Case Study - E-Commerce Company

https://github.com/KrishnaMentor/CN-03

As a data analyst at our dynamic e-commerce company, you're tasked with leveraging our extensive databases to extract insights that drive our business strategies forward. Your analysis will inform various departments, from marketing to supply chain, providing them with actionable data to optimize our operations, enhance customer satisfaction, and boost our sales performance. This case study simulates real-world tasks you will encounter and requires you to apply your SQL skills to solve practical business problems.

Business Context
Your work will directly impact the following business verticals:
- Customer Insights: Understanding our customer base to tailor marketing strategies.
- Product Analysis: Evaluating product performance to inform stock and sales strategies.
- Sales Optimization: Analyzing sales data to identify trends, opportunities, and areas for improvement.
- Inventory Management: Managing stock levels to ensure product availability while minimizing excess inventory.

Dataset details:
- Customers Dataset: customer_id, name, and location
- Products Dataset: product_id, name, category, and price.
- Orders Dataset: order_id, order_date, customer_id, and total_amount.
- OrderDetails Dataset: order_id, product_id, quantity, and price_per_unit

Challenge 5

Sales Performance

5. Sales Trend Analysis: Analyze the month-on-month percentage change in total sales to identify growth trends.

Sales Trend Analysis

Easy • Score 40/40 • 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

(%Y-%m)

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

ERD : Entity Relationship Diagram

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

P.K

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

P.K

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

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

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	

```
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;
```

Month	TotalSales
2023-03	789000
2023-04	1704000
2023-05	1582000
2023-06	1040000
2023-07	2568000
2023-08	1800000
2023-09	2927000
2023-10	1497000
2023-11	1151000
2023-12	2774000
2024-01	1555000
2024-02	396000

OVER()

```
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
    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 6

6. Average Order Value Fluctuation: Examine how the average order value changes month-on-month. Insights can guide pricing and promotional strategies to enhance order value.

Average Order Value Fluctuation

Easy • Score 40/40 • 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.*

```sql
-- Challenge 6 : Average Order Value Fluctuations

With MonthlyOrderValues AS (
    SELECT
        DATE_FORMAT(order_date , '%Y-%m') AS Month,
        ROUND(AVG(total_amount),2) AS AvgOrderValue
    FROM Orders
    GROUP BY Month
    ORDER BY Month
)
SELECT * FROM MonthlyOrderValues;
```

Month	AvgOrderValue
2023-03	60692.31
2023-04	81142.86
2023-05	87888.89
2023-06	104000.00
2023-07	98769.23
2023-08	112500.00
2023-09	121958.33
2023-10	83166.67
2023-11	95916.67
2023-12	132095.24
2024-01	129583.33
2024-02	44000.00

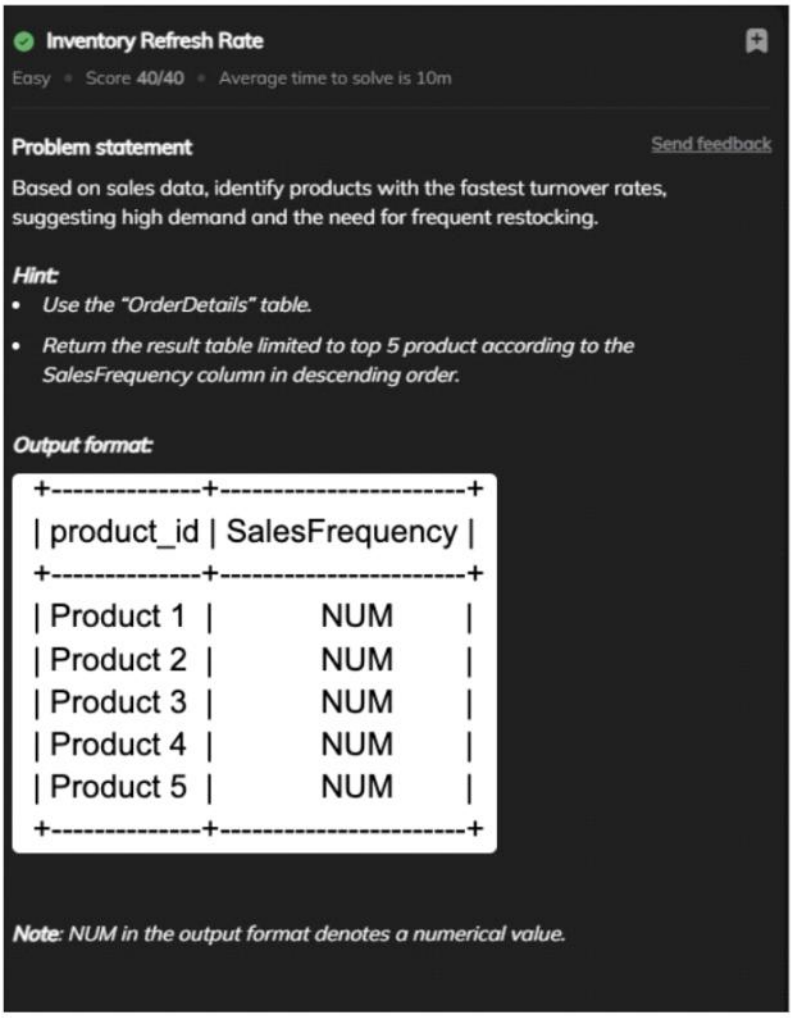
```sql
-- Challenge 6 : Average Order Value Fluctuations

With MonthlyOrderValues AS (
    SELECT
        DATE_FORMAT(order_date , '%Y-%m') AS Month,
        ROUND(AVG(total_amount),2) AS AvgOrderValue
    FROM Orders
    GROUP BY Month
    ORDER BY Month
)
SELECT
    Month,
    AvgOrderValue,
    ROUND(AvgOrderValue - LAG(AvgOrderValue) OVER(ORDER BY Month),2) AS ChangeInValue
FROM MonthlyOrderValues
ORDER BY ChangeInValue DESC;
```

Month	AvgOrderValue	ChangeInValue
2023-03	60692.31	NULL
2024-02	44000.00	-85583.33
2023-10	83166.67	-38791.66
2023-07	98769.23	-5230.77
2024-01	129583.33	-2511.91
2023-05	87888.89	6746.03
2023-09	121958.33	9458.33
2023-11	95916.67	12750.00
2023-08	112500.00	13730.77
2023-06	104000.00	16111.11
2023-04	81142.86	20450.55
2023-12	132095.24	36178.57

Challenge 7

Inventory and Stock Optimization

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



Inventory Refresh Rate

Easy • Score 40/40 • 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.*

```
-- Challenge 7 - 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 8

8. Low Engagement Products: List products purchased by less than 5% of the customer base, indicating potential mismatches between inventory and customer interest.

Low Engagement Products

Easy • Score 40/40 • 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:

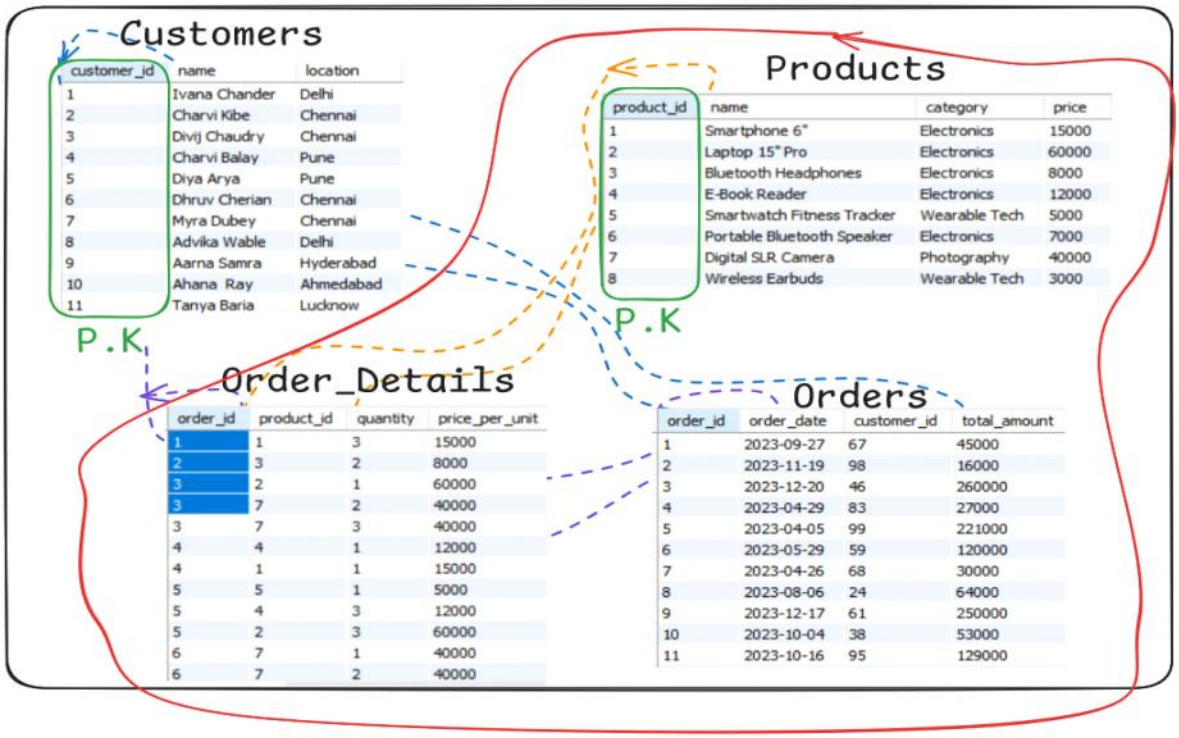
- *Use the "Products", "Orders", "OrderDetails" and "Customers" table.*
- *Return the result table which will help you get the product names along with the count of unique customers who belong to the lower 40% of the customer pool.*

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.

ERD : Entity Relationship Diagram



```sql
-- Challenge 8 - Low Engagement Products

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 p.Product_id,p.Name;
```

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

```sql
-- Challenge 8 - Low Engagement Products

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 p.Product_id,p.Name
HAVING UniqueCustomerCount < 40;
```

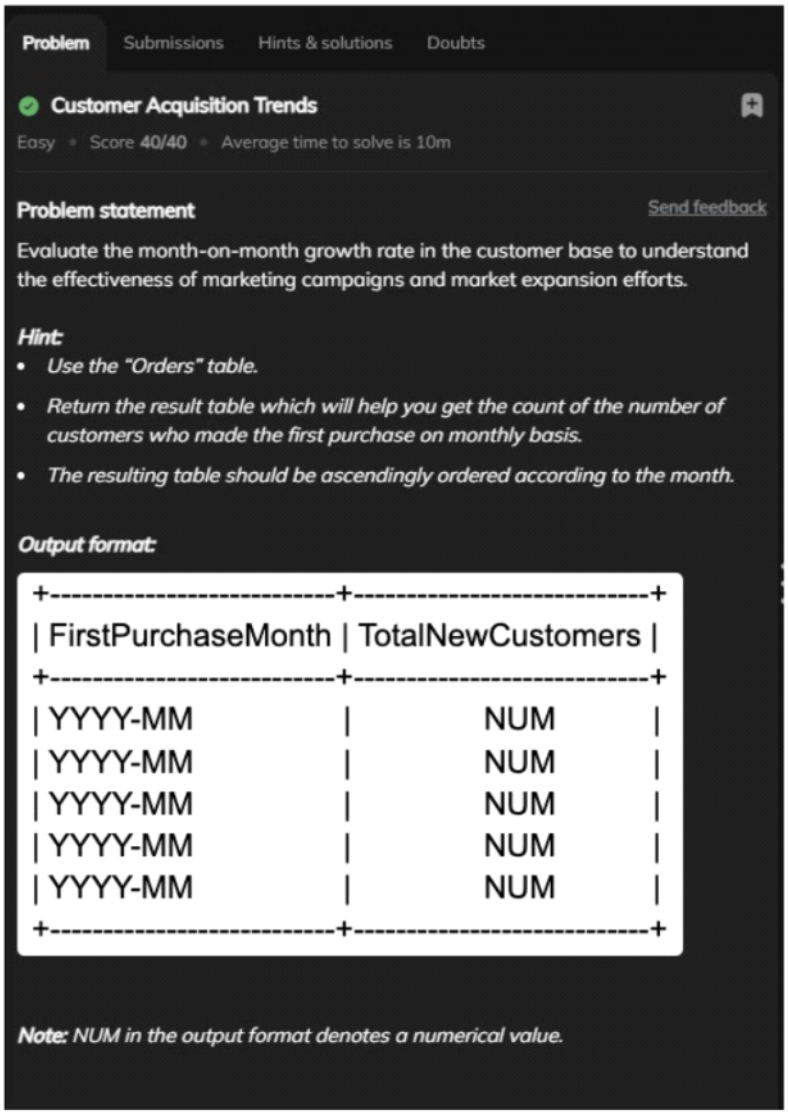
OrderDetails

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

Challenge 9

Advanced Data Analysis

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



Problem | Submissions | Hints & solutions | Doubts

Customer Acquisition Trends

Easy • Score 40/40 • 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.*

```
SELECT
    Customer_id,
    DATE_FORMAT(min(order_date) , '%Y-%m') AS FirstPurchaseMonth,
    COUNT(DISTINCT Customer_id) AS TotalNewCustomers
FROM Orders
GROUP BY Customer_id;
```

Customer_id	FirstPurchaseMonth	TotalNewCustomers
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

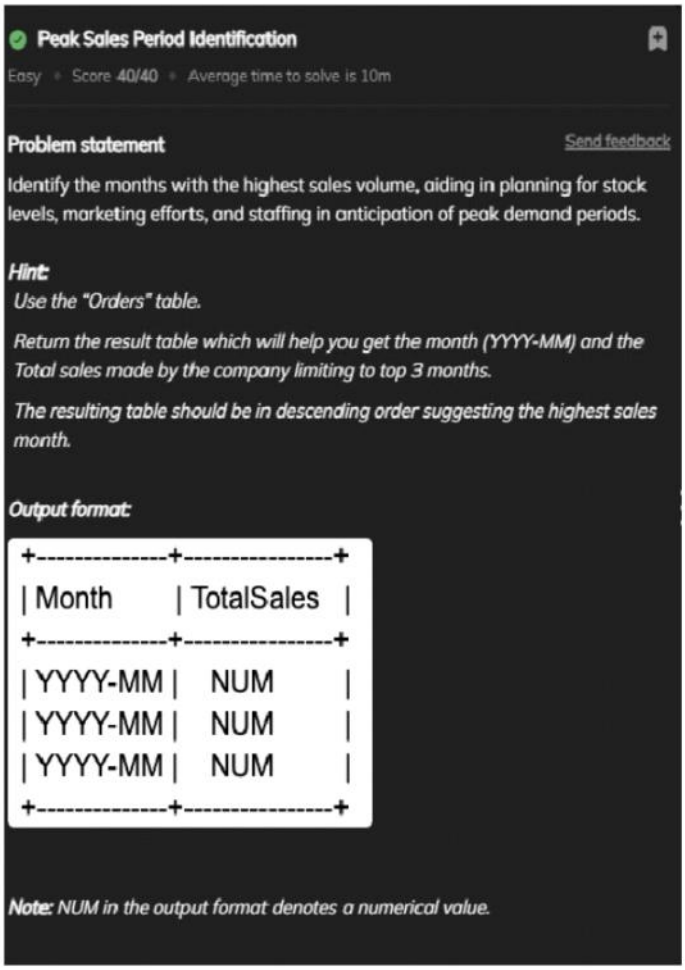
Group SUM

```sql
-- Challenge 9 - Customer Acquisitions Trends
WITH MonthlyNewCustomers AS (
    SELECT
        Customer_id,
        DATE_FORMAT(min(order_date) , '%Y-%m') AS FirstPurchaseMonth,
        COUNT(DISTINCT Customer_id) AS TotalNewCustomers
    FROM Orders
    GROUP BY Customer_id
)
SELECT
    FirstPurchaseMonth,
    SUM(TotalNewCustomers) AS TotalNewCustomers
FROM MonthlyNewCustomers
GROUP BY FirstPurchaseMonth
ORDER BY FirstPurchaseMonth ASC;
```

FirstPurchaseMonth	TotalNewCustomers
2023-03	11
2023-04	18
2023-05	11
2023-06	8
2023-07	11
2023-08	9
2023-09	5
2023-10	3
2023-11	1
2023-12	4
2024-01	2
2024-02	1

Challenge 10

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



Peak Sales Period Identification

Easy • Score 40/40 • 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 10 - 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