Case Study - E-Commerce Company

```
CREATE DATABASE Ecom;
USE Ecom;
DESC customers_india_adjusted;
DESC order_details_india_adjusted;
DESC orders_india_adjusted;
DESC products_india_adjusted;

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;
```

```
-- Understanding the dataset:
SELECT * FROM customers;
SELECT * FROM order_details;
SELECT * FROM orders;
SELECT * FROM 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

Product:

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

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	2023-04-2 2023-12-20		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

Challenge 1:

Problem Submissions Hints & solutions Doubts

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*

```
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 2:

Problem | Submissions | Hints & solutions | Doubts

Engagement Depth Analysis

Easy • Score 40/40 • 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        |
+----------------------+----------------------+
```

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

```sql
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;
```

customer_id	NumberOfOrders
42	1
66	2
45	2
32	6
94	2
81	1
35	2
93	1
30	1

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

```sql
WITH CustomerOrders AS (
    SELECT
        customer_id,
        COUNT(order_id) AS NumberOfOrders
    FROM Orders
    GROUP BY customer_id
)
SELECT
    NumberOfOrders,
    COUNT(*) AS CustomerCount
FROM CustomerOrders
GROUP BY NumberOfOrders
ORDER BY NumberOfOrders ASC;
```

Challenge 3:

Purchase High-Value Products

Easy • Score 40/40 • 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.*

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

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

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

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

```sql
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 4:

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          |
+--------------------+-----------------------+
```

Product:

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

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	2023-04-2 2023-12-20		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

```
SELECT
    p.category,
    COUNT(DISTINCT o.customer_id) AS unique_customers
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.category
ORDER BY unique_customers DESC;
```

category	unique_customers
Electronics	79
Wearable Tech	61
Photography	45

Challenge 5:

Problem | Submissions | Hints & solutions | Doubts

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         |
+--------------+---------------+--------------------+
```

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

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	2023-04-2 2023-12-20		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

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	

```sql
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 6:

Problem Submissions Hints & solutions Doubts

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.*

```sql
WITH MonthlyOrderValues AS (
    SELECT
        DATE_FORMAT(order_date , '%Y-%m') As Month,
        ROUND(AVG(total_amount),2) AS AvgOrderValue
    FROM orders
    GROUP 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
2023-12	132095.24	36178.57
2023-04	81142.86	20450.55
2023-06	104000.00	16111.11
2023-08	112500.00	13730.77
2023-11	95916.67	12750.00
2023-09	121958.33	9458.33
2023-05	87888.89	6746.03
2024-01	129583.33	-2511.91
2023-07	98769.23	-5230.77
2023-10	83166.67	-38791.66
2024-02	44000.00	-85583.33
2023-03	60692.31	NULL

Challenge 7:

Problem Submissions Hints & solutions Doubts

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.*

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

```
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:

Problem Submissions Hints & solutions Doubts

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                  |
+-------------------+-------------------+-----------------------------------+
```

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

Product:

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

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	2023-04-2 2023-12-20		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

```
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
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 < (SELECT COUNT(*) FROM Customers) * 0.4;
```

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

Challenge 9:

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.*

count() ->

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	2023-04-2 2023-12-20		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

Min() -> FirstPurchase

```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

```sql
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 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:

Problem | Submissions | Hints & solutions | Doubts

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.*

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	2023-04-2 2023-12-20		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

```sql
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