High Impact Skills Development Program

in Artificial Intelligence, Data Science, and Blockchain

NAME: **BASHIR**

ROLL NO: **SK23168**

GMAIL: **baltistanimbashir@gmail.com**

SECTION: **DSAI –SKARDU-3**

Project Title: Online Retail Segmentation.

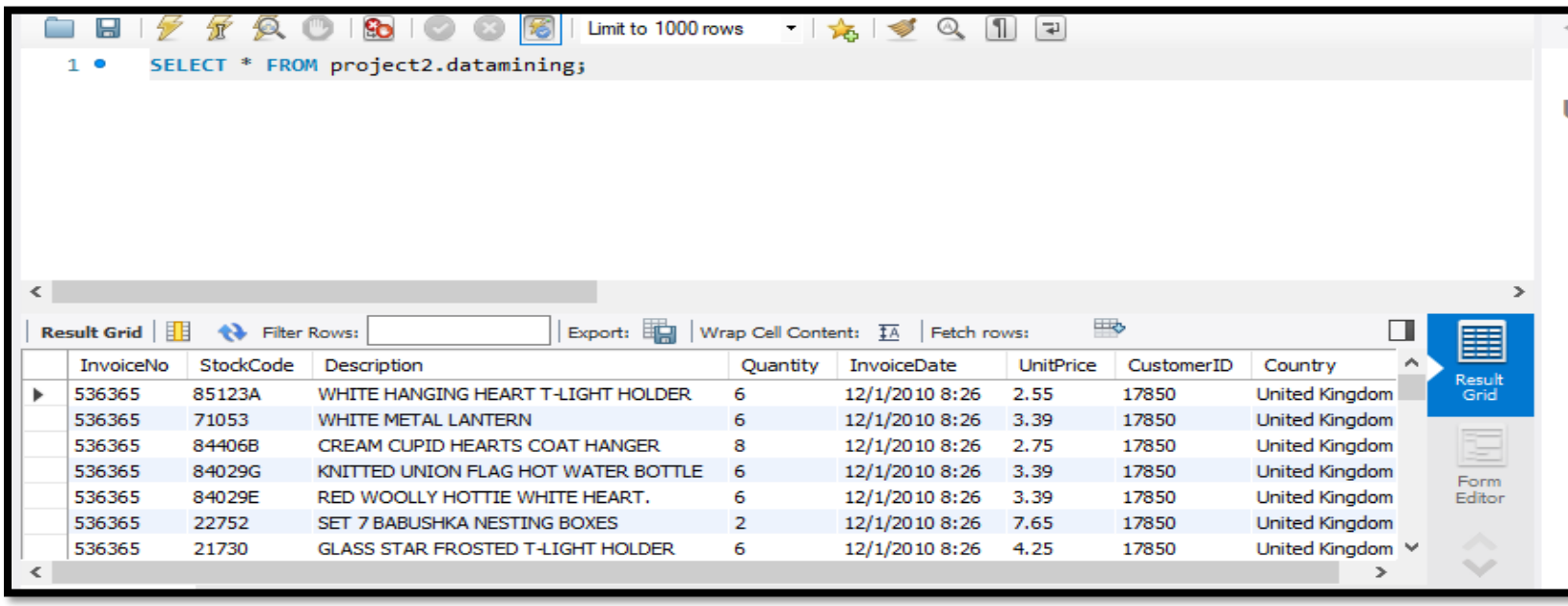
NUST

Defining Futures

1 Beginner Queries.

- Define meta data in mysql workbench

This code retrieves data from a table named "datamining" in a project called "project2" and the asterisk (*) means all columns will be selected.
The query fetches all rows and columns from the specified table.
It's a quick way to see the entire dataset's content.
This query can help in **understanding the data's structure and content for further analysis as shown below.**

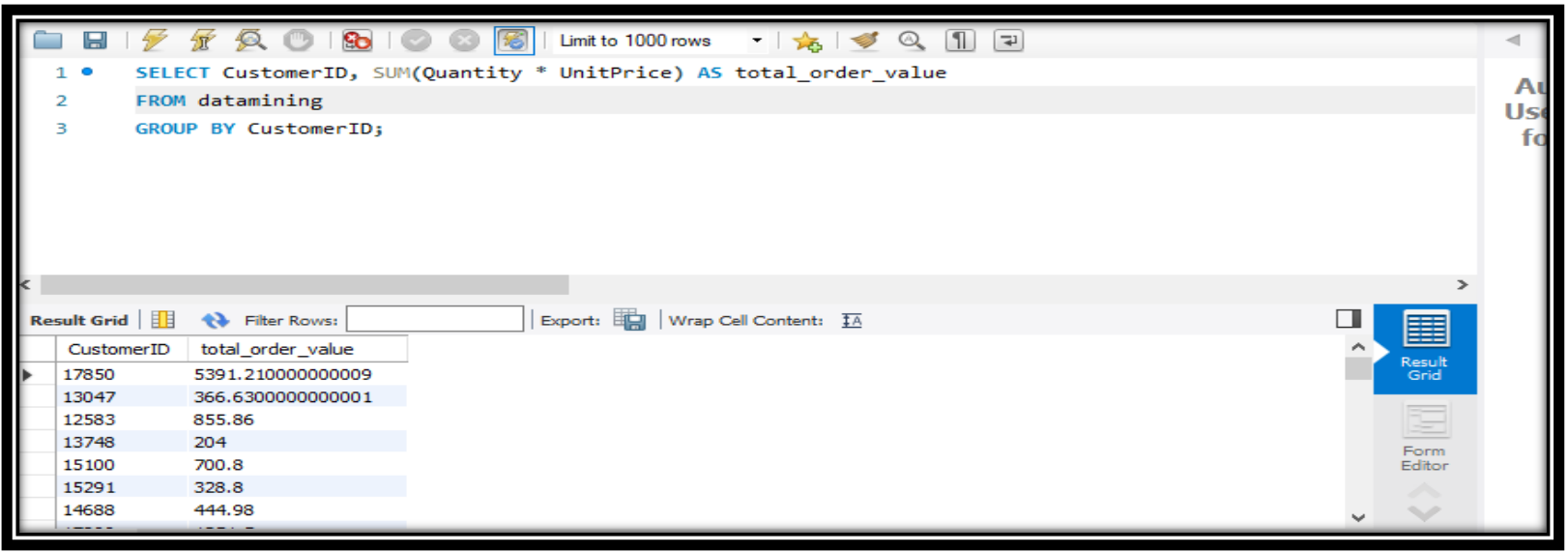


InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom

2

- What is the distribution of order values across all customers in the dataset? Top of Form

Customer Spending: This code calculates the total spending of each customer.
Data Selection: It selects the **CustomerID** from a dataset named "datamining".
Calculation: For each customer, it multiplies the **Quantity** of items by their UnitPrice and sums them up.
Grouping: Results are grouped by **CustomerID**, so each customer's spending is **summarized.Insightful Analysis:** The query helps understand individual customer spending patterns for better business insights.

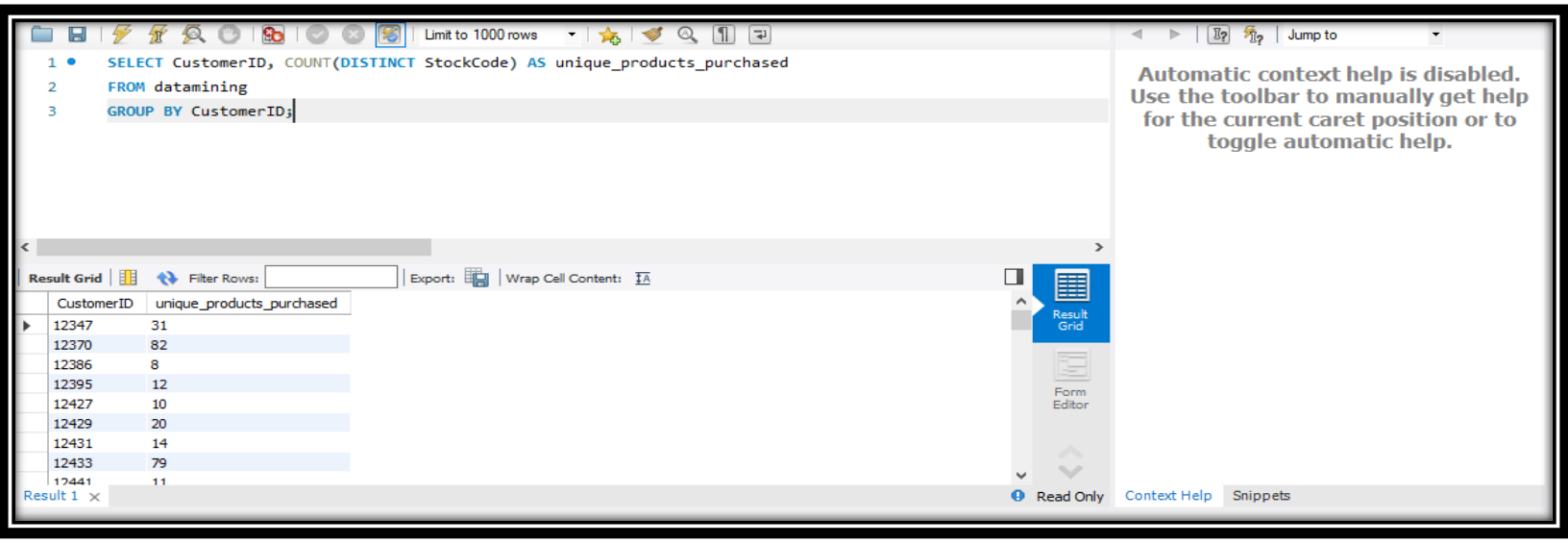


CustomerID	total_order_value
17850	5391.210000000009
13047	366.6300000000001
12583	855.86
13748	204
15100	700.8
15291	328.8
14688	444.98

3

• How many unique products has each customer purchased?

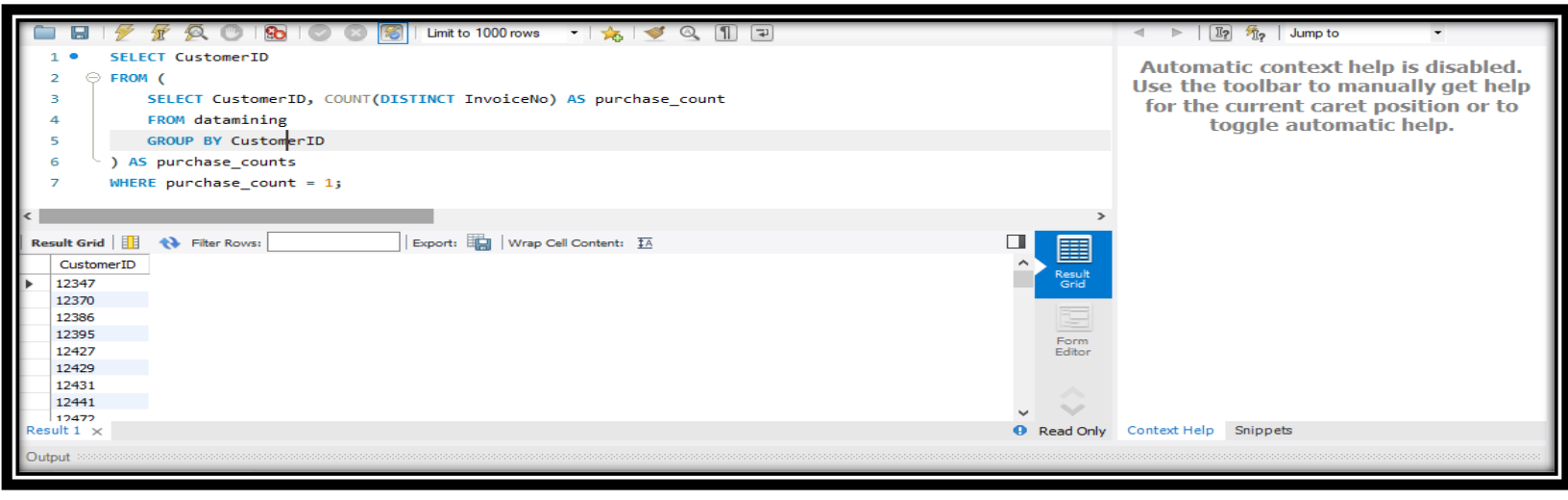
Product Diversity: This code counts unique products bought by each customer.
Data Selection: It selects **CustomerID** from a dataset named "datamining."
Counting Distinct: For each customer, it counts the distinct (different) **StockCode** values, representing unique products.
Grouping: Results are grouped by **CustomerID,** so you get the count for each customer.
Product Insight: The query helps uncover how many different products each customer has purchased, showing their shopping variety.



4

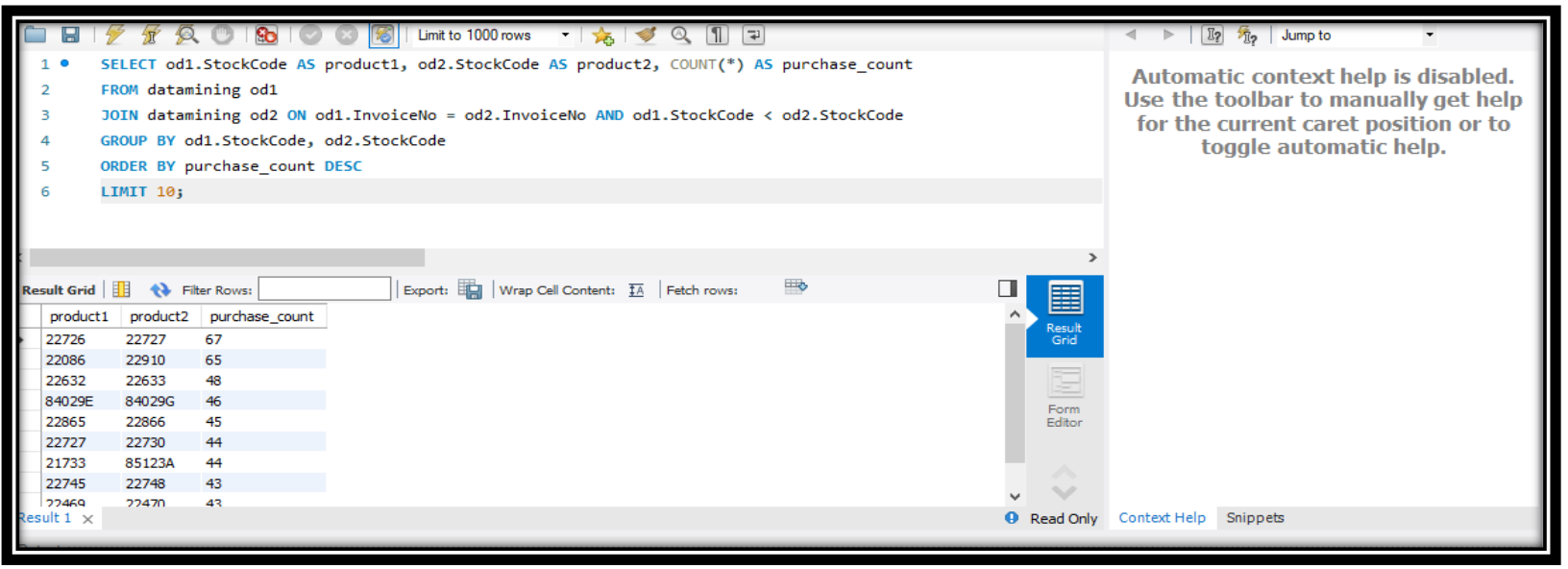
• Which customers have only made a single purchase from the company?

Single Purchase Customers: This code identifies customers who have made only one purchase.
Data Selection: It selects **CustomerID** from a dataset named "datamining."
Purchase Count: A subquery calculates, for each customer, the count of distinct **InvoiceNo** (purchase transactions).
Filtering: The outer query filters customers with a **purchase_count** of 1, indicating just one purchase.
Insightful Finding: The query helps pinpoint customers who have made a single transaction, aiding targeted engagement efforts.



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

Product Pair Analysis: This code identifies which pairs of products are often purchased together.
Data Selection: It selects two distinct product **StockCode** values from a dataset named "datamining."
Joining Transactions: The query joins the dataset with itself using **InvoiceNo**, ensuring it compares items from the same transactions.
Purchase Count: It counts how frequently each product pair appears together in transactions.
Top Pairs: Results are ordered by purchase count, showing the top 10 product pairs most commonly purchased together.



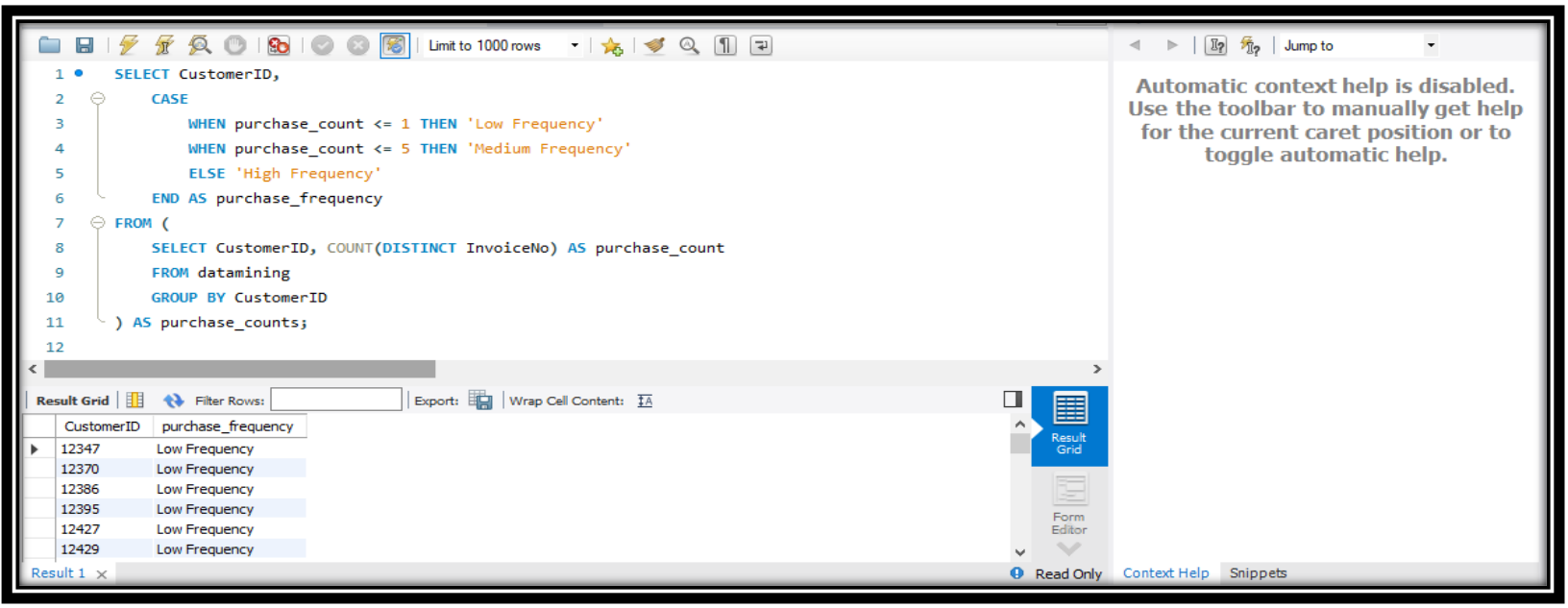
NUST

Defining Futures

Advance Queries

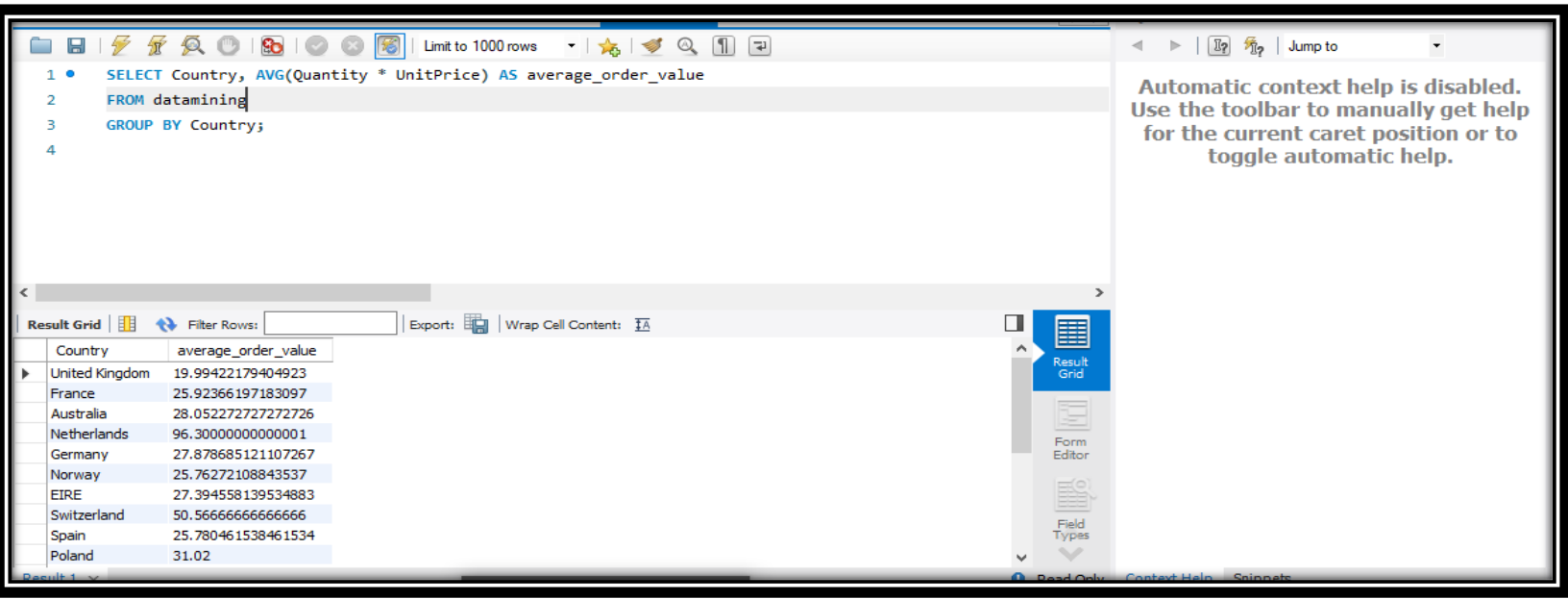
1. Customer Segmentation by Purchase Frequency

Customer Purchase Segmentation: This code groups customers based on their purchase frequency.
Data Selection: It selects **CustomerID** and calculates the distinct count of **InvoiceNo** from a dataset named "datamining."
Frequency Assignment: The query assigns a purchase frequency label to each customer based on their purchase count.
Segmentation Logic: Customers with low, medium, or high purchase counts are categorized accordingly.
Insightful Classification: The query helps segment customers into different purchase frequency groups, aiding targeted marketing strategies and engagement efforts.



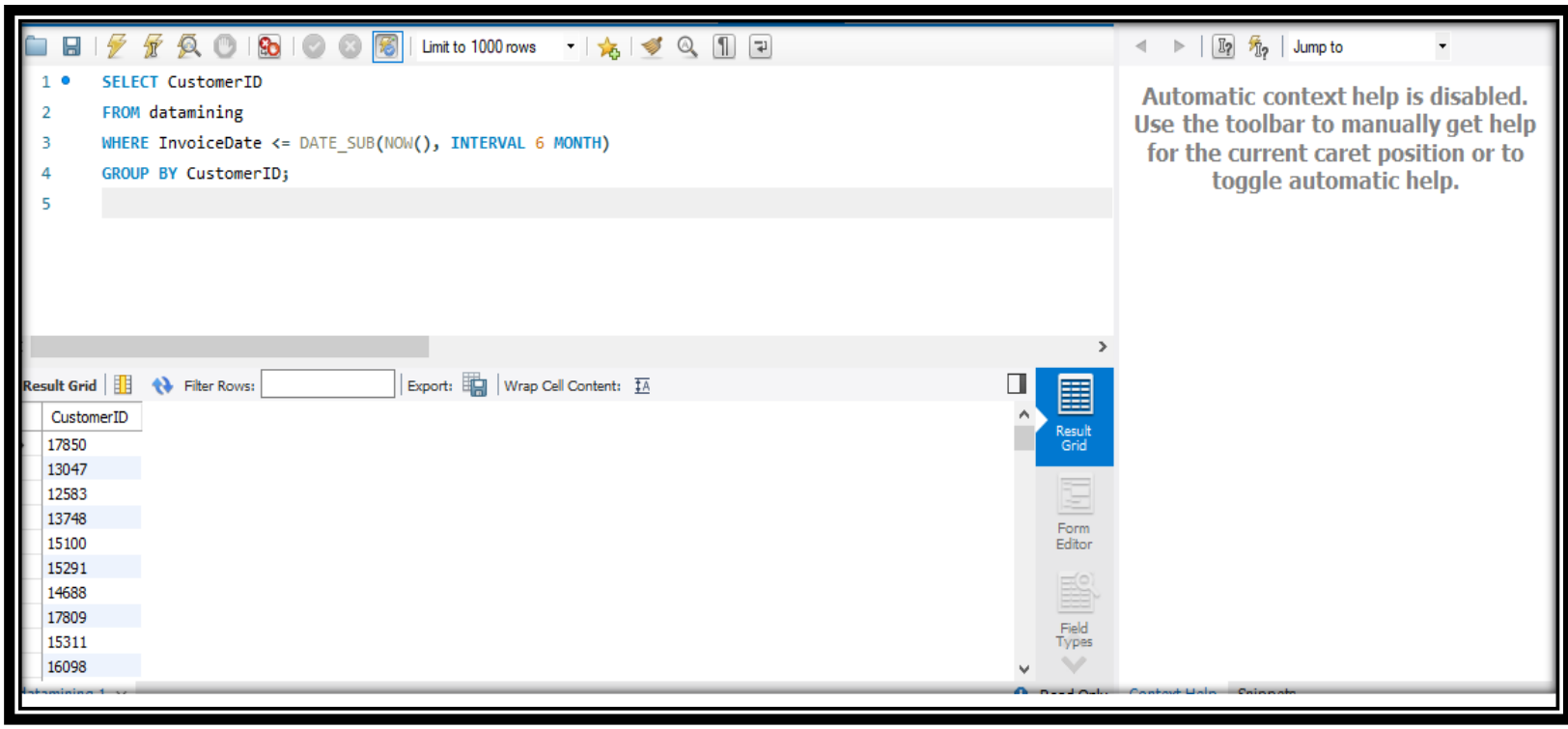
2. Average Order Value by Country

Country-wise Analysis: This code calculates the average order value for each country.
Data Selection: It selects **Country** from a dataset named "datamining."
Order Value Calculation: For each country, it calculates the average order value by multiplying **Quantity** with **UnitPrice** and finding the average.
Grouping: Results are grouped by **Country**, giving the average order value for each country.
Insightful Comparison: The query helps compare customer spending across different countries, aiding targeted marketing efforts and identifying high-value regions.



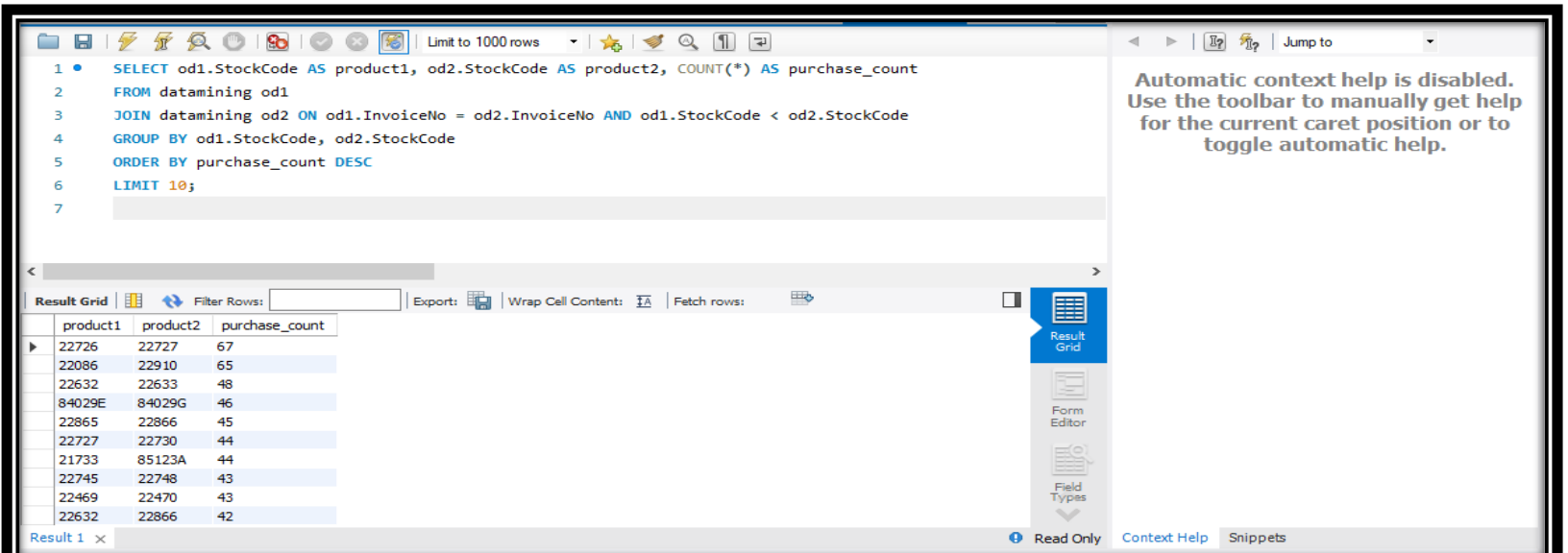
3. Customer Churn Analysis

Churn Analysis: This code identifies customers who haven't made a purchase in the last 6 months.
Data Selection: It selects **CustomerID** from a dataset named "datamining."
Date Filtering: The query checks if the **InvoiceDate** is within the last 6 months using **DATE_SUB** and **NOW()**.
Grouping: Results are grouped by **CustomerID**.
Inactive Customer Insight: The query helps find customers who might have stopped purchasing, aiding in churn analysis and targeted re-engagement strategies.



4. Product Affinity Analysis

Product Association: This code identifies pairs of products often purchased together.
Data Selection: It selects two distinct product **StockCode** values from a dataset named "datamining."
Joining Transactions: The query joins the dataset with itself using **InvoiceNo** to compare items from the same transactions.
Purchase Count: It counts how frequently each product pair appears together in transactions.
Top Associations: Results are ordered by purchase count in descending order, revealing the most common product pairs purchased together.



5. Time-based Analysis

Data Source: The code operates on a table named 'datamining'.
Date Components Extraction: It extracts the Year and Month from the **'InvoiceDate'** column.
Sales Calculation: It calculates the total sales for each month by summing the '**UnitPrice**' values.
Grouping: The results are grouped by Year and Month.
Insight Generation: This code is used to analyze and present the monthly total sales from the 'datamining' data, allowing insights into sales trends over time

```
2   YEAR (InvoiceDate) AS Year,
3   MONTH (InvoiceDate) AS MONTH,
4   SUM(UnitPrice) AS TotalSales
5   FROM
6   datamining
7   GROUP BY
8   year ,month;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Year	Month	TotalSales
2010	12	332188.80
2011	1	172752.80
2011	2	127448.77
2011	3	171486.51
2011	4	116747.96

NUST

Defining Futures

END