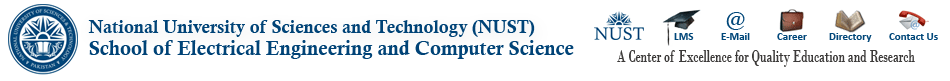
****

# High Impact Skills Development Program

# in Artificial Intelligence, Data Science, and Blockchain

***NAME: SHABBIR HUSSAIN***

***REG NO: SK-23-039***

***Module: Data Mining***

***SUBMITTED TO: Dr.Bilal Ali***

***SECTION: “A”***

***Email address : 0786shabbirhussain@gmail.com***

***github profile link :*** [***https://github.com/101SkShabbirHussainAryanKhaN?tab=repositories***](https://github.com/101SkShabbirHussainAryanKhaN?tab=repositories)

***• Define meta data in mysql workbench***

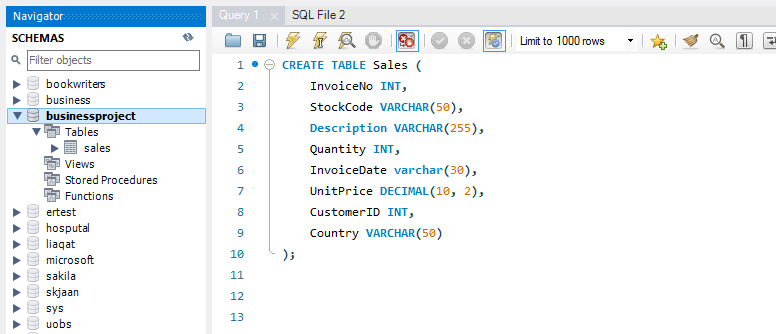
In MySQL Workbench, metadata typically refers to information about our database objects, such as tables, columns, indexes, constraints, and more. It provides information about the structure and properties of your database.

We can access metadata in MySQL Workbench in various ways:

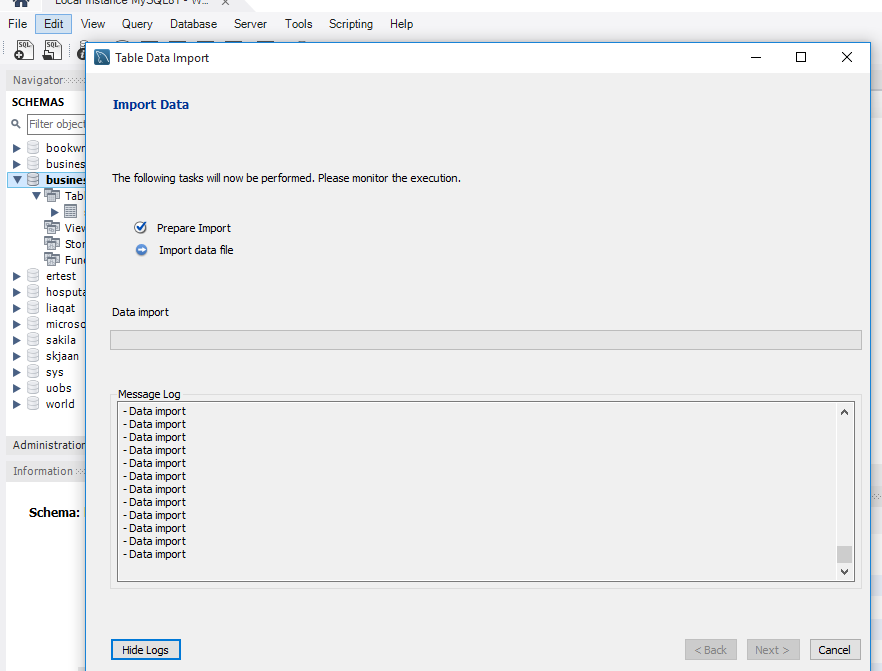
1. ***Object Browser:*** The Object Browser panel in MySQL Workbench displays the hierarchy of your database objects, allowing you to browse through tables, views, procedures, and other database elements. Right-clicking on an object and selecting "Properties" or "Alter" will show you metadata and allow you to modify it.
2. ***Schema Inspector****:* MySQL Workbench includes a "Schema Inspector" tool that provides detailed metadata about database objects. We can access it by going to the "Database" menu and selecting "Schema Inspector." This tool allows you to inspect the properties of tables, columns, indexes, constraints, and other database elements.
3. ***SQL Queries:*** We can query the database to retrieve metadata information using SQL statements. For example, you can use SQL queries like DESCRIBE, SHOW COLUMNS, SHOW TABLES, and SHOW INDEX to get information about the structure of tables, columns, indexes, and more.

***We created a table named as “online\_retail” which have following attributes,***

InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country.



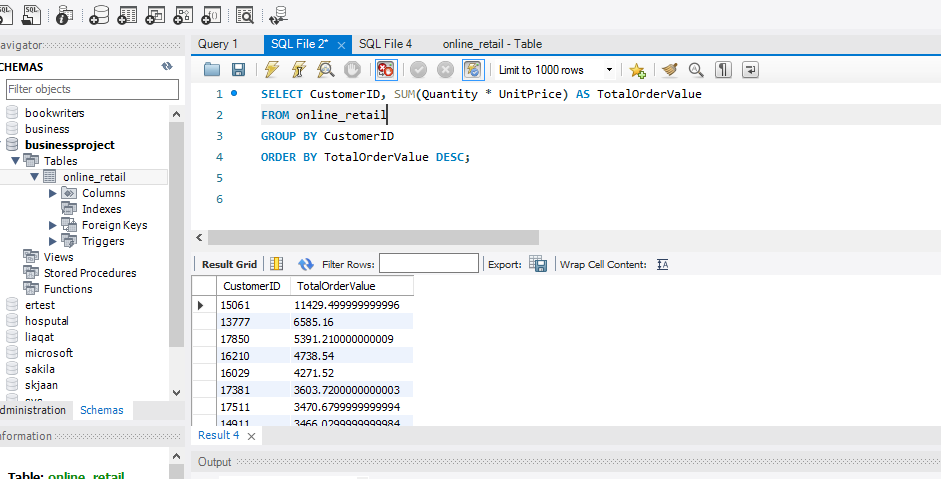
***Now we are loading data from CSV file.***



***What is the distribution of order values across all customers in the dataset?***

To calculate the distribution of order values across all customers in the dataset, you can multiply the "Quantity" by the "UnitPrice" for each item in each order and then sum these values for each order.

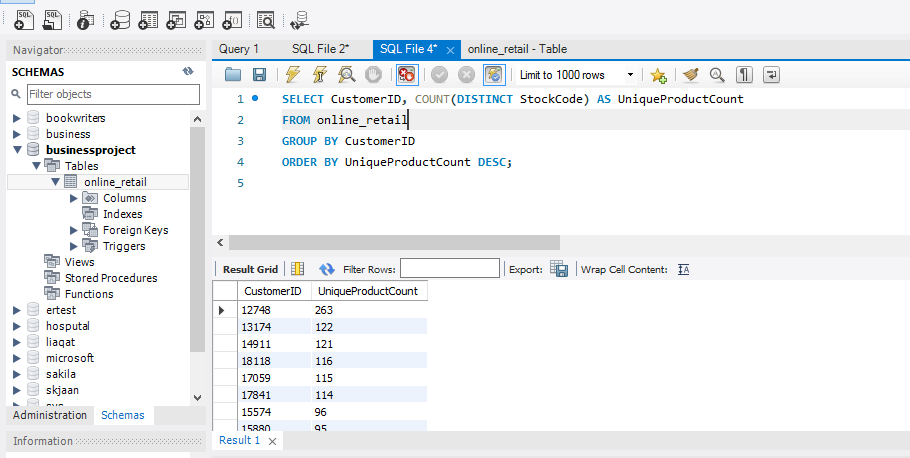
This query calculates the total order value for each invoice by summing the product of "Quantity" and "UnitPrice" for each item in the invoice. The result will give you a list of invoice numbers and their corresponding order values.



***How many unique products has each customer purchased?***

To determine how many unique products each customer has purchased, We can use a SQL query that counts the distinct products (e.g., distinct StockCodes or Product IDs) for each customer.

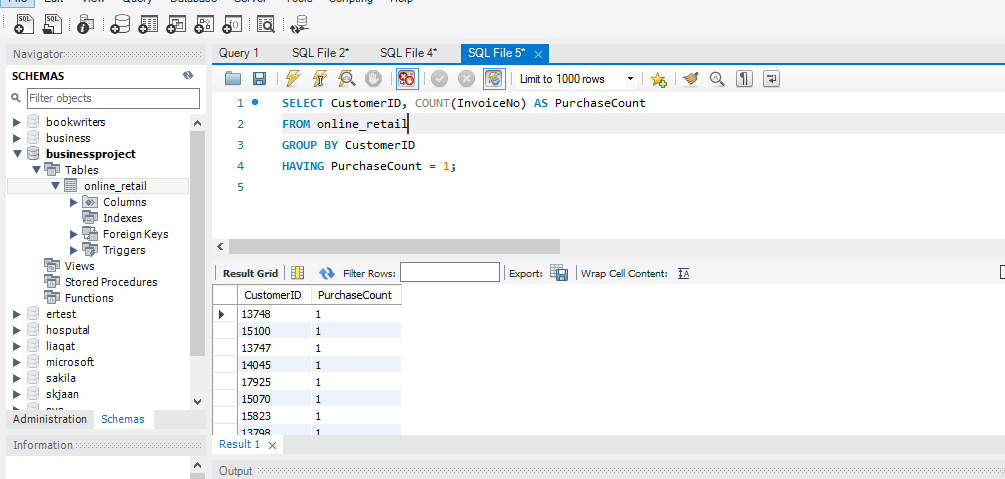
* We select the "CustomerID" column to identify each customer.
* We use the COUNT function with DISTINCT to count the unique StockCodes for each customer, representing the number of unique products they've purchased.
* We group the results by "CustomerID" to get counts for each customer.
* Finally, we order the results in descending order of the unique product count.



***Which customers have only made a single purchase from the company?***

To find customers who have made only a single purchase from the company.

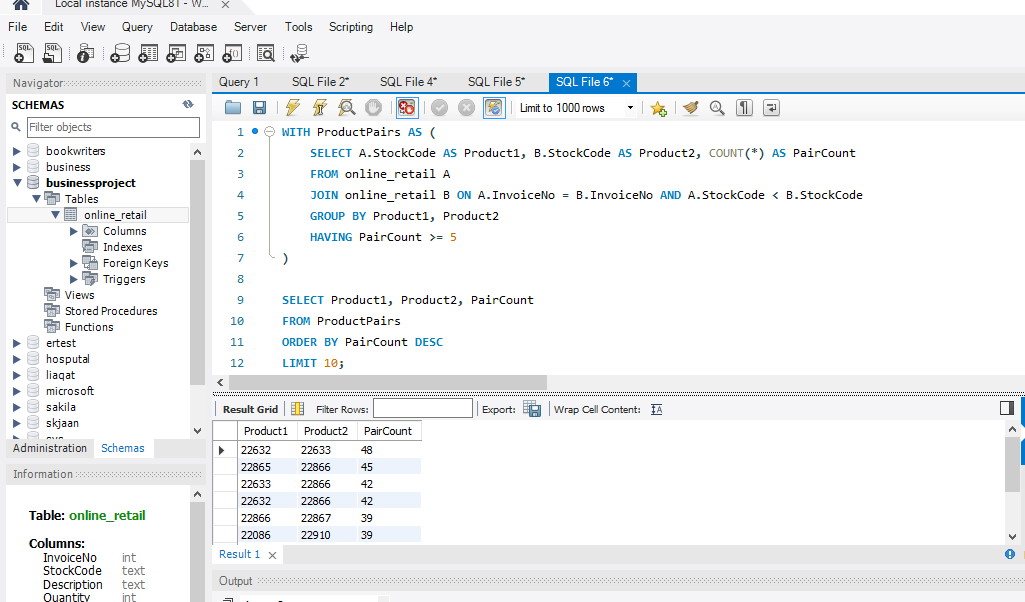
* We select the "CustomerID" column to identify each customer.
* We use the COUNT function to count the number of unique invoice numbers (purchases) for each customer.
* We group the results by "CustomerID" to get counts for each customer.
* Finally, we use the HAVING clause to filter the results to only include customers with a purchase count of 1.



***Which products are most commonly purchased together by customers in the dataset?***

To identify which products are most commonly purchased together by customers in the dataset, We can use SQL to analyze the order data and find pairs of products that are frequently bought together in the same invoice. This is often referred to as "market basket analysis" or "association rule mining."

* We create a Common Table Expression (CTE) named ProductPairs to find pairs of products that are purchased together. We join the Sales table with itself (aliased as A and B) on the condition that both products belong to the same invoice (InvoiceNo) and that the first product (A.StockCode) is less than the second product (B.StockCode) to avoid duplicate pairs.
* We use the COUNT(\*) function to count how many times each product pair occurs together in invoices.
* The HAVING clause specifies a threshold for the minimum number of occurrences (e.g., 5 times) to filter out less common pairs. You can adjust this threshold as needed.
* Finally, we select and order the results by the PairCount in descending order to identify the most commonly purchased product pairs. The LIMIT clause is used to limit the results to the top 10 pairs, but you can adjust this limit as needed.



***Customer Segmentation by Purchase Frequency***

Group customers into segments based on their purchase frequency, such as high, medium, and low frequency customers. This can help you identify your most loyal customers and those who need more attention.

To perform customer segmentation by purchase frequency (high, medium, and low), we can use SQL to analyze the dataset and categorize customers based on how frequently they make purchases.

***Calculate Purchase Frequency for Each Customer:***

We can calculate the purchase frequency for each customer by counting the number of unique invoices they have in the dataset. More invoices indicate higher purchase frequency.

***Define Segmentation Criteria***:

Decide on the criteria for segmenting customers into high, medium, and low-frequency groups. For example:

High-Frequency: Customers with a purchase frequency above a certain threshold (e.g., 10 or more).

Medium-Frequency: Customers with a purchase frequency between a lower threshold and the high-frequency threshold.

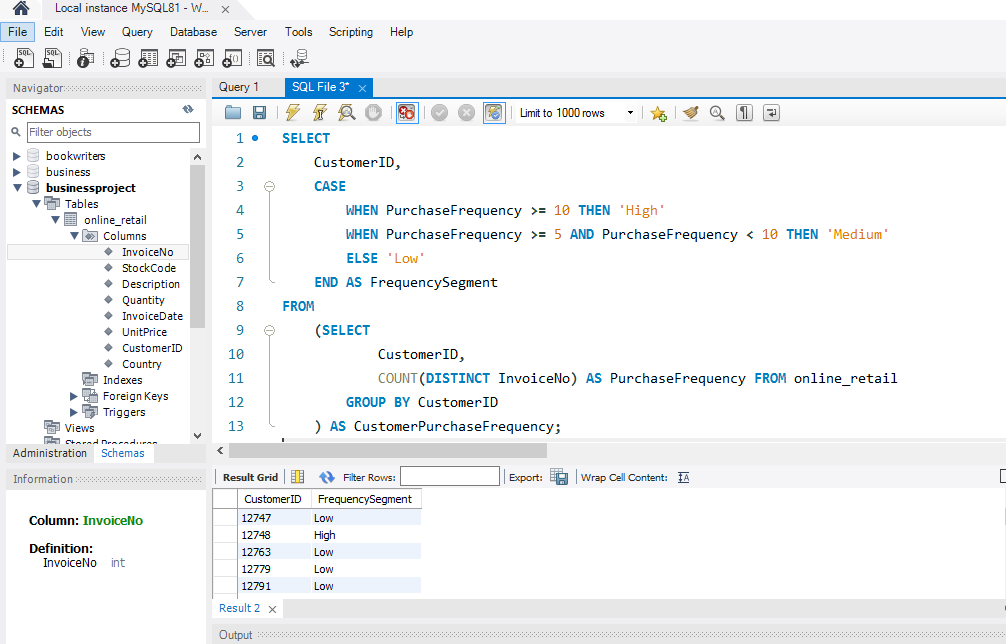
Low-Frequency: Customers with a purchase frequency below the lower threshold.

***Assign Segmentation Labels:***

We can assign labels to customers based on their purchase frequency using a CASE statement in SQL.

***Review the Results:***

After running the query, you will have a list of customers with their assigned frequency segments (High, Medium, or Low). You can use this information for marketing strategies, targeting promotions, or tailoring your communication to different customer segments.

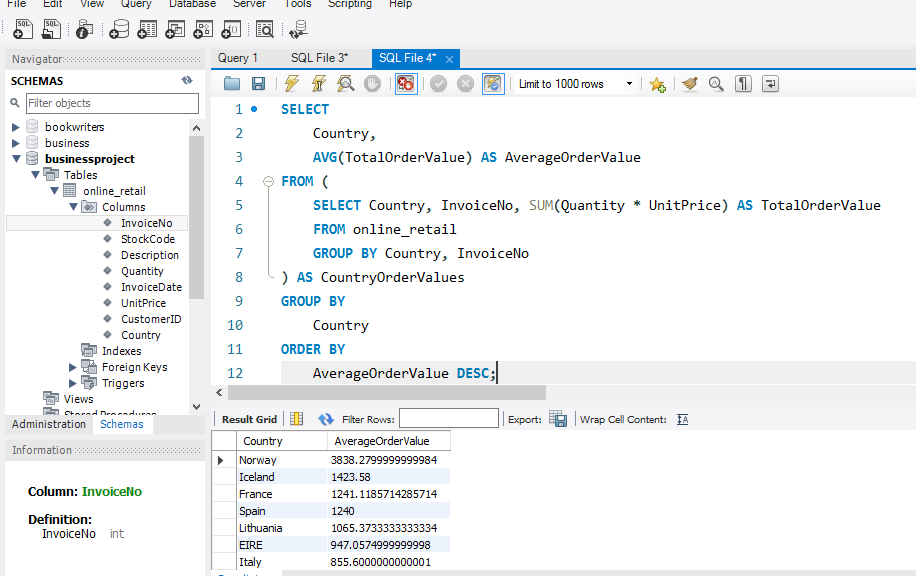
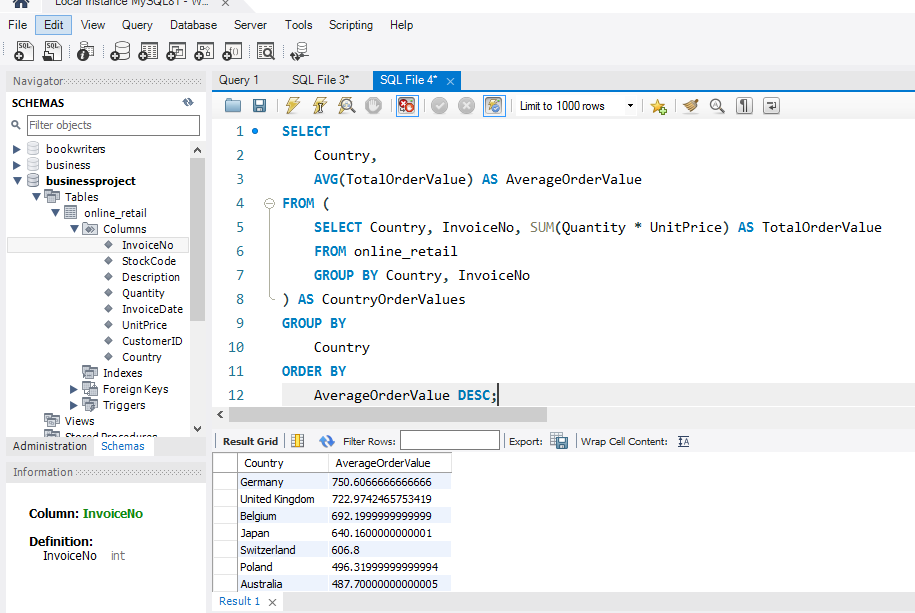


***2. Average Order Value by Country***

Calculate the average order value for each country to identify where your most valuable customers are located.

To calculate the average order value by country, you can use SQL to group the data by country and then calculate the average order value for each country.

1. We start by creating a subquery that calculates the total order value (TotalOrderValue) for each invoice in each country. This subquery groups the data by country and invoice number, calculating the total order value for each invoice.
2. We then use another outer query to group the results of the subquery by country and calculate the average order value (AverageOrderValue) for each country. The AVG function is used to compute the average of the total order values for each country.
3. Finally, we order the results in descending order of average order value to identify where your most valuable customers are located.

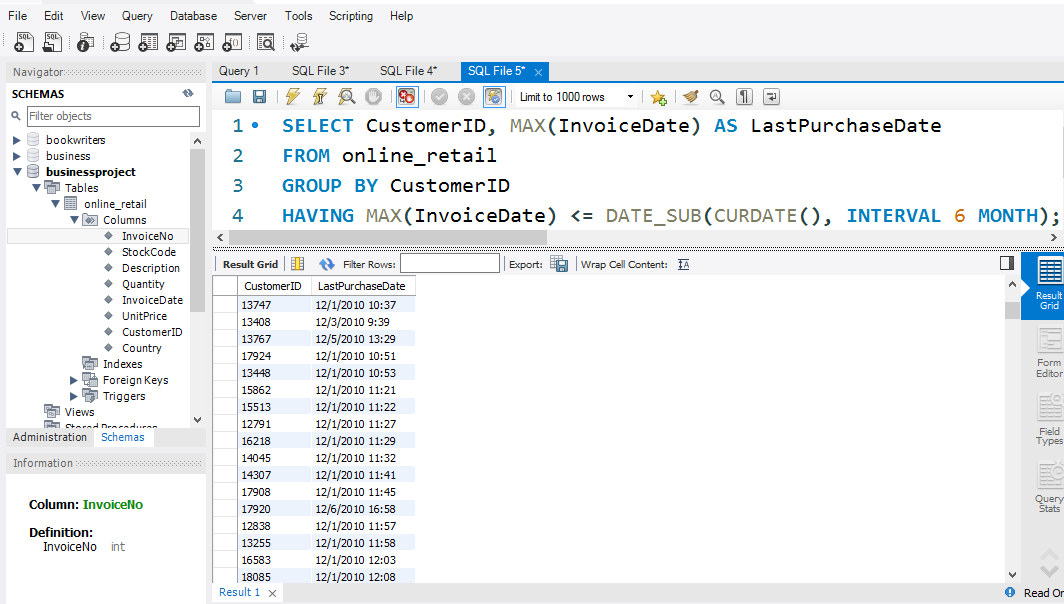
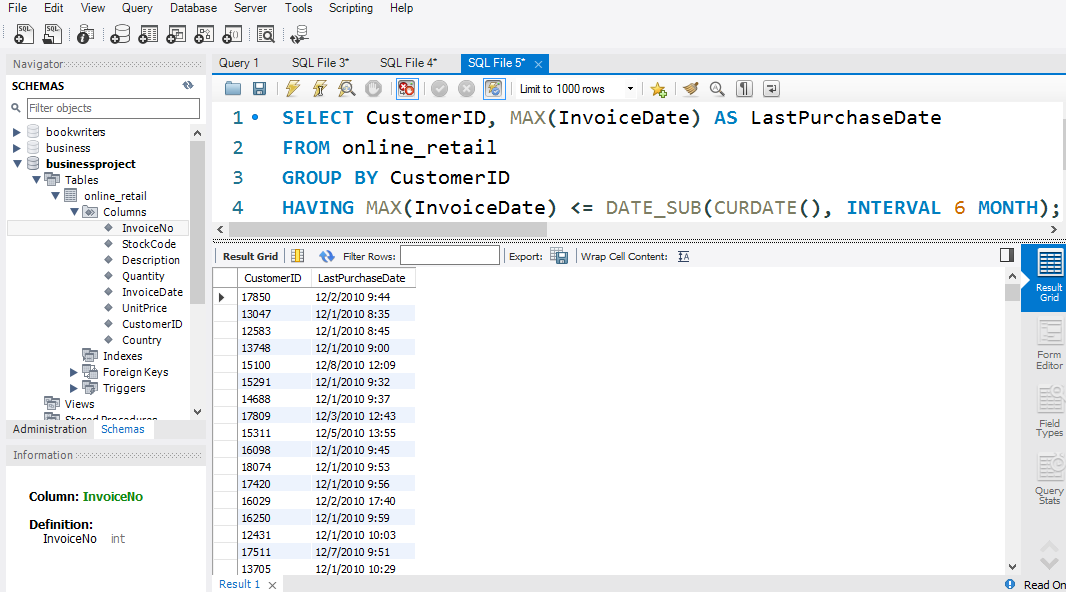


***3. Customer Churn Analysis***

Identify customers who haven't made a purchase in a specific period (e.g., last 6 months) to assess churn.

To identify customers who haven't made a purchase in a specific period (e.g., the last 6 months) for a customer churn analysis, we can use SQL to filter and select customers based on their purchase history.

1. We group the sales data by CustomerID to identify each customer's purchase history.
2. We use the MAX function to find the most recent InvoiceDate for each customer, which represents their last purchase date.
3. The HAVING clause filters the results to only include customers whose last purchase date is less than or equal to the current date minus 6 months. This condition identifies customers who haven't made a purchase in the last 6 months.

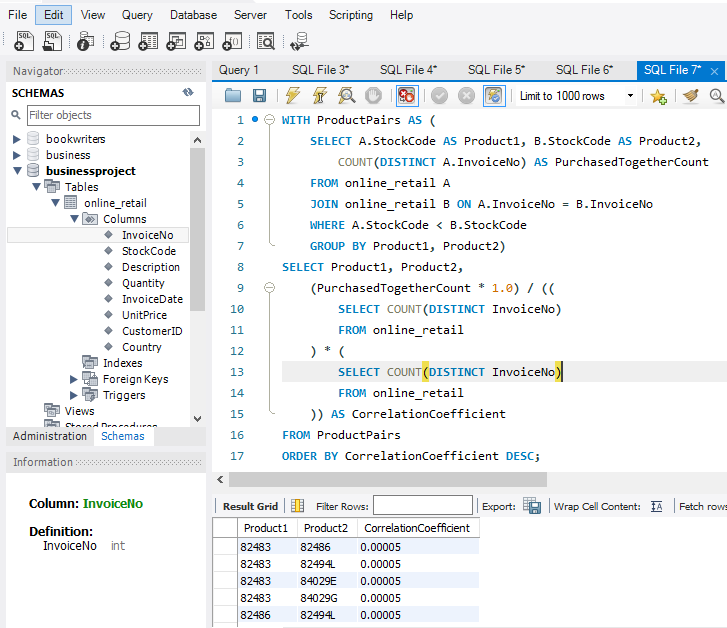
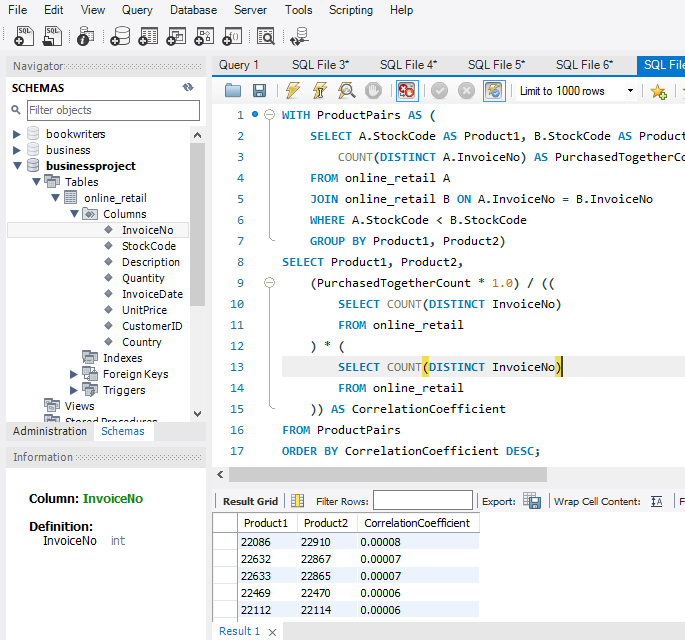
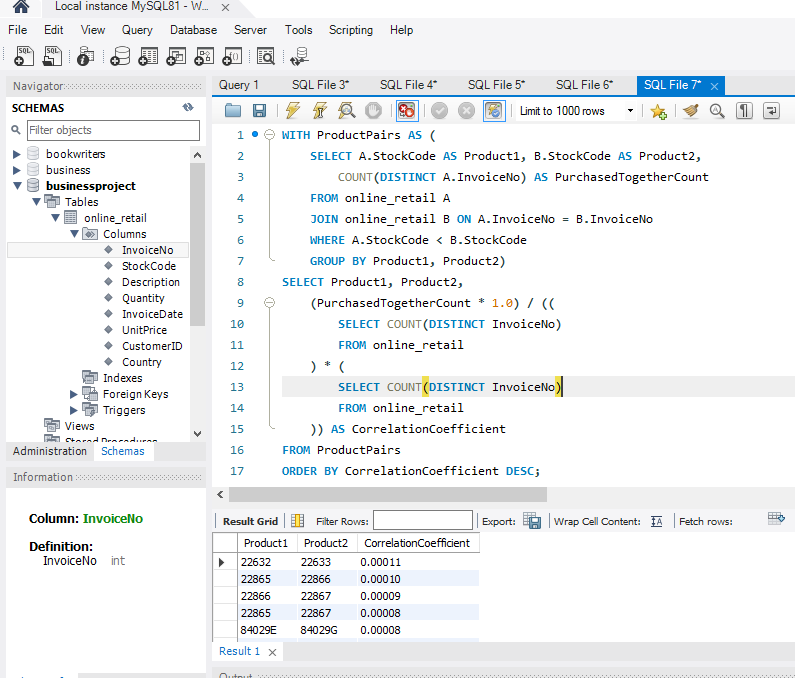


***4. Product Affinity Analysis***

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

To perform a product affinity analysis and calculate the correlation between product purchases, we can use SQL. Specifically, we can calculate the correlation coefficient between products by analyzing the purchase history of each product pair.

* We create a Common Table Expression (CTE) named ProductPairs to find pairs of products that are purchased together. We join the Sales table with itself (aliased as A and B) on the condition that both products belong to the same invoice (InvoiceNo), and we ensure that we only consider each pair once by filtering A.StockCode < B.StockCode.
* We count the number of unique invoices in which each product pair was purchased together (PurchasedTogetherCount).
* We calculate the correlation coefficient for each product pair by dividing the PurchasedTogetherCount by the product of the total number of unique invoices, effectively normalizing the count.
* The result is a list of product pairs with their corresponding correlation coefficients, ordered in descending order of correlation. Higher correlation coefficients indicate that the products are often purchased together, suggesting product affinity.



***5. Time-based Analysis***

Explore trends in customer behavior over time, such as monthly or quarterly sales patterns.

To perform a time-based analysis of customer behavior, such as exploring monthly or quarterly sales patterns, you can use SQL to aggregate and analyze the sales data over time periods.

1. We use the DATE\_FORMAT function to extract and format the year and month from the InvoiceDate column, giving us a "Month" column in the format "YYYY-MM".
2. We then calculate the following metrics for each month:
   * MonthlyOrderCount: The count of distinct invoices in that month.
   * MonthlyTotalQuantity: The total quantity of products sold in that month.
   * MonthlyTotalRevenue: The total revenue generated in that month.
3. We group the results by the "Month" column to aggregate data for each month.
4. Finally, we order the results by month in ascending order.

