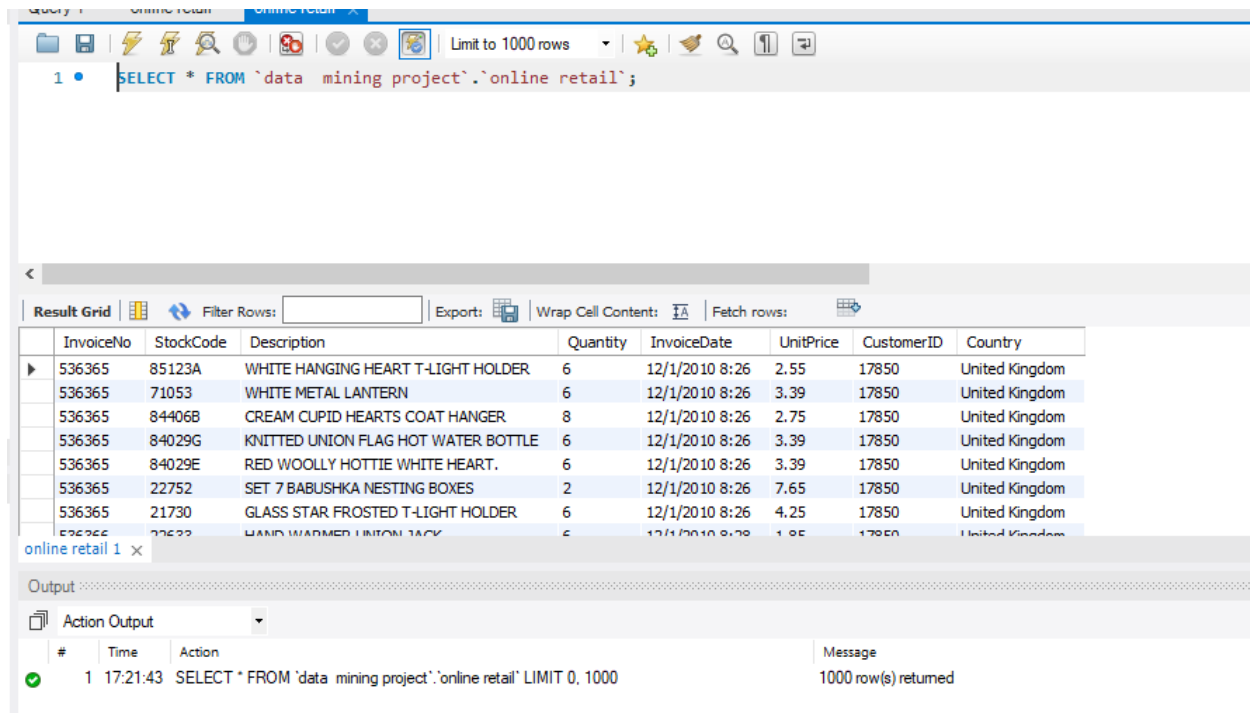


# DATA MINING PROJECT

## 1-METADATA

### a. Define metadata in my SQL workbench ?



The screenshot displays the MySQL Workbench interface. The SQL editor at the top contains the query: `SELECT * FROM `data mining project`.`online retail`;`. Below the editor, the 'Result Grid' tab is active, showing a table with 9 columns: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country. The table contains 10 rows of data, all from the same customer (17850) and date (12/1/2010 8:26). The 'Output' panel at the bottom shows a successful execution message: '1000 row(s) returned'.

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
536365	22632	HAND WARMER UNION JACK	6	12/1/2010 8:26	1.95	17850	United Kingdom
536365	22632	HAND WARMER UNION JACK	6	12/1/2010 8:26	1.95	17850	United Kingdom

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

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

```
1 • Select CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue
2 FROM dataminig
3 GROUP BY CustomerID
4 ORDER BY TotalOrderValue DESC;
```

Below the query editor is a "Result Grid" showing the top 10 results. The columns are "CustomerID" and "TotalOrderValue".

CustomerID	TotalOrderValue
15061	9407.339999999998
13777	6585.16
17850	5391.210000000009
16029	4271.52
16210	2474.7399999999993
14911	2283.5299999999997
16754	2002.4
17433	1010.1400000000008

The interface includes a toolbar with icons for file operations, a "Limit to 1000 rows" dropdown, and a "Result Grid" button. The status bar at the bottom indicates "Read Only" and "Con".

- c. How many unique products has each customer purchased?

The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

```
1 • SELECT CustomerID, COUNT(DISTINCT StockCode) AS UniqueProductsPurchased
2 FROM datamining
3 GROUP BY CustomerID;
```

The results grid displays the following data:

CustomerID	UniqueProductsPurchased
12427	10
12431	14
12433	73
12557	5
12583	20
12600	2
12662	15
12687	16

The interface includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'Result Grid' button. The status bar at the bottom indicates 'Read Only' and 'Col'.

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

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

```
1 • SELECT CustomerID
2 FROM datamini
3 GROUP BY CustomerID
4 HAVING COUNT(DISTINCT InvoiceNo) = 1;
```

Below the query editor, the 'Result Grid' is displayed, showing a table with the following data:

CustomerID
12427
12431
12433
12557
12583
12600
12662
12687

The interface includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field. The bottom status bar indicates 'datamini 1 x' and 'Read Only'.

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

The screenshot shows a SQL query editor with a query window and a result grid. The query is designed to find the most commonly purchased product pairs by joining the 'dataminig' table with itself on 'InvoiceNo' and ordering by frequency.

**Query:**

```
1 • SELECT a.StockCode AS Product1, b.StockCode AS Product2, COUNT(*) AS Frequency
2 FROM dataminig a
3 JOIN dataminig b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode < b.StockCode
4 GROUP BY Product1, Product2
5 ORDER BY Frequency DESC;
6
```

**Result Grid:**

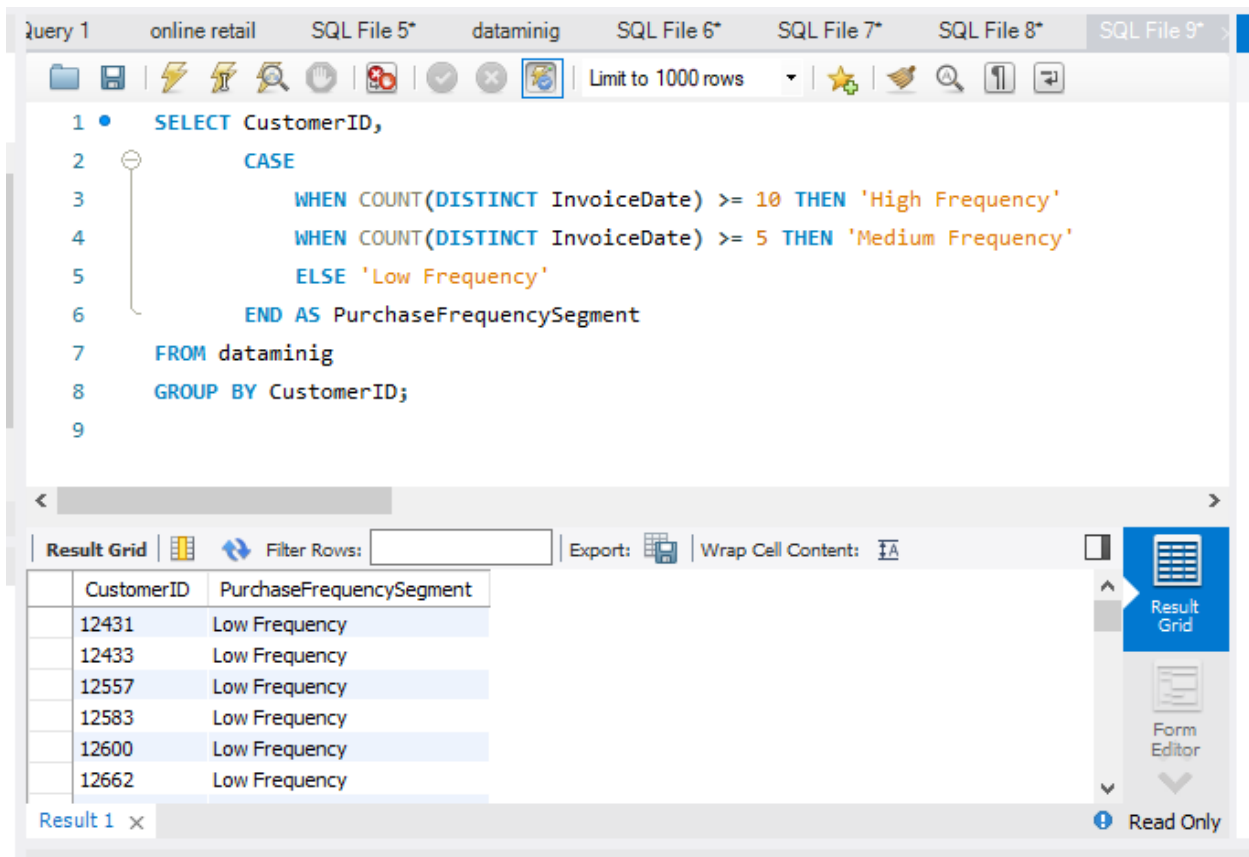
	Product1	Product2	Frequency
▶	22632	22633	26
	84029E	84029G	21
	84029E	85123A	20
	84029G	85123A	20
	21730	85123A	18
	71053	85123A	18
	22752	85123A	17
	21730	71053	17
	21730	84029G	17

Result 1 x Read Only

Output

# Advance Queries

## 1. Customer Segmentation by Purchase Frequency



The screenshot shows a SQL IDE interface with a query editor and a result grid. The query is designed to segment customers based on their purchase frequency by counting distinct invoice dates.

```
1 • SELECT CustomerID,  
2     CASE  
3         WHEN COUNT(DISTINCT InvoiceDate) >= 10 THEN 'High Frequency'  
4         WHEN COUNT(DISTINCT InvoiceDate) >= 5 THEN 'Medium Frequency'  
5         ELSE 'Low Frequency'  
6     END AS PurchaseFrequencySegment  
7 FROM dataminig  
8 GROUP BY CustomerID;  
9
```

The result grid displays the following data:

CustomerID	PurchaseFrequencySegment
12431	Low Frequency
12433	Low Frequency
12557	Low Frequency
12583	Low Frequency
12600	Low Frequency
12662	Low Frequency

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Result Grid' button on the right. The status bar at the bottom indicates 'Result 1' and 'Read Only'.

## 2. Average Order Value by Country

The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

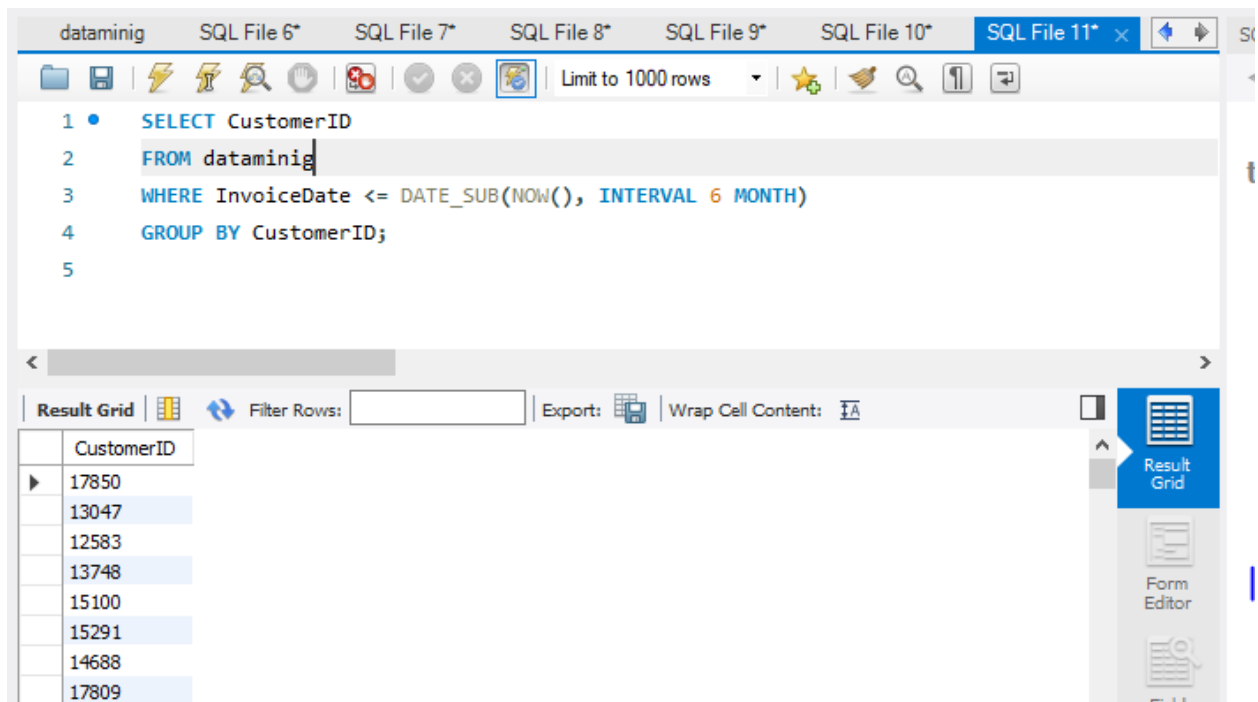
```
1 • SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue
2 FROM dataminig
3 GROUP BY Country
4 ORDER BY AvgOrderValue DESC;
5
```

The results grid displays the following data:

Country	AvgOrderValue
Spain	124
Netherlands	96.30000000000001
Switzerland	50.56666666666666
Poland	31.02
France	29.796279069767447
EIRE	27.143020833333335
Norway	26.2895890410959
Australia	25.589285714285715
United Kingdom	23.07580390436475
Germany	20.68851063829787

The interface includes a toolbar with various icons, a "Limit to 1000 rows" dropdown, and a "Result Grid" button on the right. The bottom of the window shows an "Output" section.

### 3. Customer Churn Analysis



The screenshot shows a SQL IDE interface with multiple tabs. The active tab is 'SQL File 11\*'. The query editor contains the following SQL code:

```
1 • SELECT CustomerID
2 FROM dataminig
3 WHERE InvoiceDate <= DATE_SUB(NOW(), INTERVAL 6 MONTH)
4 GROUP BY CustomerID;
5
```

Below the query editor, the 'Result Grid' is displayed, showing a table with one column, 'CustomerID', and seven rows of data:

CustomerID
17850
13047
12583
13748
15100
15291
14688
17809

The interface also includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field.



## 4. Product Affinity Analysis

The screenshot shows a SQL IDE interface with multiple tabs labeled 'SQL File 7\*' through 'SQL File 13\*'. The active tab is 'SQL File 13\*'. The query editor contains the following SQL code:

```
1 • SELECT DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month,  
2       COUNT(DISTINCT CustomerID) AS UniqueCustomers,  
3       SUM(Quantity * UnitPrice) AS TotalSales  
4 FROM datamini  
5 GROUP BY Month  
6 ORDER BY Month;  
7
```

Below the query editor, the 'Result Grid' is displayed. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are shown in a table with three columns: 'Month', 'UniqueCustomers', and 'TotalSales'. The first row shows 'NULL' for the month, 229 unique customers, and a total sales value of 113902.76999999864.

Month	UniqueCustomers	TotalSales
NULL	229	113902.76999999864

On the right side of the interface, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom right, a 'Read Only' status indicator is visible.

## 5. Time-based Analysis

The screenshot shows a SQL IDE with multiple tabs. The active tab is 'SQL File 12\*'. The query editor contains the following SQL code:

```
1 • SELECT a.StockCode AS Product1, b.StockCode AS Product2, COUNT(*) AS Frequency
2 FROM dataminig a
3 JOIN dataminig b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode < b.StockCode
4 GROUP BY Product1, Product2
5 ORDER BY Frequency DESC;
6
```

Below the query editor is the 'Result Grid' tab, which displays the results of the query. The grid has four columns: Product1, Product2, and Frequency. The results are as follows:

Product1	Product2	Frequency
22632	22633	26
84029E	84029G	21
84029E	85123A	20
84029G	85123A	20
21730	85123A	18
71053	85123A	18
22752	85123A	17
21730	71053	17
21730	84029G	17
22752	84029G	17
21730	22752	17

The IDE also features a toolbar with various icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field. On the right side, there is a vertical toolbar with icons for 'Result Grid', 'Form Editor', and 'Field Types'. The status bar at the bottom indicates 'Result 1' and 'Read Only'.