Case Study - Retail Analytics & Ecommerce

Challenge- 13

Occasional Customers

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Write a SQL query that describes the number of transaction along with the total amount spent by each customer, which will help us understand the customers who are occasional customers or have low purchase frequency in the company.

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table must have number of transactions less than or equal to 2 and corresponding total amount spent on those transactions by related customers.*
- *Return the result table of "NumberOfTransactions" in ascending order and "TotalSpent" in descending order.*

Output format:

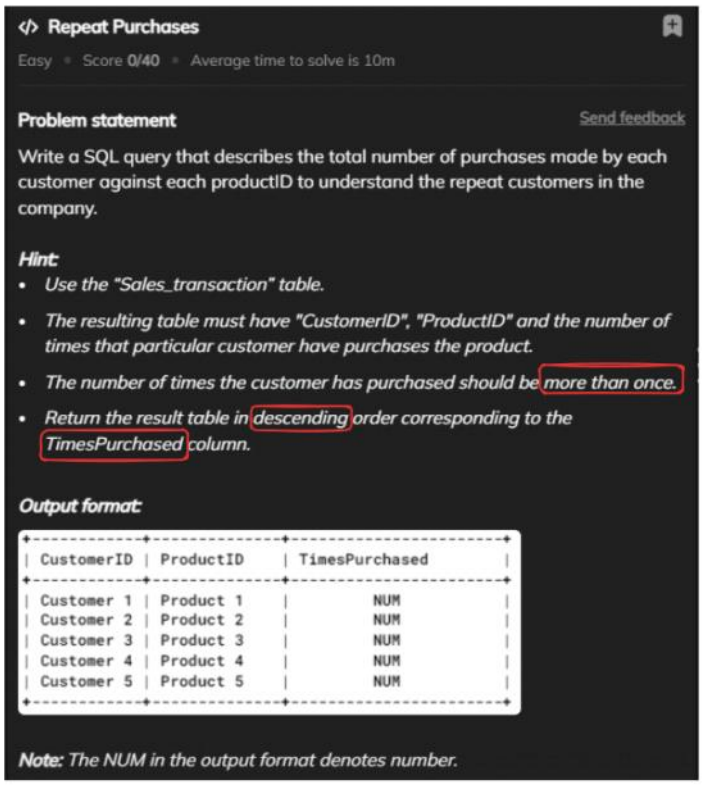
CustomerID	NumberOfTransactions	TotalSpent
Customer 1	NUM	DECINUM
Customer 2	NUM	DECINUM
Customer 3	NUM	DECINUM
Customer 4	NUM	DECINUM
Customer 5	NUM	DECINUM

Note: *The NUM in the output format denotes a numerical value and DECINUM denotes a decimal value.*

```
-- Challenge 13 : Occassional Customers
USE RetailAnalytics;

SELECT * FROM Sales;

SELECT
    CustomerID,
    COUNT(*) AS NumberOfTransactions,
    ROUND(SUM(QuantityPurchased * Price),0) AS TotalSpent
FROM Sales
GROUP BY CustomerID
HAVING NumberOfTransactions <= 2
ORDER BY NumberOfTransactions ASC , TotalSpent DESC;
```

CustomerID	NumberOfTransactions	TotalSpent
150	1	44
766	1	29
219	1	22
555	1	22
255	1	14
464	1	11
534	2	566
591	2	539
347	2	530
245	2	523
241	2	492
412	2	438
652	2	434
275	2	431
583	2	431

Challenge- 14



Repeat Purchases

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Write a SQL query that describes the total number of purchases made by each customer against each productID to understand the repeat customers in the company.

Hint:

- *Use the "Sales_transaction" table.*
- *The resulting table must have "CustomerID", "ProductID" and the number of times that particular customer have purchases the product.*
- *The number of times the customer has purchased should be more than once.*
- *Return the result table in descending order corresponding to the TimesPurchased column.*

Output format:

CustomerID	ProductID	TimesPurchased
Customer 1	Product 1	NUM
Customer 2	Product 2	NUM
Customer 3	Product 3	NUM
Customer 4	Product 4	NUM
Customer 5	Product 5	NUM

Note: *The NUM in the output format denotes number.*

```sql
-- Challenge 14 : Repeat Purchases
SELECT * FROM Sales;

SELECT
    CustomerID,
    ProductID,
    COUNT(*) AS TimesPurchased
FROM Sales
GROUP BY CustomerID,ProductID
HAVING TimesPurchased > 1
ORDER BY TimesPurchased DESC; -- 70 row(s) returned
```

CustomerID	ProductID	TimesPurchased
685	192	3
467	181	2
215	13	2
492	74	2
242	172	2
822	165	2
296	196	2
613	44	2
225	75	2
710	156	2
2	65	2
852	66	2

Challenge- 15

Loyalty Indicators

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Write a SQL query that describes the duration between the first and the last purchase of the customer in that particular company to understand the loyalty of the customer.

Hints:

- *Use the "Sales_transaction" table.*
- *The DATE column will be majorly in use in the question and the TransactionDate column in Sales_transaction is in text format. Thus, the format of the TransactionDate column should be changed.*
- *The resulting table must have the first date of purchase, the last date of purchase and the difference between the first and the last date of purchase.*
- *The difference between the first and the last date of purchase should be more than 0.*
- *Return the table in descending order corresponding to DaysBetweenPurchases.*

Output Format:

CustomerID	FirstPurchase	LastPurchase	DaysBetweenPurchases
Customer 1	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 2	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 3	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 4	YYYY-MM-DD	YYYY-MM-DD	NUM
Customer 5	YYYY-MM-DD	YYYY-MM-DD	NUM

Note: *The YYYY/MM/DD in the "FirstPurchase" and "LastPurchase" should be in this date format and the NUM in "DaysBetweenPurchases" is in integer format.*

Field	Type	Null	Key	Default	Extra
TransactionID	int	YES		NULL	
CustomerID	int	YES		NULL	
ProductID	int	YES		NULL	
QuantityPurcha...	int	YES		NULL	
TransactionDate	text	YES		NULL	
Price	double	YES		NULL	
TransactionDat...	date	YES		NULL	

```
-- Challenge 15 : Loyalty Indicators
```

```
SELECT * FROM Sales;
DESC Sales;

WITH TransactionDate AS(
    SELECT
        CustomerID,
    STR_TO_DATE(TransactionDate , '%d/%m/%y') AS TransactionDate
    FROM Sales
)
SELECT * FROM TransactionDate;
```

CustomerID	TransactionDate
103	2023-01-01
436	2023-01-01
861	2023-01-01
271	2023-01-01
107	2023-01-01
72	2023-01-01
701	2023-01-01
21	2023-01-01
615	2023-01-01
122	2023-01-01
467	2023-01-01
215	2023-01-01

```sql
-- Challenge 15 : Loyalty Indicators

SELECT * FROM Sales;
DESC Sales;

WITH TransactionDate AS(
    SELECT
        CustomerID,
        STR_TO_DATE(TransactionDate , '%d/%m/%y') AS TransactionDate
    FROM Sales
)
SELECT
    CustomerID,
    MIN(TransactionDate) AS FirstPurchase,
    MAX(TransactionDate) AS LastPurchase,
    DATEDIFF(MAX(TransactionDate),MIN(TransactionDate)) AS DaysBetweenPurchases
FROM TransactionDate
GROUP BY CustomerID
HAVING DaysBetweenPurchases > 0
ORDER BY DaysBetweenPurchases DESC;
```

CustomerID	FirstPurchase	LastPurchase	DaysBetweenPurchases
215	2023-01-01	2023-07-28	208
414	2023-01-02	2023-07-26	205
664	2023-01-01	2023-07-24	204
701	2023-01-01	2023-07-23	203
277	2023-01-02	2023-07-24	203
22	2023-01-02	2023-07-24	203
976	2023-01-02	2023-07-24	203
647	2023-01-03	2023-07-25	203
162	2023-01-05	2023-07-27	203
806	2023-01-02	2023-07-23	202
511	2023-01-02	2023-07-23	202
703	2023-01-05	2023-07-26	202
188	2023-01-06	2023-07-27	202
380	2023-01-06	2023-07-27	202
566	2023-01-04	2023-07-24	201
748	2023-01-02	2023-07-21	200

Challenge- 16

Customer Segmentation

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Write an SQL query that segments customers based on the total quantity of products they have purchased. Also, count the number of customers in each segment which will help us target a particular segment for marketing.

Hint:

- *Use the customer_profiles and sales_transaction tables.*
- *Create a separate table named customer_segment and create the segments on the total quantity of the purchased products.*
- *To segment customers based on their purchasing behavior for targeted marketing campaigns. Create Customer segments on the following criteria-*

Total Quantity of Products Purchased	Customer Segment
1-10	Low
11-30	Mid
>30	High

- *The resulting table should count the number of customers in different customer segments.*
- *Return the result table in any order.*

Output Format:

CustomerSegment	COUNT(*)
Med	NUM
Low	NUM
High	NUM

Customer_id	TotalQty

-- Challenge 16 : Customer Segmentation

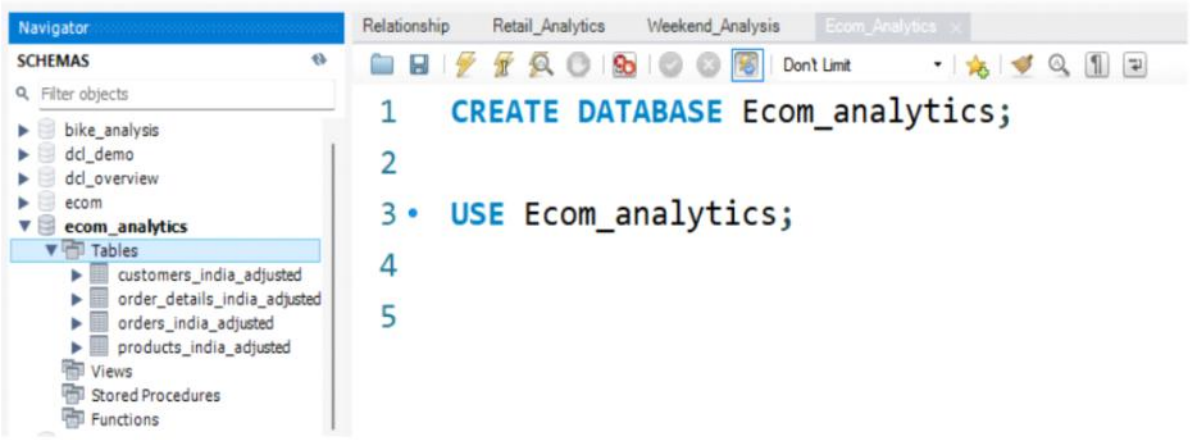
```
SELECT * FROM Customers;
SELECT * FROM Sales;

CREATE TABLE customer_segment AS
    SELECT
        CustomerID,
    CASE
        WHEN TotalQuantity BETWEEN 1 AND 10 THEN 'Low'
        WHEN TotalQuantity BETWEEN 11 AND 30 THEN 'Med'
        WHEN TotalQuantity > 30 THEN 'High'
        ELSE 'None'
    END AS CustomerSegment
    FROM (
        SELECT
            c.CustomerID,
            SUM(s.QuantityPurchased) AS TotalQuantity
        FROM Customers c
        JOIN Sales s
        ON c.CustomerID = s.CustomerID
        GROUP BY 1
    ) AS customer_totals;

SELECT * FROM Customer_Segment;
```

CustomerID	CustomerSegment
103	Med
436	Med
861	Med
271	Low
107	Low
72	Low
701	Med
21	Med
615	Low
122	Med
467	Med
215	Med
331	Low
459	Med
88	Med
373	Low
100	Low

```sql
SELECT
    CustomerSegment,
    COUNT(*)
FROM Customer_Segment
GROUP BY CustomerSegment;
```

CustomerSegment	COUNT(*)
Med	559
Low	423
High	7

Ecommerce Company Case Study




```sql
5 • DESC Customers_india_adjusted;
```

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

Field	Type	Null	Key	Default	Extra
customer_id	int	YES		NULL	
name	text	YES		NULL	
location	text	YES		NULL	

```sql
6 • DESC Order_Details_india_adjusted;
7
```

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

Field	Type	Null	Key	Default	Extra
order_id	int	YES		NULL	
product_id	int	YES		NULL	
quantity	int	YES		NULL	
price_per_unit	int	YES		NULL	

```sql
7 • DESC Orders_india_adjusted;
8
```

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

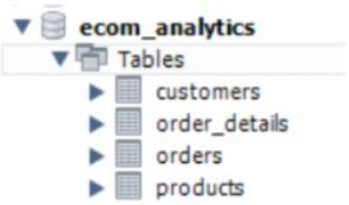
Field	Type	Null	Key	Default	Extra
order_id	int	YES		NULL	
order_date	text	YES		NULL	
customer_id	int	YES		NULL	
total_amount	int	YES		NULL	

```
8 • DESC Products_india_adjusted;
9
```

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

Field	Type	Null	Key	Default	Extra
product_id	int	YES		NULL	
name	text	YES		NULL	
category	text	YES		NULL	
price	int	YES		NULL	

```
ALTER TABLE Customers_india_adjusted
RENAME TO Customers;

ALTER TABLE Order_Details_india_adjusted
RENAME TO Order_Details;

ALTER TABLE Orders_india_adjusted
RENAME TO Orders;

ALTER TABLE Products_india_adjusted
RENAME TO Products;
```



Customers

customer_id	name	location
1	Ivana Chander	Delhi
2	Charvi Kibe	Chennai
3	Divij Chaudry	Chennai
4	Charvi Balay	Pune
5	Diya Arya	Pune
6	Dhruv Cherian	Chennai
7	Myra Dubey	Chennai
8	Advika Wable	Delhi
9	Aarna Samra	Hyderabad
10	Ahana Ray	Ahmedabad
11	Tanya Baria	Lucknow
12	Kismat Sangha	Kolkata
13	Lakshit Walia	Ahmedabad
14	Jayant Yohan...	Lucknow
15	Jivika Tiwari	Delhi

Products

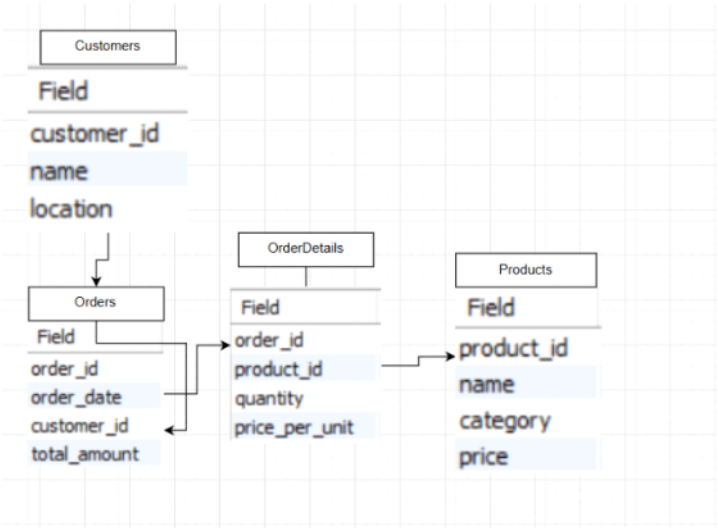
product_id	name	category	price
1	Smartphone 6"	Electronics	15000
2	Laptop 15" Pro	Electronics	60000
3	Bluetooth Headphones	Electronics	8000
4	E-Book Reader	Electronics	12000
5	Smartwatch Fitness Tracker	Wearable Tech	5000
6	Portable Bluetooth Speaker	Electronics	7000
7	Digital SLR Camera	Photography	40000
8	Wireless Earbuds	Wearable Tech	3000

Order_Details

order_id	product_id	quantity	price_per_unit
1	1	3	15000
2	3	2	8000
3	2	1	60000
3	7	2	40000
3	7	3	40000
4	4	1	12000
4	1	1	15000
5	5	1	5000
5	4	3	12000
5	2	3	60000
6	7	1	40000
6	7	2	40000
7	1	2	15000
8	7	1	40000
8	4	1	12000

Orders

order_id	order_date	customer_id	total_amount
1	2023-09-27	67	45000
2	2023-11-19	98	16000
3	2023-12-20	46	260000
4	2023-04-29	83	27000
5	2023-04-05	99	221000
6	2023-05-29	59	120000
7	2023-04-26	68	30000
8	2023-08-06	24	64000
9	2023-12-17	61	250000
10	2023-10-04	38	53000
11	2023-10-16	95	129000
12	2023-07-30	78	15000
13	2023-12-15	24	158000
14	2023-04-07	17	72000
15	2023-05-09	58	21000



Challenge 1

Analyze the Data

Easy • Score 40/40 • Average time to solve is 10m

Problem statement Send feedback

You can analyze all the tables by describing their contents.

Task: Describe the Tables:

1. Customers
2. Products
3. Orders
4. OrderDetails

Done ✅

Challenge 2

Market Segmentation Analysis

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Identify the top 3 cities with the highest number of customers to determine key markets for targeted marketing and logistic optimization.

Hint:

- *Use the "Customers" Table.*
- *Return the result table limited to top 3 locations in descending order*

Output Format:

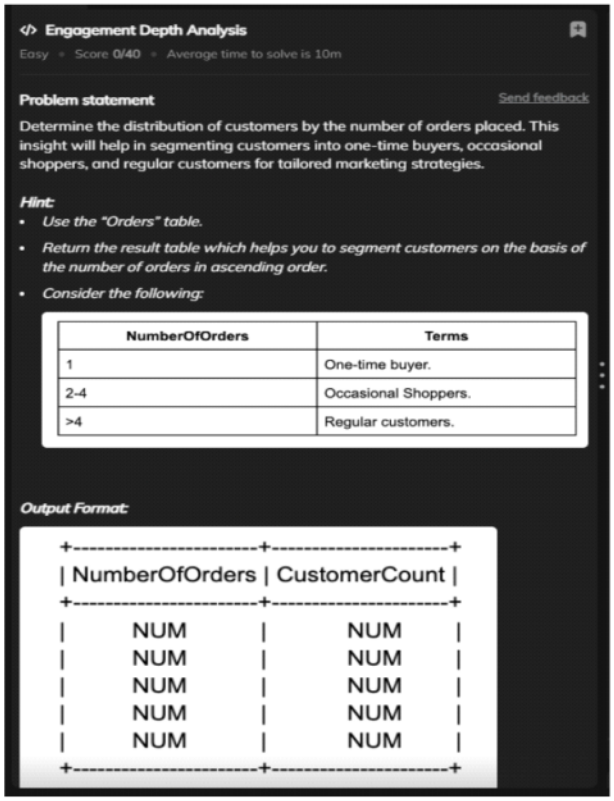
```
+--------------------+-----------------------------+
| location           | number_of_customers |
+--------------------+-----------------------------+
| City 1             |             NUM             |
| City 2             |             NUM             |
| City 3             |             NUM             |
+--------------------+-----------------------------+
```

Note: *NUM in the output format denotes a numerical value*

```sql
-- Challenge 2 : Market Segmentation Analysis

SELECT * FROM Customers;

SELECT
    Location,
    COUNT(Customer_id) AS number_of_customers
FROM Customers
GROUP BY Location
ORDER BY number_of_customers DESC
LIMIT 3;
```

Location	number_of_customers
Delhi	16
Chennai	15
Jaipur	11

Challenge 3



Engagement Depth Analysis

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Determine the distribution of customers by the number of orders placed. This insight will help in segmenting customers into one-time buyers, occasional shoppers, and regular customers for tailored marketing strategies.

Hint:

- *Use the "Orders" table.*
- *Return the result table which helps you to segment customers on the basis of the number of orders in ascending order.*
- *Consider the following:*

NumberOfOrders	Terms
1	One-time buyer.
2-4	Occasional Shoppers.
>4	Regular customers.

Output Format:

NumberOfOrders	CustomerCount
NUM	NUM
NUM	NUM
NUM	NUM
NUM	NUM
NUM	NUM

```sql
SELECT
    Customer_id,
    COUNT(Order_id) AS NumberOfOrders
FROM Orders
GROUP BY Customer_id;
```

Customer_id	NumberOfOrders
67	2
98	3
46	1
83	2
99	2
59	2
68	5
24	4
61	5
38	3
95	3
78	4
17	4
58	2
36	2

Count Group

```sql
-- Challenge 3 - Engagement Depth Analysis

SELECT * FROM Orders;


SELECT
    NumberOfOrders,
    COUNT(*) AS CustomerCount
FROM(
    SELECT
        Customer_id,
        COUNT(Order_id) AS NumberOfOrders
    FROM Orders
    GROUP BY Customer_id
) AS CustomerOrders
GROUP BY NumberOfOrders
ORDER BY NumberOfOrders ASC;
```

NumberOfOrders	CustomerCount
1	26
2	26
3	18
4	6
5	6
6	1
8	1

Challenge 4

Purchase High-Value Products

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Identify products where the average purchase quantity per order is 2 but with a high total revenue, suggesting premium product trends.

Hint:

- *Use "OrderDetails".*
- *Return the result table which includes average quantity and the total revenue in descending order.*

Output format:

Product_Id	AvgQuantity	TotalRevenue
Product 1	NUM	NUM

Note: *NUM in the output format denotes a numerical value.*

```sql
SELECT
    Product_id,
    AVG(quantity) AS AvgQuantity,
    SUM(quantity * price_per_unit) AS TotalRevenue
FROM Order_Details
GROUP BY product_id;
```

Product_id	AvgQuantity	TotalRevenue
1	2.0000	1620000
3	1.9853	1080000
2	1.8806	7560000
7	1.9359	6040000
4	1.9118	1560000
5	1.9833	595000
8	2.0000	390000
6	2.2712	938000

```
-- Challenge 4 : Purchase High-Value Products
```

```
SELECT * FROM Order_Details;

SELECT
    Product_id,
    AVG(quantity) AS AvgQuantity,
    SUM(quantity * price_per_unit) AS TotalRevenue
FROM Order_Details
GROUP BY product_id
HAVING AvgQuantity = 2
ORDER BY TotalRevenue DESC;
```

Product_id	AvgQuantity	TotalRevenue
1	2.0000	1620000
8	2.0000	390000

Challenge 5

Category-wise Customer Reach

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

For each product category, calculate the unique number of customers purchasing from it. This will help understand which categories have wider appeal across the customer base.

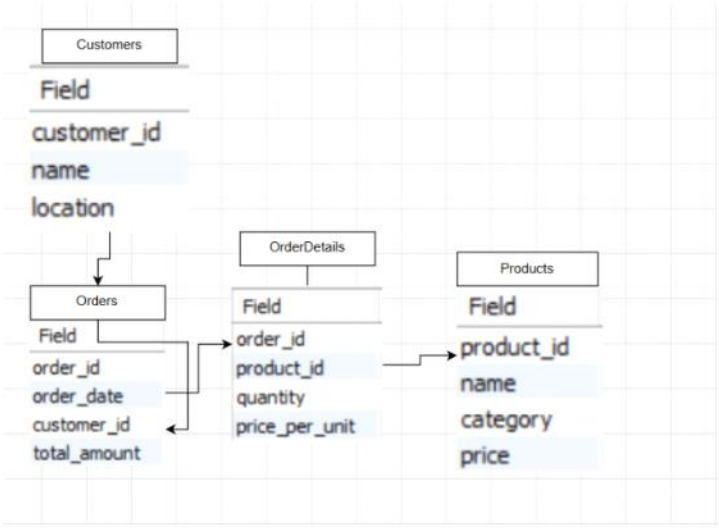
Hint:

- *Use the "Products", "OrderDetails" and "Orders" table.*
- *Return the result table which will help you count the unique number of customers in descending order.*

Output format:

category	unique_customers
Category 1	NUM
Category 2	NUM
Category 3	NUM

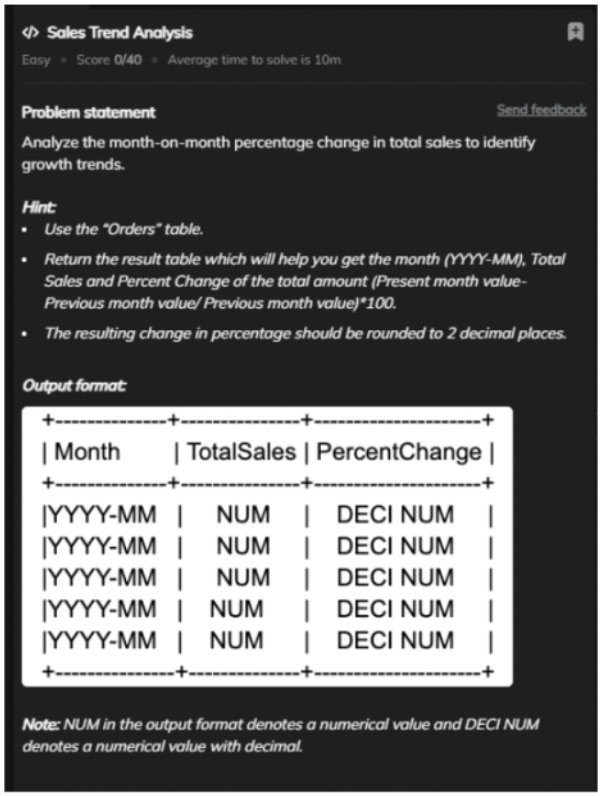
Note: *NUM in the output format denotes a numerical value.*




```
-- Challenge 5 : Category-Wise Customer Reach

SELECT
    p.Category,
    COUNT(DISTINCT o.customer_id) AS unique_customers
FROM Products p
JOIN Order_Details od
ON od.Product_id = p.Product_id
JOIN Orders o
ON o.order_id = od.order_id
GROUP BY p.Category
ORDER BY unique_customers DESC;
```

Category	unique_customers
Electronics	79
Wearable Tech	61
Photography	45

Challenge 6



Sales Trend Analysis

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Analyze the month-on-month percentage change in total sales to identify growth trends.

Hint:

- *Use the "Orders" table.*
- *Return the result table which will help you get the month (YYYY-MM), Total Sales and Percent Change of the total amount (Present month value-Previous month value/ Previous month value)*100.*
- *The resulting change in percentage should be rounded to 2 decimal places.*

Output format:

```
+--------------+---------------+---------------------+
| Month        | TotalSales | PercentChange |
+--------------+---------------+---------------------+
|YYYY-MM  |    NUM    |   DECI NUM    |
|YYYY-MM  |    NUM    |   DECI NUM    |
|YYYY-MM  |    NUM    |   DECI NUM    |
|YYYY-MM  |   NUM     |   DECI NUM    |
|YYYY-MM  |   NUM     |   DECI NUM    |
+--------------+---------------+---------------------+
```

Note: *NUM in the output format denotes a numerical value and DECI NUM denotes a numerical value with decimal.*

```sql
-- Challenge 6 - Sales Trend Analysis
SELECT * FROM Orders;

WITH MonthlySales AS (
    SELECT
        DATE_FORMAT(order_date , '%Y-%m') AS Month,
        SUM(total_amount) AS TotalSales
    FROM Orders
    GROUP BY Month
    ORDER BY Month
)
SELECT * FROM MonthlySales;
```

```sql
-- Challenge 6 - Sales Trend Analysis
SELECT * FROM Orders;

WITH MonthlySales AS (
    SELECT
        DATE_FORMAT(order_date , '%Y-%m') AS Month,
        SUM(total_amount) AS TotalSales
    FROM Orders
    GROUP BY Month
)
SELECT
    Month,
    TotalSales,
    ROUND((TotalSales - LAG(TotalSales) OVER(Order BY Month))
     / (LAG(TotalSales) OVER(Order BY Month)) * 100 , 2) AS PercentChange
FROM MonthlySales;
```

Month	TotalSales	PercentChange
2023-03	789000	NULL
2023-04	1704000	115.97
2023-05	1582000	-7.16
2023-06	1040000	-34.26
2023-07	2568000	146.92
2023-08	1800000	-29.91
2023-09	2927000	62.61
2023-10	1497000	-48.86
2023-11	1151000	-23.11
2023-12	2774000	141.01
2024-01	1555000	-43.94
2024-02	396000	-74.53

Challenge 7

</> Average Order Value Fluctuation

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Examine how the average order value changes month-on-month. Insights can guide pricing and promotional strategies to enhance order value.

Hint:

- *Use the "Orders" Table.*
- *Return the result table which will help you get the month (YYYY-MM), Average order value and Change in the average order value (Present month value- Previous month value).*
- *Both the resulting AvgOrderValue and ChangeInValue column should be rounded to two decimal places, with the final results ordered in descending order by ChangeInValue.*

Output format:

```
+---------------+--------------------+---------------------+
| Month         | AvgOrderValue | ChangeInValue |
+---------------+--------------------+---------------------+
|YYYY-MM   |   DECI NUM   |     DECI NUM    |
|YYYY-MM   |   DECI NUM   |     DECI NUM    |
|YYYY-MM   |   DECI NUM   |     DECI NUM    |
|YYYY-MM   |   DECI NUM   |     DECI NUM    |
|YYYY-MM   |   DECI NUM   |     DECI NUM    |
+---------------+--------------------+---------------------+
```

Note: *DECI NUM in the output format denotes a numerical value with decimal.*

```
-- Challenge 7 - Average Order Value Fluctuation...
```

```sql
SELECT * FROM Orders;

WITH MonthlyOrderValue AS (
    SELECT
        DATE_FORMAT(order_date , '%Y-%m') AS Month,
        ROUND(AVG(total_amount),2) AS AverageOrderValue
    FROM Orders
    GROUP BY Month
)
SELECT
    Month,
    AverageOrderValue,
    ROUND((AverageOrderValue - LAG(AverageOrderValue) OVER(Order BY Month))
     / (LAG(AverageOrderValue) OVER(Order BY Month)) * 100 , 2) AS ChangeInValue
FROM MonthlyOrderValue
ORDER BY ChangeInValue DESC;
```

Month	AverageOrderValue	ChangeInValue
2023-03	60692.31	NULL
2023-04	81142.86	33.70
2023-05	87888.89	8.31
2023-06	104000.00	18.33
2023-07	98769.23	-5.03
2023-08	112500.00	13.90
2023-09	121958.33	8.41
2023-10	83166.67	-31.81
2023-11	95916.67	15.33
2023-12	132095.24	37.72
2024-01	129583.33	-1.90
2024-02	44000.00	-66.05

Challenge 8

Inventory Refresh Rate

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Based on sales data, identify products with the fastest turnover rates, suggesting high demand and the need for frequent restocking.

Hint:

- *Use the "OrderDetails" table.*
- *Return the result table limited to top 5 product according to the SalesFrequency column in descending order.*

Output format:

```
+-------------+---------------------+
| product_id | SalesFrequency |
+-------------+---------------------+
| Product 1  |        NUM          |
| Product 2  |        NUM          |
| Product 3  |        NUM          |
| Product 4  |        NUM          |
| Product 5  |        NUM          |
+-------------+---------------------+
```

Note: *NUM in the output format denotes a numerical value.*

```sql
-- Challenge 8 : Inventory Refresh Rate

SELECT * FROM Order_Details;

SELECT
    Product_id,
    COUNT(order_id) AS SalesFrequency
FROM Order_Details
GROUP BY Product_id
ORDER BY SalesFrequency DESC
LIMIT 5;
```

Product_id	SalesFrequency
7	78
3	68
4	68
2	67
8	65

Challenge 9

Low Engagement Products

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

List products purchased by less than 40% of the customer base, indicating potential mismatches between inventory and customer interest.

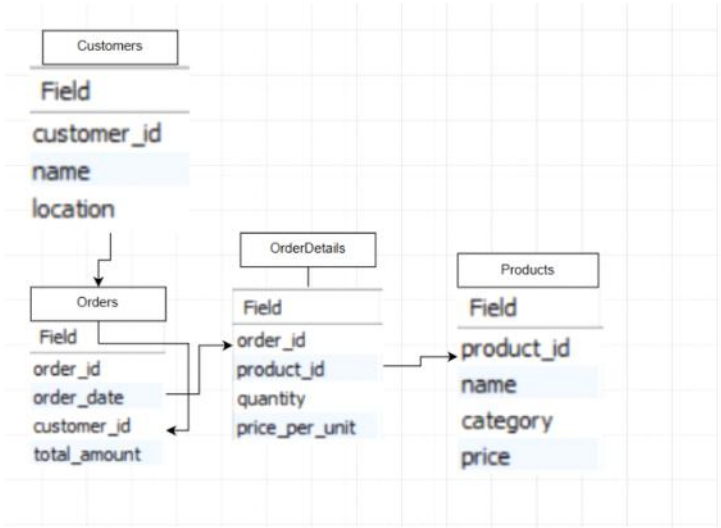
Hint:

- *Use the "Products", "Orders", "OrderDetails" and "Customers" table.*
- *Return the result table which will help you get the product names along with the count of unique customers who belong to the lower 40% of the customer pool.*

Output format:

Product_id	Name	UniqueCustomerCount
Product 1	Product1 name	NUM
Product 2	Product2 name	NUM

Note: NUM in the output format denotes a numerical value and Product name denote name of the product.




```sql
-- Challenge 9 : Low Engagement Products
SELECT * FROM Orders;

SELECT
    p.product_id,
    p.Name,
    COUNT(DISTINCT o.customer_id) AS UniqueCustomerCount
FROM Products p
JOIN Order_Details od
ON p.product_id = od.product_id
JOIN Orders o
ON o.order_id = od.order_id
GROUP BY 1,2;
```

product_id	Name	UniqueCustomerCount
1	Smartphone 6"	36
2	Laptop 15" Pro	41
3	Bluetooth Headphones	46
4	E-Book Reader	47
5	Smartwatch Fitness Tracker	44
6	Portable Bluetooth Speaker	40
7	Digital SLR Camera	45
8	Wireless Earbuds	38

```
SELECT
    p.product_id,
    p.Name,
    COUNT(DISTINCT o.customer_id) AS UniqueCustomerCount
FROM Products p
JOIN Order_Details od
ON p.product_id = od.product_id
JOIN Orders o
ON o.order_id = od.order_id
GROUP BY 1,2
HAVING UniqueCustomerCount < 40;
```

product_id	Name	UniqueCustomerCount
1	Smartphone 6"	36
8	Wireless Earbuds	38

```
135 •  SELECT COUNT(*) FROM Customers;
```

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

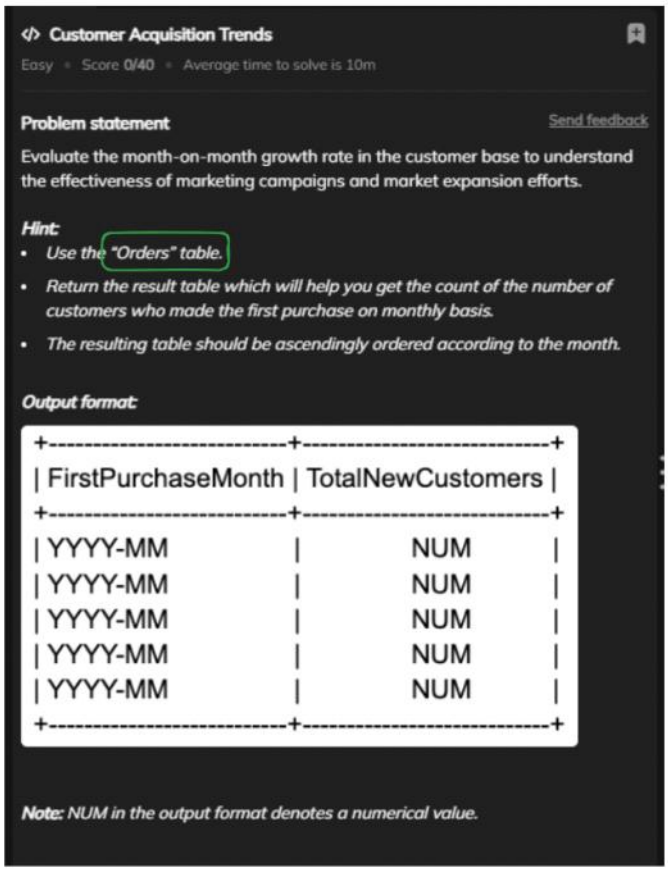
COUNT(*)
100

```
-- Challenge 9 : Low Engagement Products
SELECT * FROM Orders;
```

```
SELECT
    p.product_id,
    p.Name,
    COUNT(DISTINCT o.customer_id) AS UniqueCustomerCount
FROM Products p
JOIN Order_Details od
ON p.product_id = od.product_id
JOIN Orders o
ON o.order_id = od.order_id
GROUP BY 1,2
HAVING UniqueCustomerCount < (SELECT COUNT(*) FROM Customers) * 0.4;
```

product_id	Name	UniqueCustomerCount
1	Smartphone 6"	36
8	Wireless Earbuds	38

Challenge 10



Customer Acquisition Trends

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Evaluate the month-on-month growth rate in the customer base to understand the effectiveness of marketing campaigns and market expansion efforts.

Hint:

- *Use the "Orders" table.*
- *Return the result table which will help you get the count of the number of customers who made the first purchase on monthly basis.*
- *The resulting table should be ascendingly ordered according to the month.*

Output format:

```
+-------------------------+-------------------------+
| FirstPurchaseMonth | TotalNewCustomers |
+-------------------------+-------------------------+
| YYYY-MM                 |           NUM           |
| YYYY-MM                 |           NUM           |
| YYYY-MM                 |           NUM           |
| YYYY-MM                 |           NUM           |
| YYYY-MM                 |           NUM           |
+-------------------------+-------------------------+
```

Note: *NUM in the output format denotes a numerical value.*

```sql
SELECT
    Customer_id,
    DATE_FORMAT(Min(Order_Date) , '%Y-%m') AS FirstPurchaseMonth,
    COUNT(DISTINCT Customer_id) AS NewCustomers
FROM Orders
GROUP BY Customer_id;
```

Customer_id	FirstPurchaseMonth	NewCustomers
1	2023-10	1
2	2023-05	1
4	2023-06	1
5	2023-09	1
7	2023-03	1
9	2023-09	1
10	2023-04	1
11	2023-05	1
12	2023-04	1
13	2023-08	1
14	2023-03	1
16	2023-07	1
17	2023-04	1
18	2023-04	1
19	2023-07	1
20	2023-12	1
22	2023-05	1
23	2023-04	1
24	2023-08	1

```
-- Challenge 10 : Customer Acquisition Trends

SELECT * FROM Orders;

WITH MonthlyNewCustomers AS (
    SELECT
        Customer_id,
        DATE_FORMAT(Min(Order_Date) , '%Y-%m') AS FirstPurchaseMonth,
        COUNT(DISTINCT Customer_id) AS NewCustomers
    FROM Orders
    GROUP BY Customer_id
)
SELECT
    FirstPurchaseMonth,
    SUM(NewCustomers) AS TotalNewCustomers
FROM MonthlyNewCustomers
GROUP BY 1
ORDER BY 1;
```

Challenge 11

</> Peak Sales Period Identification

Easy • Score 0/40 • Average time to solve is 10m

Problem statement Send feedback

Identify the months with the highest sales volume, aiding in planning for stock levels, marketing efforts, and staffing in anticipation of peak demand periods.

Hint:

Use the "Orders" table.

Return the result table which will help you get the month (YYYY-MM) and the Total sales made by the company limiting to top 3 months.

The resulting table should be in descending order suggesting the highest sales month.

Output format:

```
+-------------+---------------+
| Month       | TotalSales    |
+-------------+---------------+
| YYYY-MM     |   NUM         |
| YYYY-MM     |   NUM         |
| YYYY-MM     |   NUM         |
+-------------+---------------+
```

Note: *NUM in the output format denotes a numerical value.*

```
-- Challenge 11 : Peak Sales Period Identification

SELECT * FROM Orders;

SELECT
    DATE_FORMAT(order_date , '%Y-%m') AS Month,
    SUM(total_amount) AS TotalSales
FROM Orders
GROUP BY Month
ORDER BY TotalSales DESC
LIMIT 3;
```

Month	TotalSales
2023-09	2927000
2023-12	2774000
2023-07	2568000