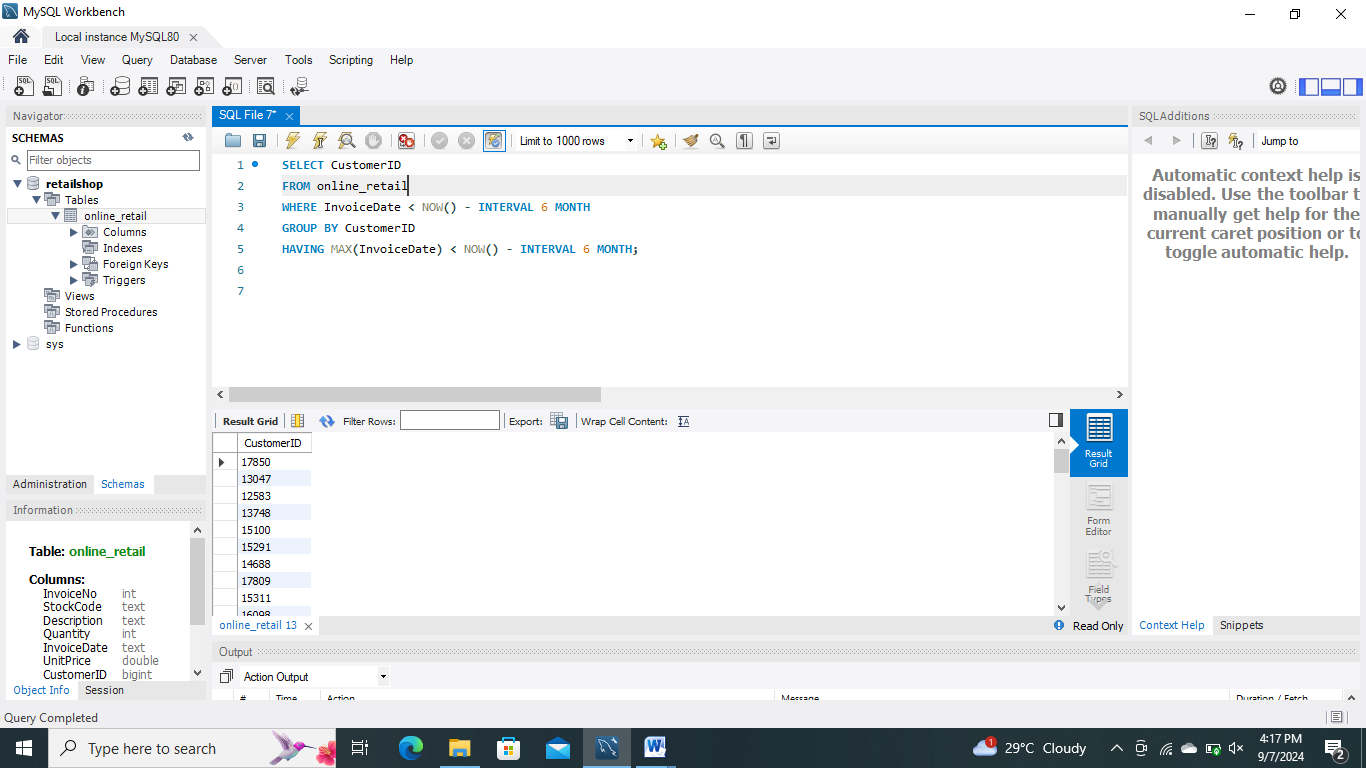
FROM online\_retail

WHERE InvoiceDate < NOW() - INTERVAL 6 MONTH

GROUP BY CustomerID

HAVING MAX(InvoiceDate) < NOW() - INTERVAL 6 MONTH;

This query finds customers who have not made any purchases in the past 6 months.



1. **Product Affinity Analysis**

Determine which products are often purchased together by calculating correlations between product purchases.

**Query:**

SELECT A.StockCode AS Product1, B.StockCode AS Product2, COUNT(\*) AS PurchaseCount

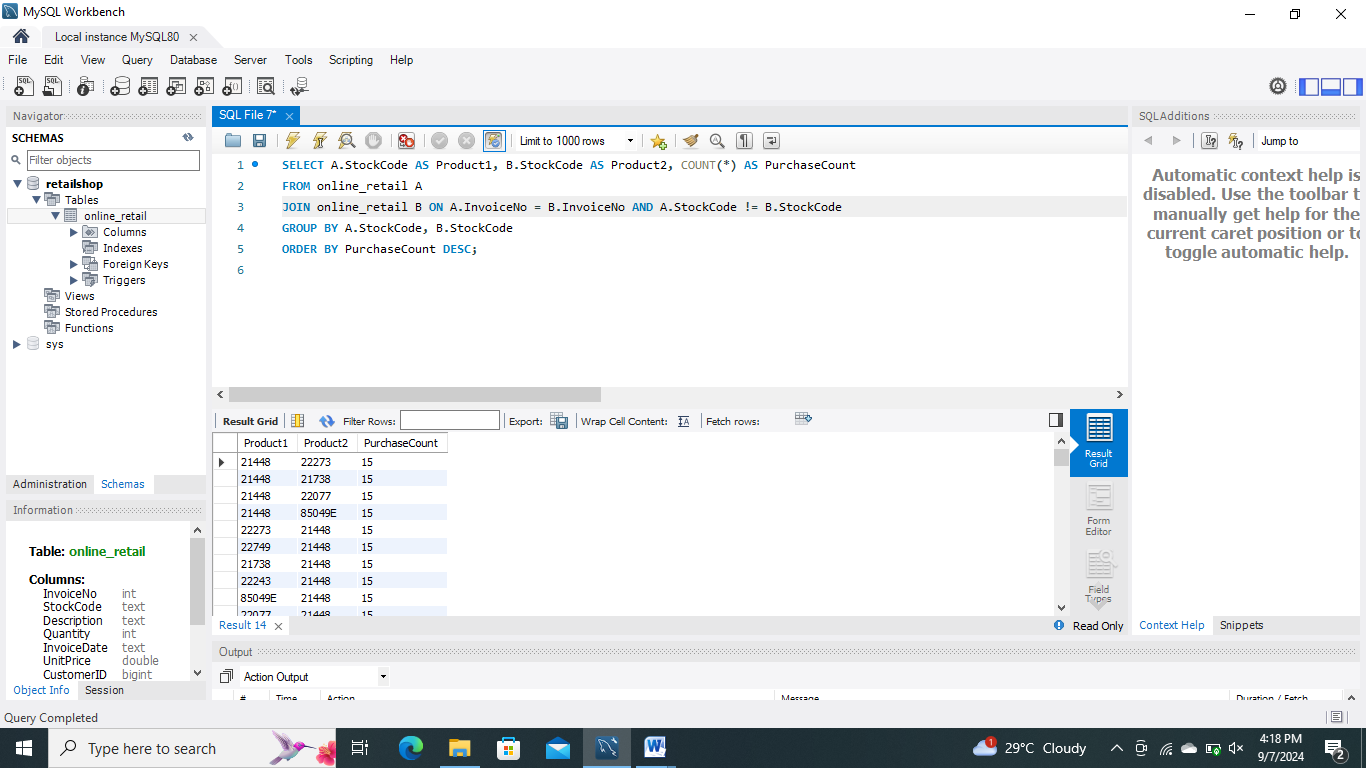
FROM online\_retail A

JOIN online\_retail B ON A.InvoiceNo = B.InvoiceNo AND A.StockCode != B.StockCode

GROUP BY A.StockCode, B.StockCode

ORDER BY PurchaseCount DESC;

This query identifies pairs of different products frequently purchased together by joining the transactions table with itself.



5. **Time-based Analysis**

Explore sales trends over time, such as monthly sales patterns.

**Query:**

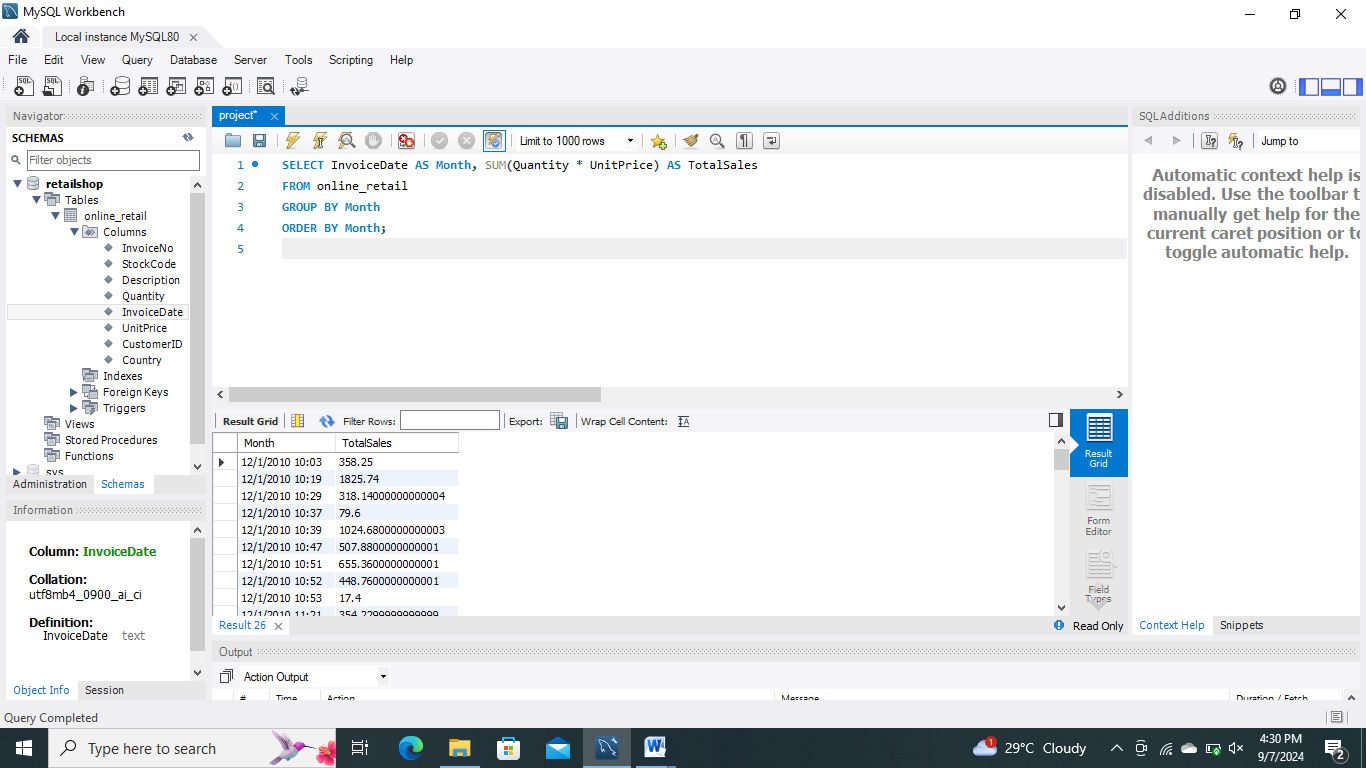
SELECT InvoiceDate AS Month, SUM(Quantity \* UnitPrice) AS TotalSales

FROM online\_retail

GROUP BY Month

ORDER BY Month;

This query aggregates total sales by month, providing insights into monthly sales trends.



**Conclusion**

The SQL queries performed in this analysis offer valuable insights into customer behavior and purchasing patterns. By understanding the distribution of order values, identifying unique product purchases, and analyzing product affinities, businesses can segment their customer base more effectively and tailor their strategies accordingly. The advanced queries further provide deeper insights into customer segmentation, geographic performance, and time-based trends, enabling more informed decision-making and targeted marketing strategies.

**GitHub Profile Link:** https://github.com/sania-arif/Data-mining-project