ONLINE RETAIL SEGMENTATION

DATA MINING FINAL PROJECT

Presented by
AFTAB NAFEES
GitHub Username
aftab1038
Repository Link
https://shorturl.at/5tZOf

Table of Contents

Introduction

The objective of this project is to perform customer segmentation using data mining techniques, specifically through SQL. Customer segmentation allows businesses to categorize clients based on various factors such as demographics and purchasing patterns. This segmentation is crucial for tailoring marketing strategies, improving customer satisfaction, and boosting overall business performance.

The dataset used in this project contains transaction-level data from an online retail store, with key variables including:

- **InvoiceNo:** Unique invoice number for each transaction.
- **StockCode:** Unique product code.
- **Description:** Product description.
- **Quantity:** Quantity of the product sold.
- **InvoiceDate:** Date and time of the transaction.
- **UnitPrice:** Price per unit of the product.
- **CustomerID:** Unique identifier for each customer.
- **Country:** Country where the transaction occurred.

Number of Rows: This dataset contains **581587** rows

Beginner Queries

Metadata Definition

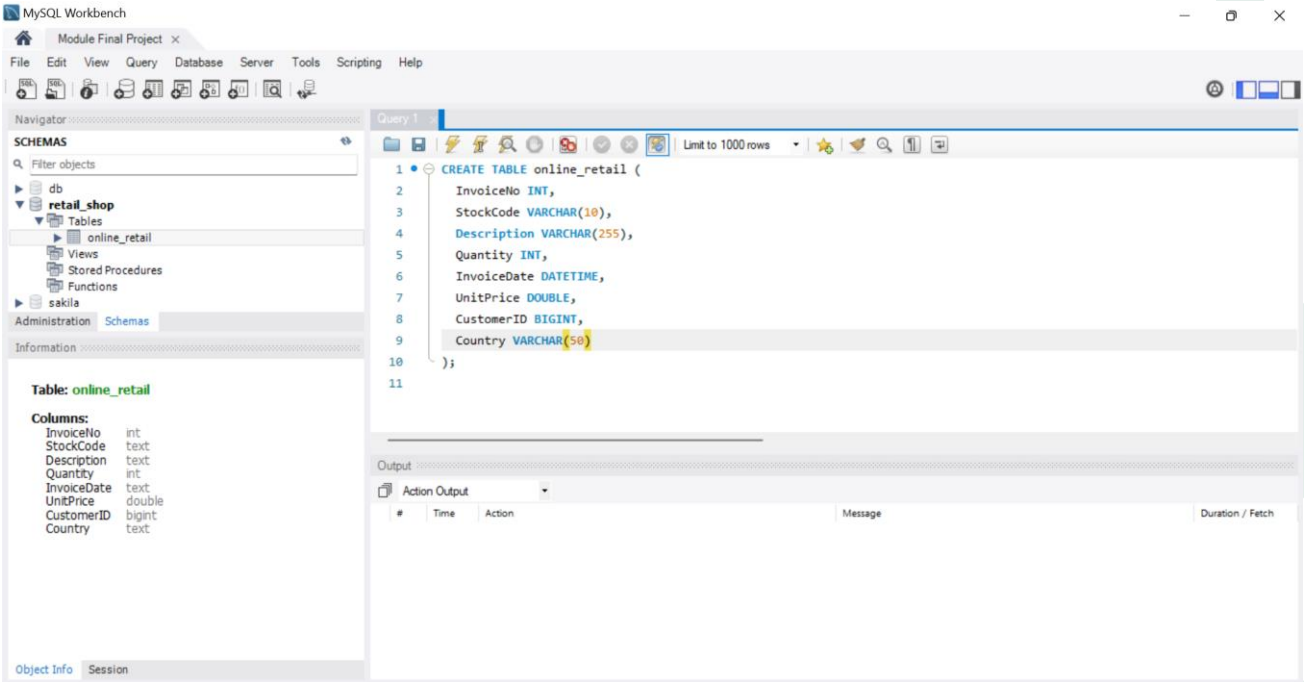
Query:

```
CREATE TABLE online_retail (
InvoiceNo VARCHAR(10),
StockCode VARCHAR(10),
Description VARCHAR(255),
Quantity INT,
InvoiceDate DATETIME,
UnitPrice DECIMAL(10, 2),
CustomerID VARCHAR(10),
Country VARCHAR(50)
);
```

Explanation:

This query creates a table structure in SQL, defining the necessary fields to store and manipulate the dataset.

Output:



Distribution of Order Values

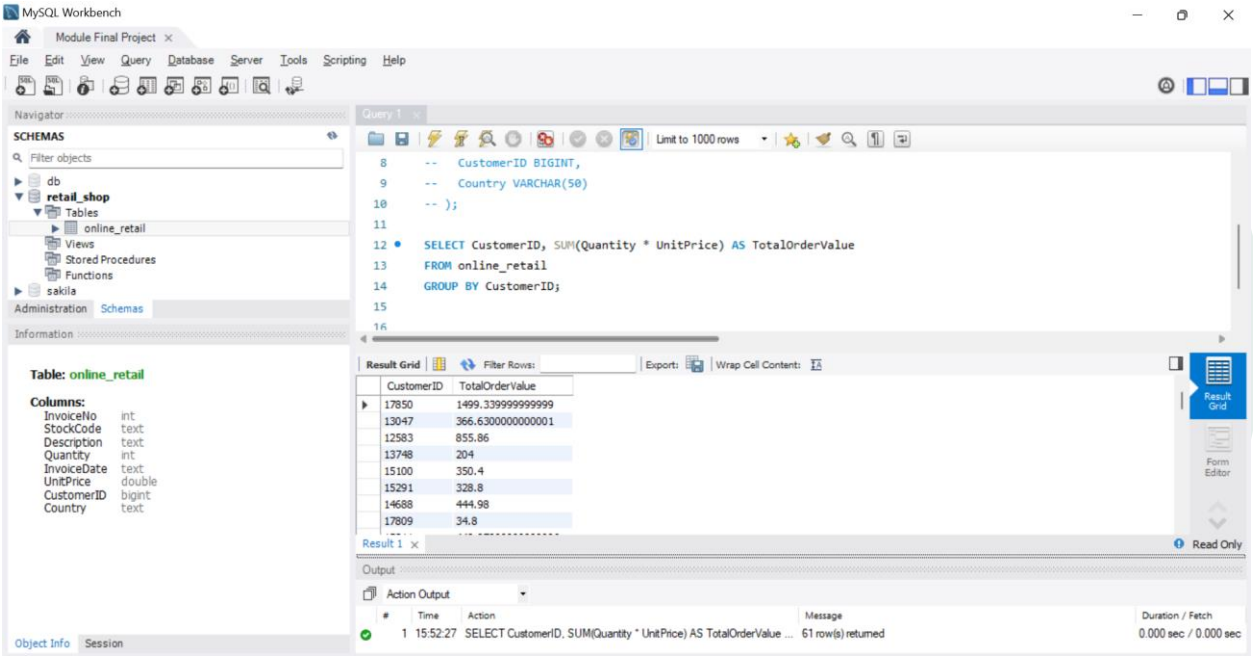
Query:

```
SELECT CustomerID, SUM(Quantity * UnitPrice)
AS TotalOrderValue
FROM online_retail
GROUP BY CustomerID;
```

Explanation:

This query calculates the total value of orders placed by each customer. It is crucial for understanding the spending habits of different customer segments.

Output:



Unique Products Purchased

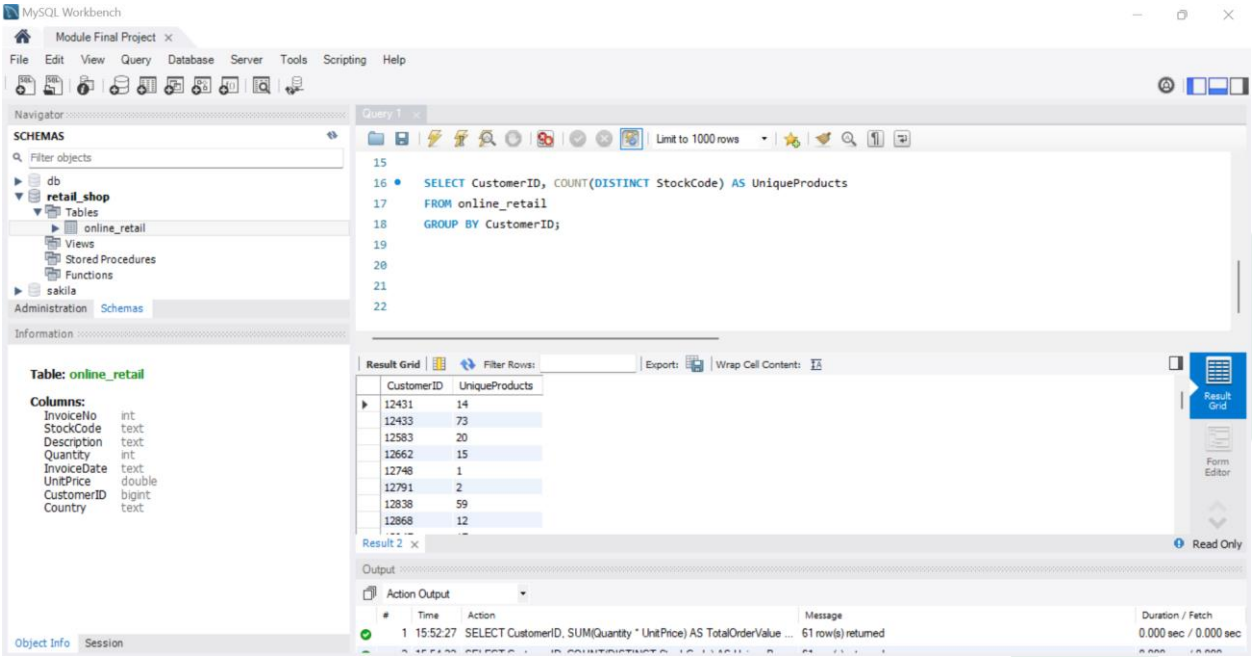
Query:

```
SELECT CustomerID, COUNT(DISTINCT StockCode)
As UniqueProducts
FROM retailGROUP
BY CustomerID;
```

Explanation:

This query determines how many unique products each customer has purchased, offering insights into customer diversity and product reach.

Output:



Single Purchase Customers

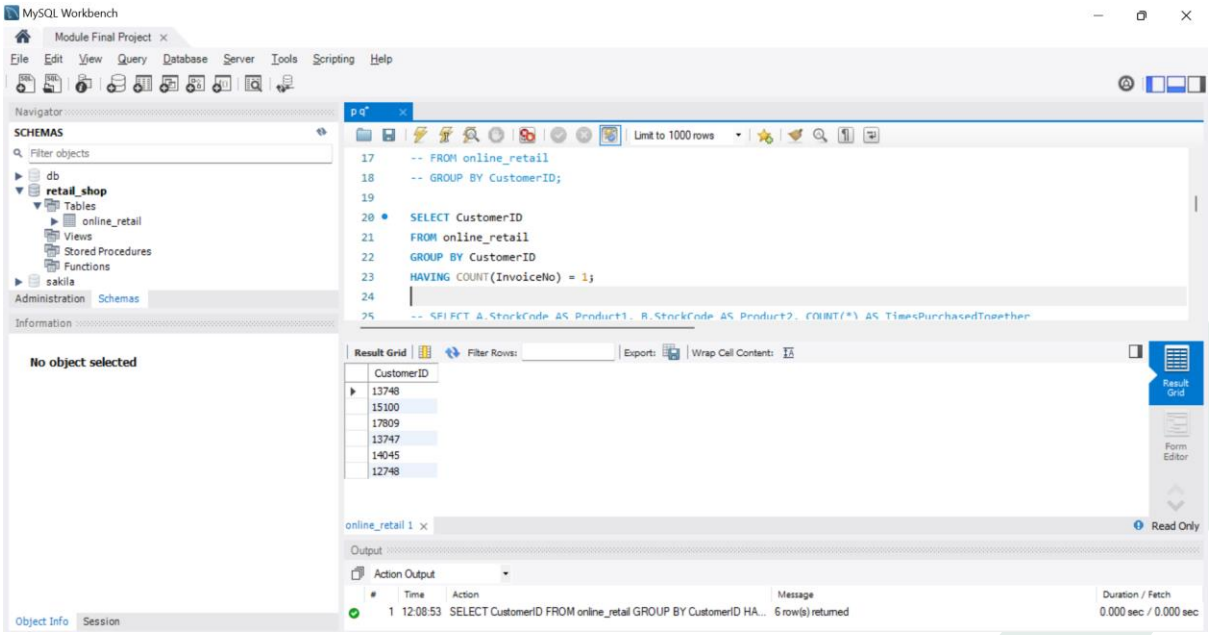
Query:

```
SELECT CustomerID
FROM online_retail
GROUP BY CustomerID
HAVING COUNT(InvoiceNo) = 1;
```

Explanation:

Identifying customers who have made only a single purchase can help in designing targeted retention strategies.

Output:



Commonly Purchased Products Together

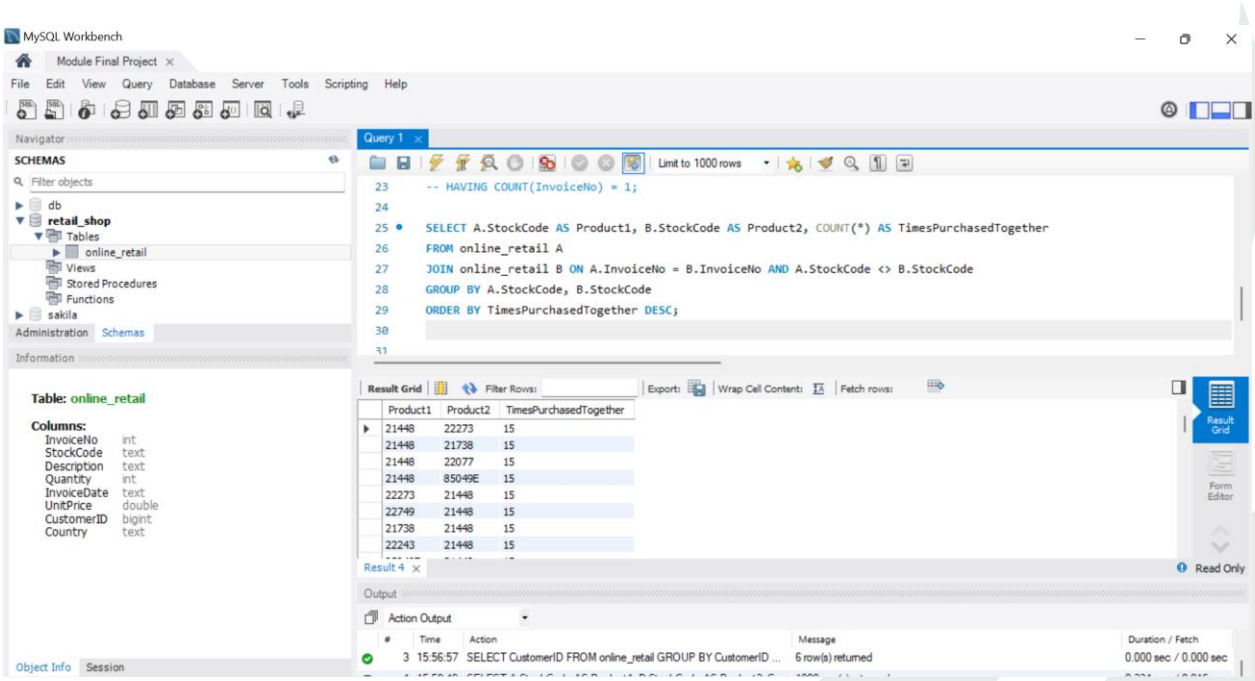
Query:

```
SELECT A.StockCode AS Product1, B.StockCode AS Product2,
COUNT(*)
AS TimesPurchasedTogether
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 TimesPurchasedTogether DESC;
```

Explanation:

This query finds pairs of products that are often purchased together by customers. It helps in identifying product bundling opportunities or understanding customer buying behavior.

Output:



Advance Queries

Customer Segmentation by Purchase Frequency

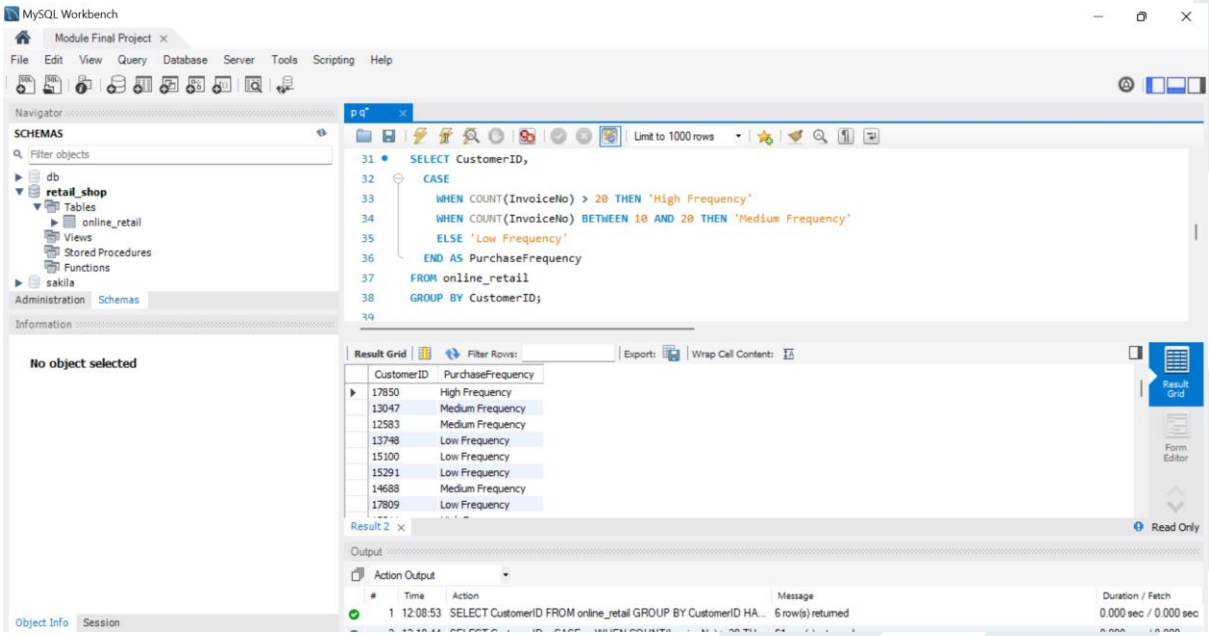
Query:

```
SELECT CustomerID,
  CASE
    WHEN COUNT(InvoiceNo) > 20 THEN 'High Frequency'
    WHEN COUNT(InvoiceNo) BETWEEN 10 AND 20 THEN 'Medium
Frequency'
    ELSE 'Low Frequency'
  END AS PurchaseFrequency
FROM online_retail
GROUP BY CustomerID;
```

Explanation:

Customers are segmented based on their purchase frequency, helping identify loyal customers and those who may need more attention.

Output:



Average Order Value by Country

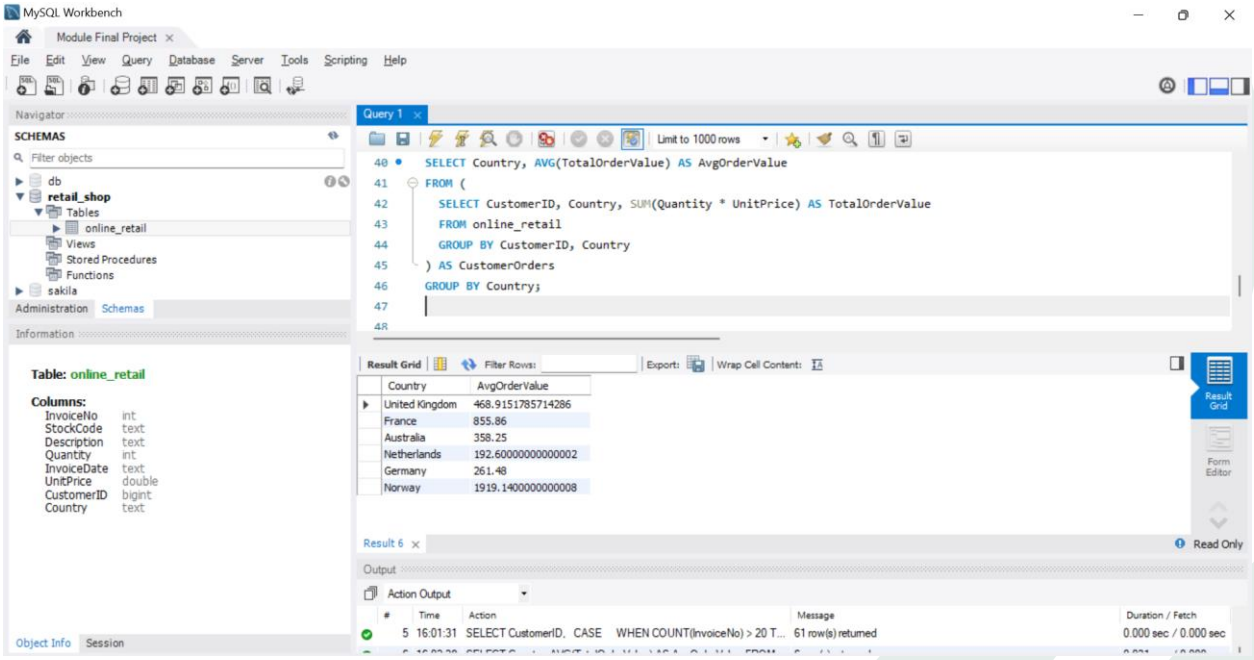
Query:

```
SELECT Country, AVG(TotalOrderValue) AS AvgOrderValue
FROM (
  SELECT CustomerID, Country, SUM(Quantity * UnitPrice)
AS TotalOrderValue
  FROM online_retail
  GROUP BY CustomerID, Country
) AS CustomerOrders
GROUP BY Country;
```

Explanation:

This query calculates the average order value by country, helping to identify regions with the highest spending customers.

Output:



Customer Churn Analysis

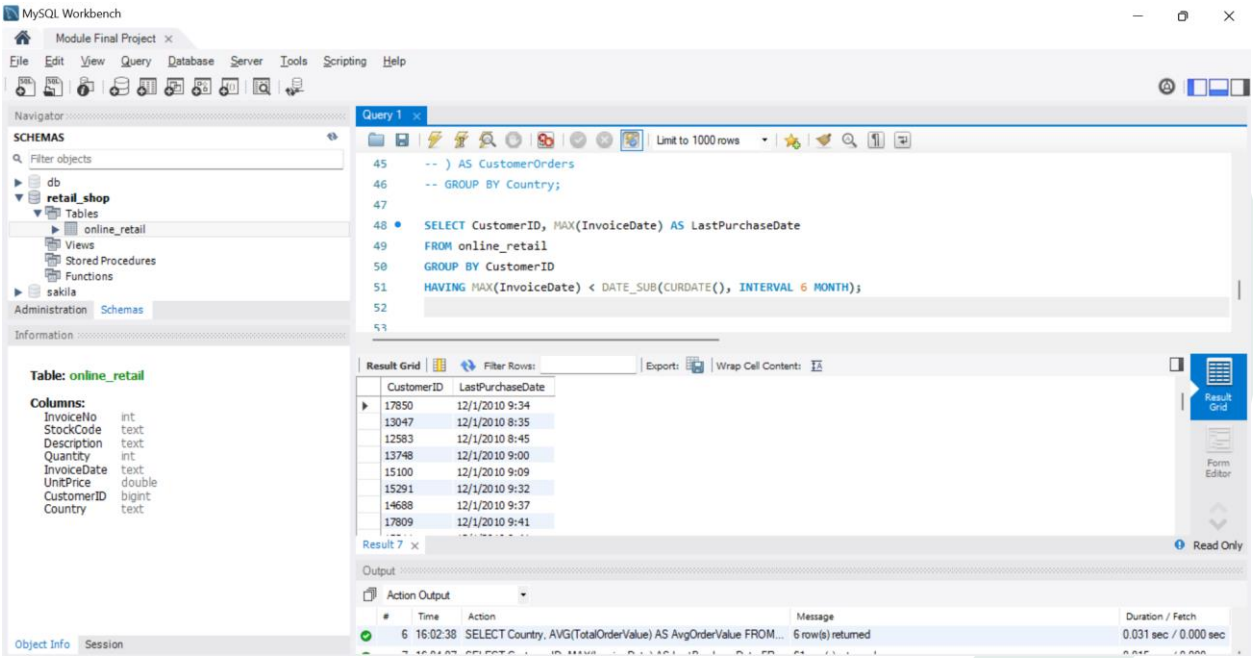
Query:

```
SELECT CustomerID, MAX(InvoiceDate) AS LastPurchaseDate
FROM online_retail
GROUP BY CustomerID
HAVING MAX(InvoiceDate) < DATE_SUB(CURDATE(), INTERVAL 6
MONTH);
```

Explanation:

Customers who haven't made a purchase in the last six months are identified, providing a basis for churn analysis.

Output:



Product Affinity Analysis

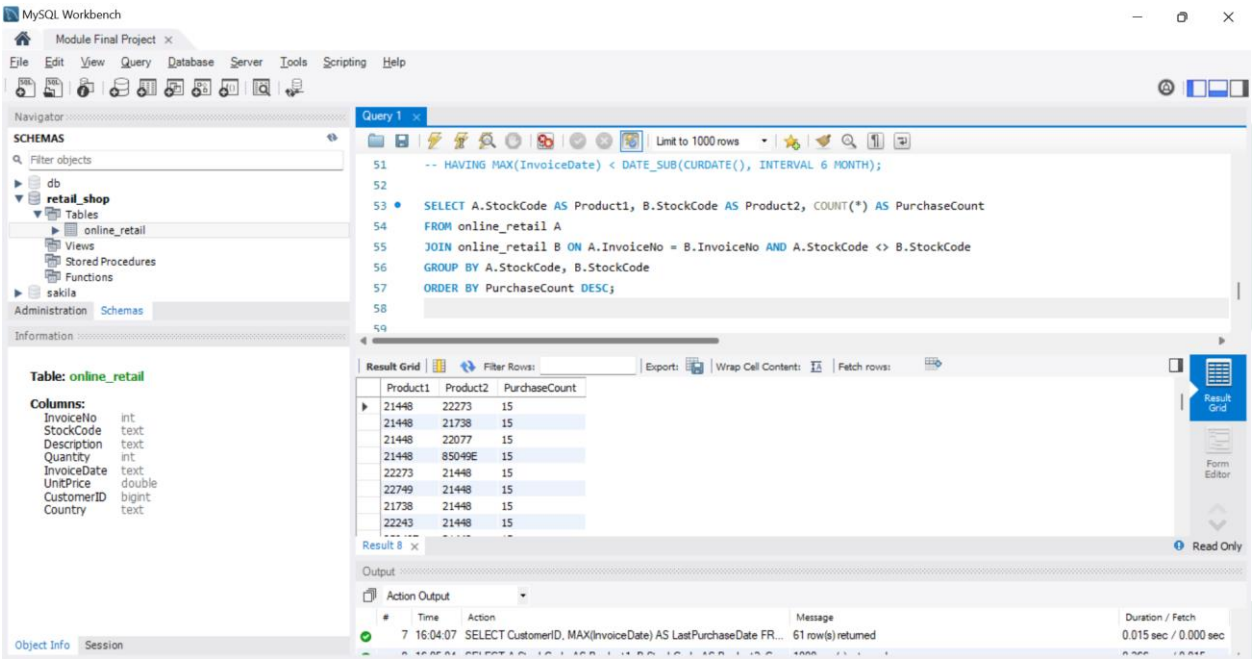
Query:

```
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;
```

Explanation:

This analysis identifies products frequently purchased together, useful for cross-selling strategies.

Output:



Time-based Analysis

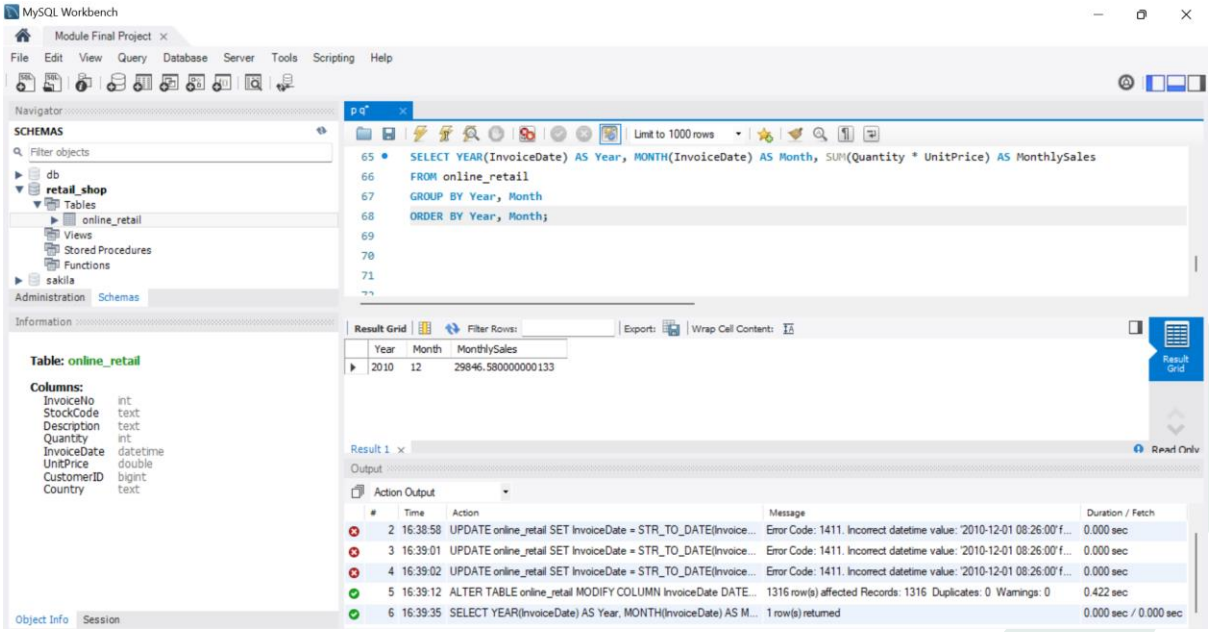
Query:

```
SELECT YEAR(InvoiceDate) AS Year, MONTH(InvoiceDate) AS
Month, SUM(Quantity * UnitPrice) AS MonthlySales
FROM online_retail
GROUP BY Year, Month
ORDER BY Year, Month;
```

Explanation:

Monthly sales trends are explored to understand seasonal or temporal patterns in customer behavior.

Output:



Conclusion

The SQL queries executed in this project provided valuable insights into customer segmentation, purchase behaviors, and potential strategies for improving customer engagement and retention. By leveraging these insights, businesses can better tailor their marketing efforts and improve overall business performance.