



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.





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.**

The screenshot shows the MySQL Workbench interface. The query editor at the top contains the following SQL query:

```
1 • SELECT * FROM project2.datamining;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query. The results are shown in a table with the following columns: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country. The table contains 7 rows of data.

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.

The screenshot shows the MySQL Workbench interface. The query editor at the top contains the following SQL query:

```
1 • SELECT CustomerID, SUM(Quantity * UnitPrice) AS total_order_value
2 FROM datamining
3 GROUP BY CustomerID;
```

Below the query editor, the 'Result Grid' tab is active, displaying the results of the query. The results are shown in a table with the following columns: CustomerID and total_order_value. The table contains 7 rows of data.

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.

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

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

The results grid displays the following data:

CustomerID	unique_products_purchased
12347	31
12370	82
12386	8
12395	12
12427	10
12429	20
12431	14
12433	79
17441	11

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.

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

```
1 SELECT CustomerID
2 FROM (
3     SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_count
4     FROM datamining
5     GROUP BY CustomerID
6 ) AS purchase_counts
7 WHERE purchase_count = 1;
```

The results grid displays the following data:

CustomerID
12347
12370
12386
12395
12427
12429
12431
12441
17477



5

- 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.

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

```
1 • SELECT od1.StockCode AS product1, od2.StockCode AS product2, COUNT(*) AS purchase_count
2 FROM datamining od1
3 JOIN datamining od2 ON od1.InvoiceNo = od2.InvoiceNo AND od1.StockCode < od2.StockCode
4 GROUP BY od1.StockCode, od2.StockCode
5 ORDER BY purchase_count DESC
6 LIMIT 10;
```

The results are displayed in a table with the following data:

product1	product2	purchase_count
22726	22727	67
22086	22910	65
22632	22633	48
84029E	84029G	46
22865	22866	45
22727	22730	44
21733	85123A	44
22745	22748	43
22469	22470	43

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





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.

```
1 • SELECT CustomerID,
2     CASE
3     WHEN purchase_count <= 1 THEN 'Low Frequency'
4     WHEN purchase_count <= 5 THEN 'Medium Frequency'
5     ELSE 'High Frequency'
6     END AS purchase_frequency
7 FROM (
8     SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS purchase_count
9     FROM datamining
10    GROUP BY CustomerID
11 ) AS purchase_counts;
12
```

CustomerID	purchase_frequency
12347	Low Frequency
12370	Low Frequency
12386	Low Frequency
12395	Low Frequency
12427	Low Frequency
12429	Low Frequency

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.

```
1 • SELECT Country, AVG(Quantity * UnitPrice) AS average_order_value
2 FROM datamining
3 GROUP BY Country;
4
```

Country	average_order_value
United Kingdom	19.99422179404923
France	25.92366197183097
Australia	28.052272727272726
Netherlands	96.30000000000001
Germany	27.878685121107267
Norway	25.76272108843537
EIRE	27.394558139534883
Switzerland	50.566666666666666
Spain	25.780461538461534
Poland	31.02



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.

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

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

CustomerID
17850
13047
12583
13748
15100
15291
14688
17809
15311
16098

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.

```
1 • SELECT od1.StockCode AS product1, od2.StockCode AS product2, COUNT(*) AS purchase_count
2 FROM datamining od1
3 JOIN datamining od2 ON od1.InvoiceNo = od2.InvoiceNo AND od1.StockCode < od2.StockCode
4 GROUP BY od1.StockCode, od2.StockCode
5 ORDER BY purchase_count DESC
6 LIMIT 10;
7
```

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

product1	product2	purchase_count
22726	22727	67
22086	22910	65
22632	22633	48
84029E	84029G	46
22865	22866	45
22727	22730	44
21733	85123A	44
22745	22748	43
22469	22470	43
22632	22866	42



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.

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



END