

ONLINE RETAIL SEGMENTATION.

BEGINNER QUERIES.

DEFINE META DATA IN MYSQL WORKBENCH

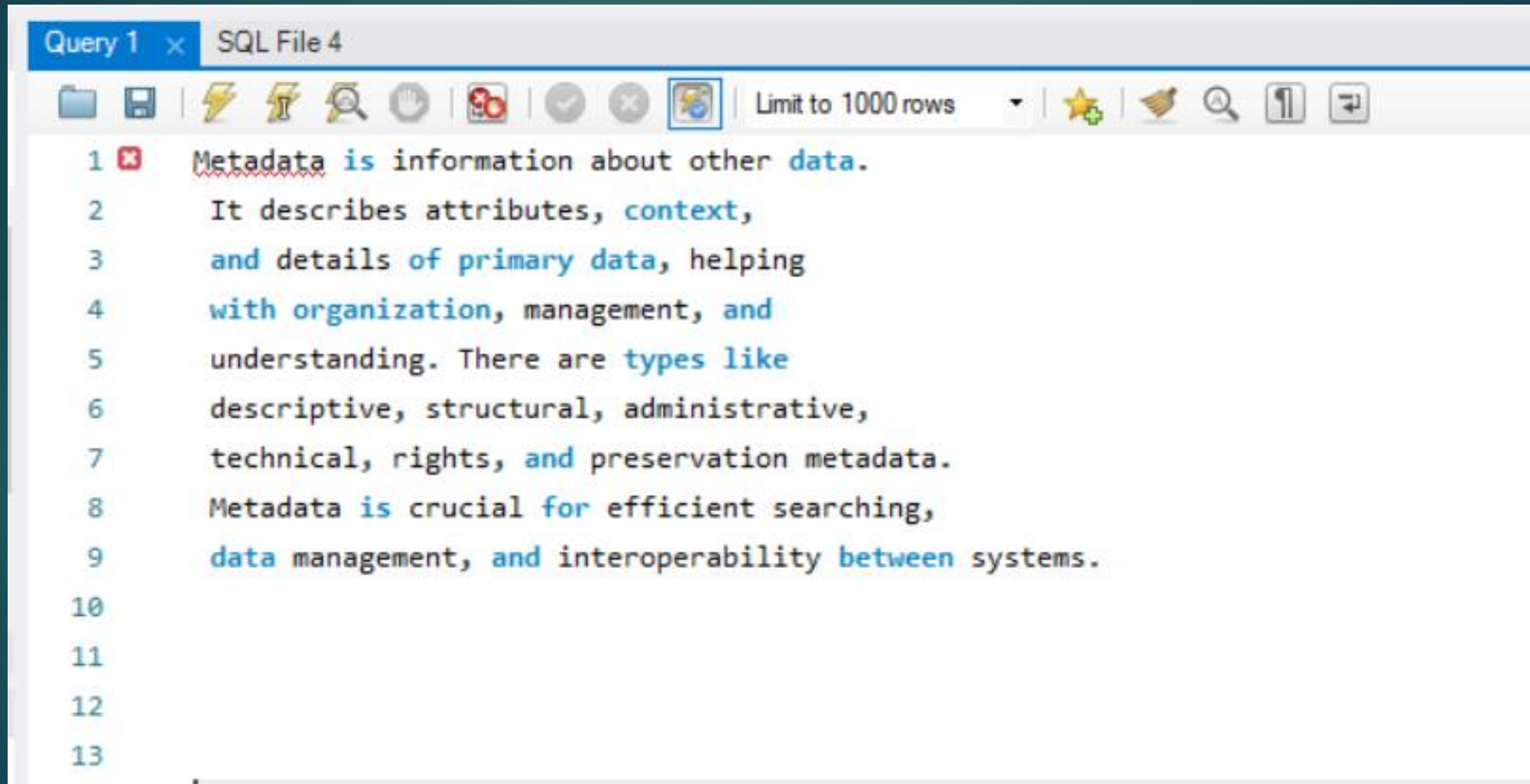
WHAT IS THE DISTRIBUTION OF ORDER VALUES ACROSS ALL CUSTOMERS IN THE DATASET?

HOW MANY UNIQUE PRODUCTS HAS EACH CUSTOMER PURCHASED?

WHICH CUSTOMERS HAVE ONLY MADE A SINGLE PURCHASE FROM THE COMPANY?

WHICH PRODUCTS ARE MOST COMMONLY PURCHASED TOGETHER BY CUSTOMERS IN THE DATASET?




Define meta data in mysql workbench

A screenshot of the MySQL Workbench SQL Editor interface. The window title is 'Query 1 x SQL File 4'. The toolbar includes icons for file operations (folder, save, lightning bolt, copy, paste, search, zoom in, zoom out), execution (play, stop, refresh), and other features like 'Limit to 1000 rows', a star icon, a hand icon, and a refresh icon. The SQL editor area contains a single line of text: 'Metadata is information about other data.' followed by several lines of descriptive text. The text is color-coded: 'Metadata' is red, 'is' is blue, 'information' is black, 'about' is black, 'other' is blue, and 'data.' is blue. The text continues: 'It describes attributes, context, and details of primary data, helping with organization, management, and understanding. There are types like descriptive, structural, administrative, technical, rights, and preservation metadata. Metadata is crucial for efficient searching, data management, and interoperability between systems.' The line numbers 1 through 13 are visible on the left side of the editor.

```
1 Metadata is information about other data.  
2 It describes attributes, context,  
3 and details of primary data, helping  
4 with organization, management, and  
5 understanding. There are types like  
6 descriptive, structural, administrative,  
7 technical, rights, and preservation metadata.  
8 Metadata is crucial for efficient searching,  
9 data management, and interoperability between systems.  
10  
11  
12  
13
```

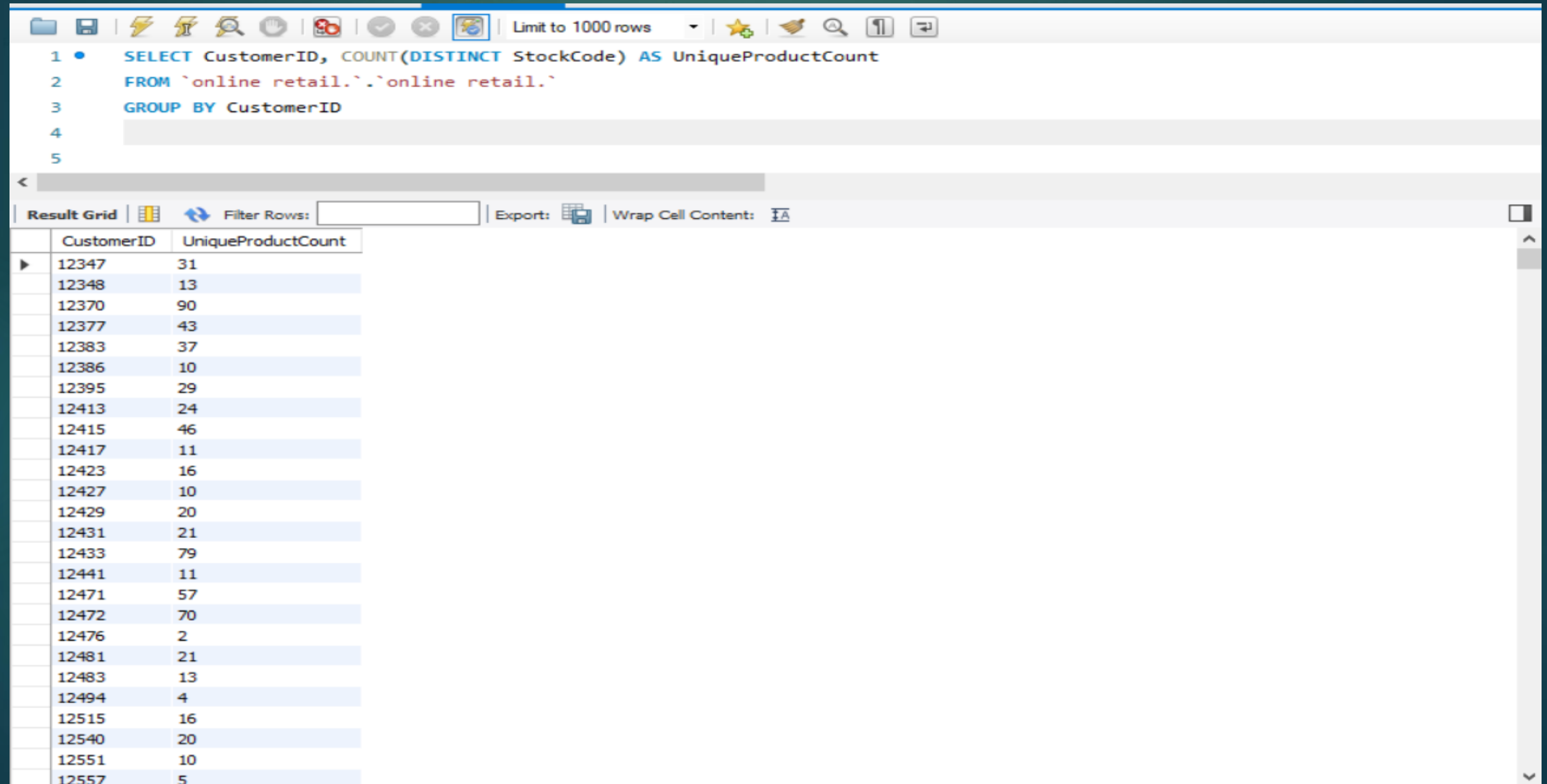
What is the distribution of order values across all customers in the dataset?

```
1 • SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue
2 FROM `online retail`.`online retail`
3 GROUP BY CustomerID
4 ORDER BY TotalOrderValue DESC;
5
```

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

CustomerID	TotalOrderValue
18102	27834.61
15061	19950.660000000007
16029	13202.52
17511	10573.219999999998
14646	8591.879999999997
13089	7738.670000000001
14911	7737.939999999999
12415	7011.379999999997
16210	7000.639999999999
13777	6961.78
13081	5894.419999999997

How many unique products has each customer purchased?



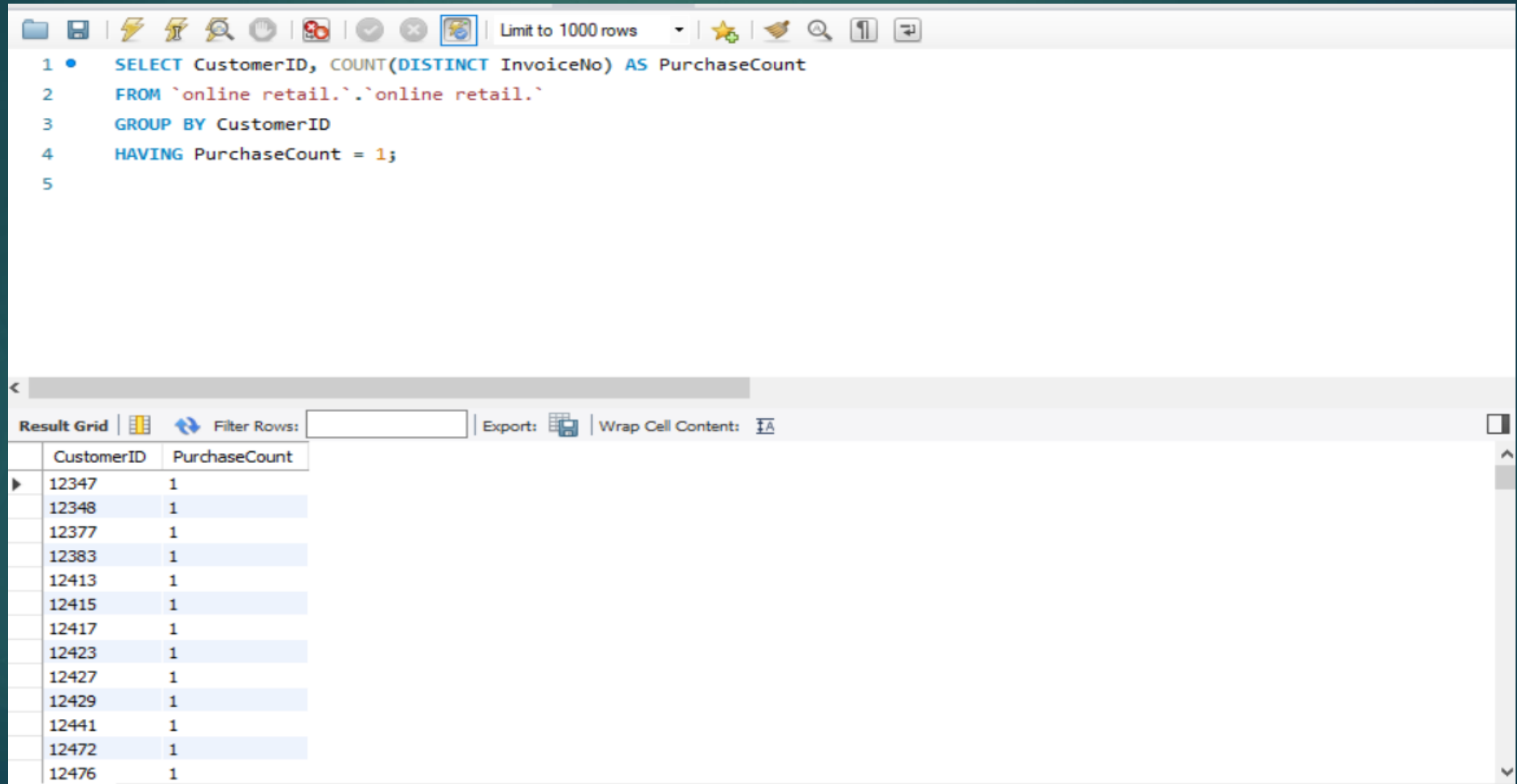
The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```
1 • SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductCount
2 FROM `online retail`.`online retail`
3 GROUP BY CustomerID
4
5
```

Below the query editor is a "Result Grid" section. It includes a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. The grid displays the results of the query in a table with two columns: "CustomerID" and "UniqueProductCount". There are 25 rows of data.

CustomerID	UniqueProductCount
12347	31
12348	13
12370	90
12377	43
12383	37
12386	10
12395	29
12413	24
12415	46
12417	11
12423	16
12427	10
12429	20
12431	21
12433	79
12441	11
12471	57
12472	70
12476	2
12481	21
12483	13
12494	4
12515	16
12540	20
12551	10
12557	5

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



The screenshot shows a SQL query editor window with a toolbar at the top. The query is as follows:

```
1 • SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS PurchaseCount
2 FROM `online retail`.`online retail`
3 GROUP BY CustomerID
4 HAVING PurchaseCount = 1;
5
```

Below the query editor is the 'Result Grid' section, which includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with two columns: 'CustomerID' and 'PurchaseCount'.

CustomerID	PurchaseCount
12347	1
12348	1
12377	1
12383	1
12413	1
12415	1
12417	1
12423	1
12427	1
12429	1
12441	1
12472	1
12476	1

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

The screenshot shows a SQL query editor with the following query:

```
1 • SELECT A.StockCode AS Product1, B.StockCode AS Product2, COUNT(*) AS Frequency
2 FROM `online retail`.`online retail.` A
3 JOIN `online retail`.`online retail.` B ON A.InvoiceNo = B.InvoiceNo AND A.StockCode < B.StockCode
4 GROUP BY Product1, Product2
5 ORDER BY Frequency DESC
6 LIMIT 10;
7
```

Below the query, the results are displayed in a table with columns Product1, Product2, and Frequency. The table shows the top 10 most frequently purchased product pairs.

Product1	Product2	Frequency
22726	22727	82
22086	22910	71
21733	85123A	61
22469	22470	59
21485	22111	57
22111	22112	56
82482	82494L	56
22470	85123A	52
22727	22730	51
22112	22835	51

Advance Queries

- ▶ **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.

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

- ▶ Calculate the average order value for each country 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.

- ▶ **4. Product Affinity Analysis**

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

- ▶ **5. Time-based Analysis**

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



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.

```
1 • SELECT CustomerID,
2     CASE
3         WHEN PurchaseCount > 5 THEN 'High Frequency'
4         WHEN PurchaseCount > 2 THEN 'Medium Frequency'
5         ELSE 'Low Frequency'
6     END AS PurchaseFrequencySegment
7 FROM (
8     SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS PurchaseCount
9     FROM `online retail`.`online retail`
10    GROUP BY CustomerID
```

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

CustomerID	PurchaseFrequencySegment
12921	Medium Frequency
12928	Low Frequency
12931	Low Frequency
12942	Low Frequency
12944	Low Frequency
12947	Low Frequency
12957	Low Frequency
12963	Low Frequency
12967	Low Frequency
12971	High Frequency

2. Average Order Value by Country

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

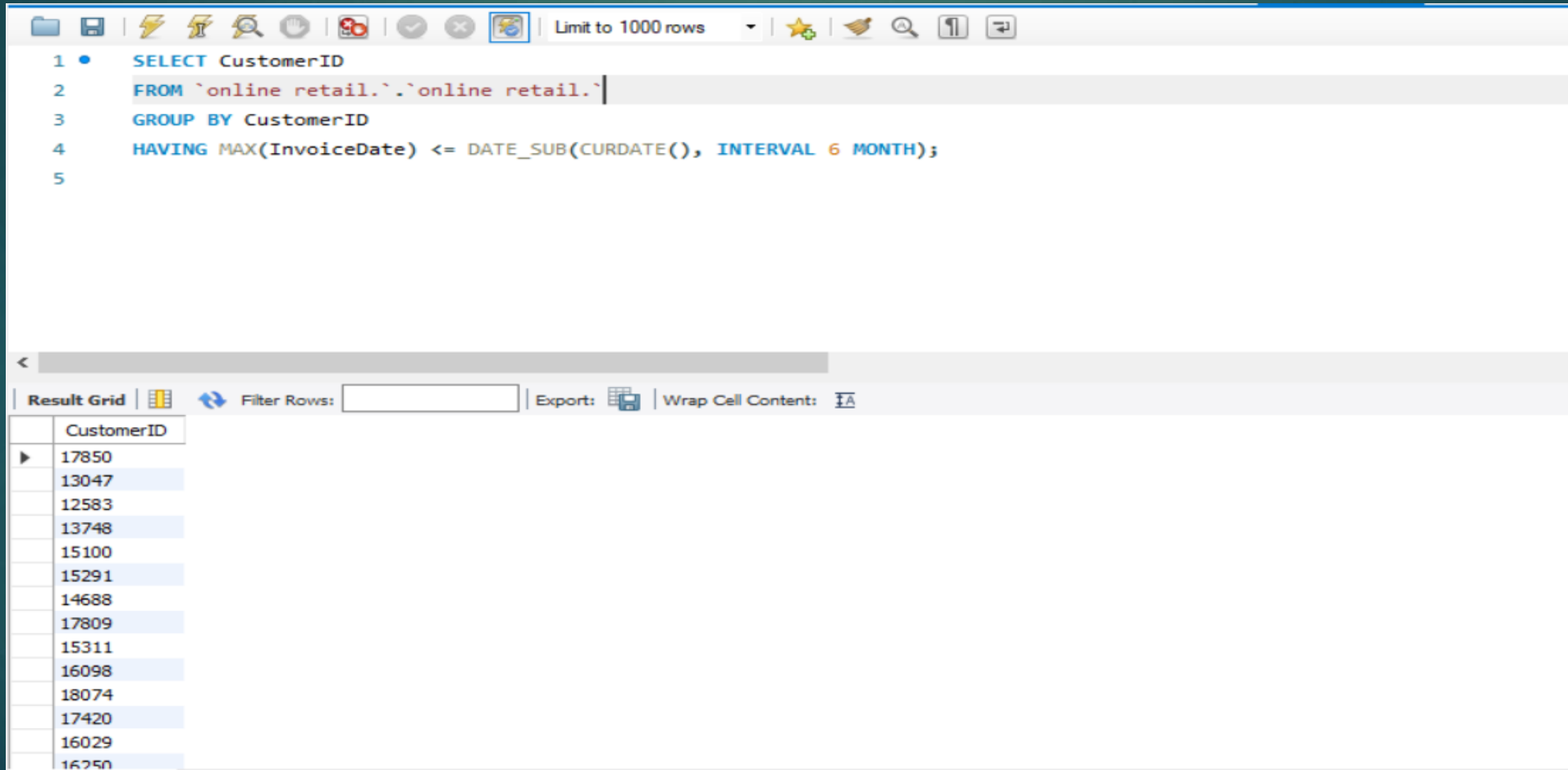
```
1 • SELECT Country, AVG(TotalOrderValue) AS AverageOrderValue
2 FROM (
3     SELECT Country, InvoiceNo, SUM(Quantity * UnitPrice) AS TotalOrderValue
4     FROM `online retail`.`online retail`
5     GROUP BY Country, InvoiceNo
6 ) AS CountryOrderValues
7 GROUP BY Country
8 ORDER BY AverageOrderValue DESC;
9
```

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

Country	AverageOrderValue
Netherlands	2928.1599999999994
Japan	2568.3566666666666
Norway	1893.5600000000004
Australia	1637.4459999999995
Cyprus	1590.82
Sweden	1460.62
Denmark	1281.5000000000002
Finland	892.8000000000001
Iceland	711.79
Switzerland	652.4599999999998
EIRE	587.592
Germany	508.03800000000007
France	472.26040000000006
Spain	460.9325

3. Customer Churn Analysis

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



The screenshot shows a SQL query editor with the following query:

```
1 SELECT CustomerID
2 FROM `online retail`.`online retail`
3 GROUP BY CustomerID
4 HAVING MAX(InvoiceDate) <= DATE_SUB(CURDATE(), INTERVAL 6 MONTH);
5
```




Below the query editor, the results are displayed in a table with the following columns and rows:

CustomerID
17850
13047
12583
13748
15100
15291
14688
17809
15311
16098
18074
17420
16029
16250

. Time-based Analysis

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

```
1 • SELECT
2     DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month,
3     SUM(TotalOrderValue) AS TotalSales
4 FROM (
5     SELECT
6         InvoiceDate,
7         SUM(Quantity * UnitPrice) AS TotalOrderValue
8     FROM
9         `online retail`.`online retail`
10    GROUP BY
11        InvoiceNo, InvoiceDate
12 ) AS InvoiceTotals
13 GROUP BY
14     Month
15 ORDER BY
16     Month;
```

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

Month	TotalSales
HULL	647876.6700000007

4. Product Affinity Analysis

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

```
1 • SELECT
2     p1.StockCode AS Product1,
3     p2.StockCode AS Product2,
4     COUNT(DISTINCT p1.InvoiceNo) AS CoPurchaseCount
5 FROM
6     `online retail`.`online retail.` p1
7 JOIN
8     `online retail`.`online retail.` p2 ON p1.InvoiceNo = p2.InvoiceNo AND p1.StockCode < p2.StockCode
9 GROUP BY
10    Product1, Product2
11 HAVING
12    CoPurchaseCount > 10 -- Adjust the threshold as needed
13 ORDER BY
14    CoPurchaseCount DESC
15 LIMIT
16    10;
```

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

Product1	Product2	CoPurchaseCount
22086	22910	60
22469	22470	58
21733	85123A	57
22726	22727	57
21485	22111	53
82482	82494L	52
22111	22112	51
22112	22835	48
84029E	84029G	47
22470	85123A	46

Export recordset to an external file



Github project link

<https://github.com/atheralizair/datamining>