

Project : DATA-MINING

Name : Hasnain Asad

Roll No:GIL-DASI-137

Section : 3

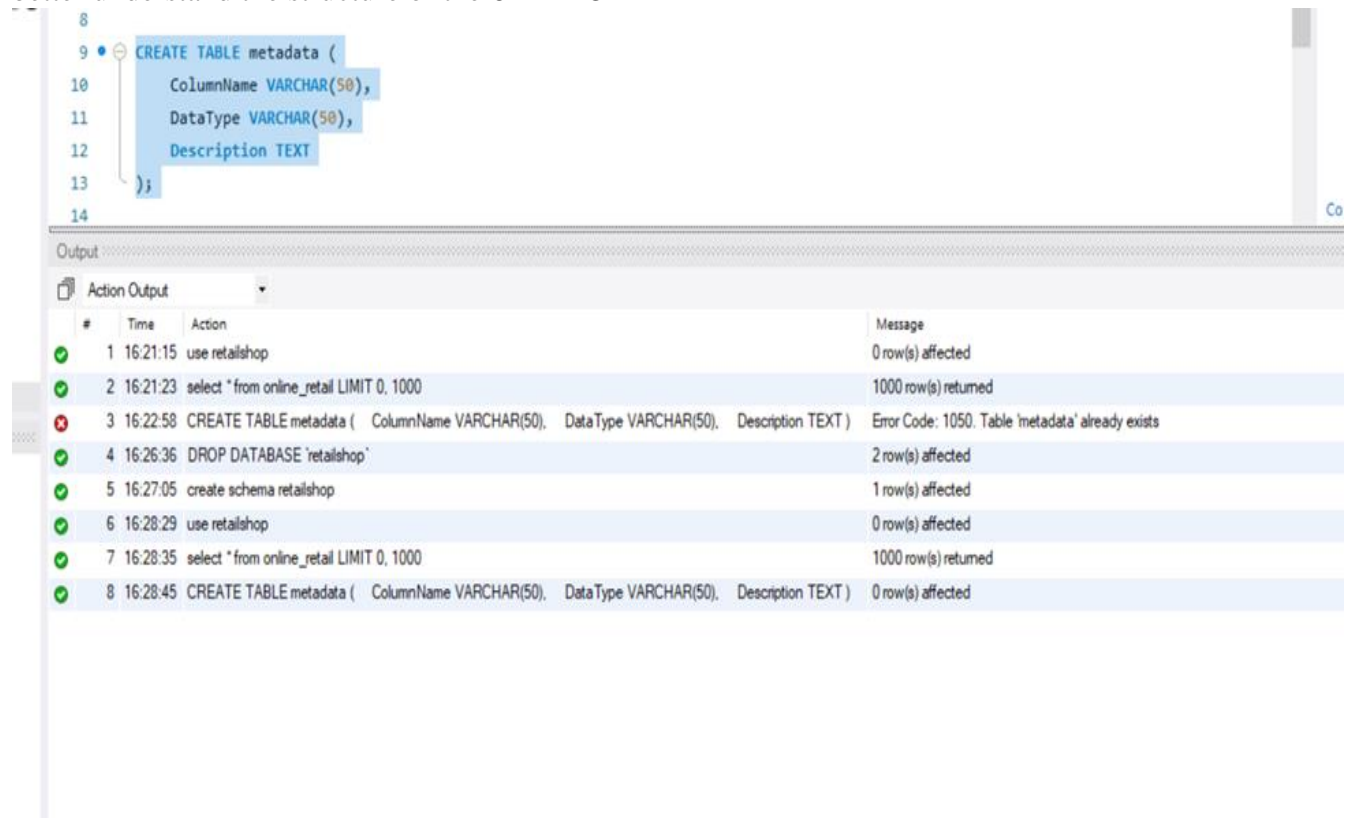
✓ Retail Shop Database Project Report

■ Introduction

The objective of this project is to produce insightful business analysis using the dataset of an online retail store. The dataset includes a variety of parameters, including transaction amounts, customer and product details, invoice numbers, and other data. Our goal is to extract valuable data from SQL queries, including purchase patterns, consumer behavior, product affinity, and sales trends.

■ 2. Schema Setup

To start, the `retailshop` schema was created and they are followed by selecting the working database using the `USE retailshop;` command. A metadata table was also created to maintain information about the columns and their respective data types. This metadata helps to better understand the structure of the `online`



```
8
9 CREATE TABLE metadata (
10     ColumnName VARCHAR(50),
11     DataType VARCHAR(50),
12     Description TEXT
13 );
14
```

Output

#	Time	Action	Message
✓ 1	16:21:15	use retailshop	0 row(s) affected
✓ 2	16:21:23	select * from online_retail LIMIT 0, 1000	1000 row(s) returned
✗ 3	16:22:58	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	Error Code: 1050. Table 'metadata' already exists
✓ 4	16:26:36	DROP DATABASE 'retailshop'	2 row(s) affected
✓ 5	16:27:05	create schema retailshop	1 row(s) affected
✓ 6	16:28:29	use retailshop	0 row(s) affected
✓ 7	16:28:35	select * from online_retail LIMIT 0, 1000	1000 row(s) returned
✓ 8	16:28:45	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	0 row(s) affected

retail table.

```
6 • select * from online_retail;
7
```

The screenshot displays a database management interface. At the top, a query editor shows the SQL statement `select * from online_retail;`. Below the editor, a 'Result Grid' tab is active, showing a table with 8 columns: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country. The table contains 15 rows of data, with the second row (InvoiceNo 536365, StockCode 71053) highlighted. To the right of the grid, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. Below the grid, an 'Output' section shows a log of actions. The first action is 'use retailshop' at 16:21:15, which affected 0 rows. The second action is 'select * from online_retail LIMIT 0, 1000' at 16:21:23, which returned 1000 rows.

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
536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850	United Kingdom
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047	United Kingdom
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010 8:34	2.1	13047	United Kingdom
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010 8:34	2.1	13047	United Kingdom
536367	22740	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:34	7.65	13047	United Kingdom

online_retail 1 x

Output

#	Time	Action	Message
1	16:21:15	use retailshop	0 row(s) affected
2	16:21:23	select * from online_retail LIMIT 0, 1000	1000 row(s) returned

3. Query Execution and Results

○ Distribution of Order Values Across All Customers

In order to determine which customers spend the most money, this query totals the purchases made by each and sorts the results in decreasing order.

```

26
27 -- Distribution of Order Values Across All Customers
28 • SELECT
29     CustomerID,
30     SUM(Quantity * UnitPrice) AS TotalOrderValue
31 FROM online_retail
32 GROUP BY CustomerID
33 ORDER BY TotalOrderValue DESC;
34

```

CustomerID	TotalOrderValue
16029	3702.12
16210	2474.7399999999993
12433	1919.1400000000008
17511	1825.74
17850	1499.3399999999999

Result 3 x

Output

#	Time	Action	Message
2	16:21:23	select * from online_retail LIMIT 0, 1000	1000 row(s) returned
3	16:22:58	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	Error Code: 1050. Table 'metadata' already exists
4	16:26:36	DROP DATABASE 'retailshop'	2 row(s) affected
5	16:27:05	create schema retailshop	1 row(s) affected
6	16:28:29	use retailshop	0 row(s) affected
7	16:28:35	select * from online_retail LIMIT 0, 1000	1000 row(s) returned
8	16:28:45	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	0 row(s) affected
9	16:33:41	INSERT INTO metadata (ColumnName, DataType, Description) VALUES ('invoice_no', 'VARCHAR', 'The invoice...')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
10	16:33:49	SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM online_retail GROUP BY CustomerID	61 row(s) returned

The query calculates the total value of all purchases by multiplying the `Quantity` and `UnitPrice` for each transaction and grouping them by `CustomerID`.

- Unique Products Purchased by Each Customer

Customers that have only made a single unique purchase are identified by this query.

```

35 -- How many unique products has each customer purchased?
36 • SELECT
37     CustomerID
38 FROM online_retail
39 GROUP BY CustomerID
40 HAVING COUNT(DISTINCT InvoiceNo) = 1;
41
42 -- Customers Who Have Only Made a Single Purchase

```

Result Grid

CustomerID
12431
12433
12583
12662
12748

online_retail 4

Output

#	Time	Action	Message
3	16:22:58	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	Error Code: 1050. Table 'metadata' already exists
4	16:26:36	DROP DATABASE 'retailshop'	2 row(s) affected
5	16:27:05	create schema retailshop	1 row(s) affected
6	16:28:29	use retailshop	0 row(s) affected
7	16:28:35	select * from online_retail LIMIT 0, 1000	1000 row(s) returned
8	16:28:45	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	0 row(s) affected
9	16:33:41	INSERT INTO metadata (ColumnName, DataType, Description) VALUES ('Invoice_no', 'VARCHAR', 'The Invo...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
10	16:33:49	SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM online_retail GROUP BY Cu...	61 row(s) returned
11	16:34:34	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned

The HAVING clause offers insight into one-time buyers by removing clients who have made many separate purchases.

- Customers Who Have Made a Single Purchase

The query retrieves customers who have made only a single purchase based on the `InvoiceNo` field.

```

42
43 -- Customers Who Have Only Made a Single Purchase
44 • SELECT
45     CustomerID
46 FROM online_retail
47 GROUP BY CustomerID
48 HAVING COUNT(DISTINCT InvoiceNo) = 1;
49
50 -- Most Commonly Purchased Products Together

```

Result Grid

CustomerID
12431
12433
12583
12662
12748

online_retail 5

Output

Action Output

#	Time	Action	Message
4	16:26:36	DROP DATABASE 'retailshop'	2 row(s) affected
5	16:27:05	create schema retailshop	1 row(s) affected
6	16:28:29	use retailshop	0 row(s) affected
7	16:28:35	select * from online_retail LIMIT 0, 1000	1000 row(s) returned
8	16:28:45	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	0 row(s) affected
9	16:33:41	INSERT INTO metadata (ColumnName, DataType, Description) VALUES ('invoice_no', 'VARCHAR', 'The invoice...')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
10	16:33:49	SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM online_retail GROUP BY Cu...	61 row(s) returned
11	16:34:34	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
12	16:35:16	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned

○ Most Commonly Purchased Products Together

This query joins the same InvoiceNo and counts product pairings to investigate the products that are frequently purchased together.

```

50
51 -- Most Commonly Purchased Products Together
52 SELECT
53     a.StockCode AS ProductA,
54     b.StockCode AS ProductB,
55     COUNT(*) AS TimesPurchasedTogether
56 FROM online_retail a
57 JOIN online_retail b ON a.InvoiceNo = b.InvoiceNo AND a.StockCode < b.StockCode
58 GROUP BY a.StockCode, b.StockCode
59 ORDER BY TimesPurchasedTogether DESC
60 LIMIT 10;
61

```

ProductA	ProductB	TimesPurchasedTogether
21448	22749	15
21448	22273	15
21448	21738	15
21448	22077	15
21448	22243	15
21448	85049E	15

Result 6 x

Output

#	Time	Action	Message
7	16:28:35	select * from online_retail LIMIT 0, 1000	1000 row(s) returned
8	16:28:45	CREATE TABLE metadata (ColumnName VARCHAR(50), DataType VARCHAR(50), Description TEXT)	0 row(s) affected
9	16:33:41	INSERT INTO metadata (ColumnName, DataType, Description) VALUES ('Invoice_no', 'VARCHAR', 'The invoice...')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
10	16:33:49	SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM online_retail GROUP BY Cu...	61 row(s) returned
11	16:34:34	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
12	16:35:16	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
13	16:36:30	SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, COUNT(*) AS TimesPurchasedToget...	10 row(s) returned

Giving insights on product affinity and cross-selling opportunities, the data display the top 10 product pairs most frequently purchased together.

○ Customer Segmentation by Purchase Frequency

The number of purchases a customer has made divides them into segments, which is helpful for focused marketing.

```

63 -- Advanced Queries
64
65 -- Customer Segmentation by Purchase Frequency
66 • SELECT
67     CustomerID,
68     COUNT(DISTINCT InvoiceNo) AS PurchaseFrequency,
69     CASE
70         WHEN COUNT(DISTINCT InvoiceNo) > 20 THEN 'High'
71         WHEN COUNT(DISTINCT InvoiceNo) BETWEEN 5 AND 20 THEN 'Medium'
72         ELSE 'Low'
73     END AS FrequencySegment
74 FROM online_retail
75 GROUP BY CustomerID;
76
77 -- Average Order Value by Country

```

CustomerID	PurchaseFrequency	FrequencySegment
12431	1	Low
12433	1	Low
12583	1	Low
12662	1	Low
12748	1	Low

Result 7 x

Output

#	Time	Action	Message
10	16:33:49	SELECT CustomerID, SUM(Quantity * UnitPrice) AS TotalOrderValue FROM online_retail GROUP BY Cu...	61 row(s) returned
11	16:34:34	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
12	16:35:16	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
13	16:36:30	SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, COUNT(*) AS TimesPurchasedToget...	10 row(s) returned
14	16:37:41	SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS PurchaseFrequency, CASE WHEN CO...	61 row(s) returned

The CASE statement categorizes customers into high, medium, and low-frequency segments.

- Average Order Value by Country

In order to determine the locations of higher-value purchases, this query computes the average order value across several nations.


```

78 -- Average Order Value by Country
79
80 • SELECT
81     Country,
82     AVG(Quantity * UnitPrice) AS AvgOrderValue
83 FROM online_retail
84 GROUP BY Country
85 ORDER BY AvgOrderValue DESC;
86
87 -- Customer Churn Analysis (Identify Customers Who Haven't Made a Purchase in the Last 6 Months)
88
89 • SELECT

```

Result Grid

Country	AvgOrderValue
Netherlands	96.30000000000001
France	42.793
Norway	26.2895890410959
Australia	25.589285714285715
United Kingdom	22.029572147651113

Output

Action Output

#	Time	Action	Message
✓ 11	16:34:34	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
✓ 12	16:35:16	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
✓ 13	16:36:30	SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, COUNT(*) AS TimesPurchasedToget...	10 row(s) returned
✓ 14	16:37:41	SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS PurchaseFrequency, CASE WHEN CO...	61 row(s) returned
✓ 15	16:38:44	SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue FROM online_retail GROUP BY Country ...	6 row(s) returned

○ Customer Churn Analysis

For the purpose of creating retention strategies, it is imperative to identify consumers who have not made a purchase within the last six months as part of a churn analysis.

```

86
87 -- Customer Churn Analysis (Identify Customers Who Haven't Made a Purchase in the Last 6 Months)
88
89 • SELECT
90     CustomerID
91 FROM online_retail
92 WHERE InvoiceDate < DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
93 GROUP BY CustomerID
94 HAVING COUNT(InvoiceNo) > 0;
95
96
97
98 -- Product Affinity Analysis

```

Result Grid

CustomerID
17850
13047
12583
13748
15100

Output

Action Output

#	Time	Action	Message
✓ 12	16:35:16	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
✓ 13	16:36:30	SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, COUNT(*) AS TimesPurchasedToget...	10 row(s) returned
✓ 14	16:37:41	SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS PurchaseFrequency, CASE WHEN CO...	61 row(s) returned
✓ 15	16:38:44	SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue FROM online_retail GROUP BY Country ...	6 row(s) returned
✓ 16	16:39:37	SELECT CustomerID FROM online_retail WHERE InvoiceDate < DATE_SUB(CURDATE(), INTERVAL 6 M...	61 row(s) returned

○ Product Affinity Analysis

This query helps identify the most popular products based on the number of purchases.

The screenshot displays a SQL IDE interface. The top pane shows a SQL query for Product Affinity Analysis. The bottom pane shows the results of the query in a table format.

```
96
97 -- Product Affinity Analysis
98
99 • SELECT
100     StockCode,
101     COUNT(*) AS PurchaseCount
102 FROM online_retail
103 GROUP BY StockCode
104 ORDER BY PurchaseCount DESC
105 LIMIT 10;
```

The results table shows the top 10 products by purchase count:

StockCode	PurchaseCount
22632	14
22866	13
22961	11
22865	10
22633	10

The bottom pane shows the output of the SQL IDE, including a list of actions and their messages:

#	Time	Action	Message
13	16:36:30	SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, COUNT(*) AS TimesPurchasedToget...	10 row(s) returned
14	16:37:41	SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS PurchaseFrequency, CASE WHEN CO...	61 row(s) returned
15	16:38:44	SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue FROM online_retail GROUP BY Country ...	6 row(s) returned
16	16:39:37	SELECT CustomerID FROM online_retail WHERE InvoiceDate < DATE_SUB(CURDATE(), INTERVAL 6 M...	61 row(s) returned
17	16:40:18	SELECT StockCode, COUNT(*) AS PurchaseCount FROM online_retail GROUP BY StockCode ORDER...	10 row(s) returned

○ Time-Based Analysis (Monthly Sales Patterns)

This query aggregates sales on a monthly basis in order to detect patterns in sales over time.

```

106
107
108
109 -- Time-based Analysis (Monthly Sales Patterns)
110
111 • SELECT
112     DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month,
113     SUM(Quantity * UnitPrice) AS TotalSales
114 FROM online_retail
115 GROUP BY DATE_FORMAT(InvoiceDate, '%Y-%m')
116 ORDER BY Month;
117

```

Month	TotalSales
MAR	29846.580000000133

#	Time	Action	Message
12	16:35:16	SELECT CustomerID FROM online_retail GROUP BY CustomerID HAVING COUNT(DISTINCT InvoiceNo) ...	56 row(s) returned
13	16:36:30	SELECT a.StockCode AS ProductA, b.StockCode AS ProductB, COUNT(*) AS TimesPurchasedToget...	10 row(s) returned
14	16:37:41	SELECT CustomerID, COUNT(DISTINCT InvoiceNo) AS PurchaseFrequency, CASE WHEN CO...	61 row(s) returned
15	16:38:44	SELECT Country, AVG(Quantity * UnitPrice) AS AvgOrderValue FROM online_retail GROUP BY Country ...	6 row(s) returned
16	16:39:37	SELECT CustomerID FROM online_retail WHERE InvoiceDate < DATE_SUB(CURDATE(), INTERVAL 6 M...	61 row(s) returned
17	16:40:18	SELECT StockCode, COUNT(*) AS PurchaseCount FROM online_retail GROUP BY StockCode ORDER...	10 row(s) returned
18	16:41:00	SELECT DATE_FORMAT(InvoiceDate, '%Y-%m') AS Month, SUM(Quantity * UnitPrice) AS TotalSales F...	1 row(s) returned

Forecasting and spotting seasonality might be aided by the monthly sales performance information this query offers.

▪ Conclusion

This research used SQL queries on the online_retail dataset to provide important insights into product trends, customer behavior, and sales patterns. We classified customers by frequency of purchases, found high-spending customers, and discovered common product combinations. Monthly sales patterns provided a clear picture of peak times, and customer retention prospects were identified via churn analysis. These insights have the potential to enhance product offers, marketing tactics, and overall business performance. Deeper analytics has the potential to improve customer engagement and decision-making processes in the future.

Project Link : <https://github.com/hasnainasad1/Data-mining-Project>

Github Link: <https://github.com/hasnainasad1>