1. Top 5 customers by total purchase amount:

Explanation: This query identifies the top 5 customers based on their total spending

SELECT c.Customer\_ID, CONCAT(c.First\_Name, ' ', c.Last\_Name) AS Customer\_Name,

SUM(o.Total\_Amount) AS Total\_Spent

FROM Customer c

JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

GROUP BY c.Customer\_ID, Customer\_Name

ORDER BY Total\_Spent DESC

LIMIT 5;

1. Monthly sales trend:

Explanation: This query shows the sales trend by month.

SELECT DATE\_FORMAT(Order\_Date, '%Y-%m') AS Month,

SUM(Total\_Amount) AS Monthly\_Sales

FROM Orders

GROUP BY Month

ORDER BY Month;

1. Product popularity by category:

Explanation: This query shows the popularity of products within each category.

SELECT c.C\_Name AS Category, p.P\_Name AS Product,

COUNT(op.Order\_ID) AS Times\_Ordered

FROM Category c

JOIN Product p ON c.Category\_ID = p.Category\_ID

LEFT JOIN Order\_Product op ON p.Product\_ID = op.Product\_ID

GROUP BY c.C\_Name, p.P\_Name

ORDER BY c.C\_Name, Times\_Ordered DESC;

1. Average order value by customer location:

Explanation: This query calculates the average order value for each state.

SELECT c.State, AVG(o.Total\_Amount) AS Avg\_Order\_Value

FROM Customer c

JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

GROUP BY c.State

ORDER BY Avg\_Order\_Value DESC;

1. Customers who haven't placed an order in the last 30 days:

Explanation: This query identifies customers who haven't placed an order in the last 30 days.

SELECT c.Customer\_ID, CONCAT(c.First\_Name, ' ', c.Last\_Name) AS Customer\_Name

FROM Customer c

LEFT JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

AND o.Order\_Date >= DATE\_SUB(CURDATE(), INTERVAL 30 DAY)

WHERE o.Order\_ID IS NULL;

1. Products with low stock (less than 10 items):

Explanation: This query lists products with low stock.

SELECT Product\_ID, P\_Name, Stock\_Quantity

FROM Product

WHERE Stock\_Quantity < 10

ORDER BY Stock\_Quantity;

1. Customer retention rate:

Explanation: This query calculates the customer retention rate.

WITH CurrentMonthCustomers AS (

SELECT DISTINCT Customer\_ID

FROM Orders

WHERE Order\_Date >= DATE\_FORMAT(CURDATE() - INTERVAL 1 MONTH, '%Y-%m-01')

),

PreviousMonthCustomers AS (

SELECT DISTINCT Customer\_ID

FROM Orders

WHERE Order\_Date >= DATE\_FORMAT(CURDATE() - INTERVAL 2 MONTH, '%Y-%m-01')

AND Order\_Date < DATE\_FORMAT(CURDATE() - INTERVAL 1 MONTH, '%Y-%m-01')

)

SELECT

COUNT(DISTINCT cmc.Customer\_ID) AS RetainedCustomers,

COUNT(DISTINCT pmc.Customer\_ID) AS PreviousMonthCustomers,

COUNT(DISTINCT cmc.Customer\_ID) / COUNT(DISTINCT pmc.Customer\_ID) \* 100 AS RetentionRate

FROM PreviousMonthCustomers pmc

LEFT JOIN CurrentMonthCustomers cmc ON pmc.Customer\_ID = cmc.Customer\_ID;

1. Rolling 3-month average of sales:

Explanation: This query calculates a rolling 3-month average of sales using a window function.

SELECT

DATE\_FORMAT(Order\_Date, '%Y-%m') AS Month,

SUM(Total\_Amount) AS Monthly\_Sales,

AVG(SUM(Total\_Amount)) OVER (

ORDER BY DATE\_FORMAT(Order\_Date, '%Y-%m')

ROWS BETWEEN 2 PRECEDING AND CURRENT ROW

) AS Rolling\_3\_Month\_Avg

FROM Orders

GROUP BY Month

ORDER BY Month;

1. Rank products by sales within their category:

Explanation: This query ranks products by their sales within each category using a window function.

SELECT

c.C\_Name AS Category,

p.P\_Name AS Product,

SUM(op.Quantity \* p.Price) AS Total\_Sales,

RANK() OVER (

PARTITION BY c.Category\_ID

ORDER BY SUM(op.Quantity \* p.Price) DESC

) AS Sales\_Rank

FROM Category c

JOIN Product p ON c.Category\_ID = p.Category\_ID

JOIN Order\_Product op ON p.Product\_ID = op.Product\_ID

GROUP BY c.Category\_ID, p.Product\_ID

ORDER BY c.C\_Name, Sales\_Rank;

1. Customer segmentation by total spend (OLAP query):

Explanation: This OLAP query segments customers based on their total spend and provides a summary with a rollup.

SELECT

CASE

WHEN Total\_Spent < 100 THEN 'Low Spender'

WHEN Total\_Spent BETWEEN 100 AND 500 THEN 'Medium Spender'

ELSE 'High Spender'

END AS Customer\_Segment,

COUNT(\*) AS Customer\_Count,

AVG(Total\_Spent) AS Avg\_Spend

FROM (

SELECT c.Customer\_ID, SUM(o.Total\_Amount) AS Total\_Spent

FROM Customer c

JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

GROUP BY c.Customer\_ID

) AS CustomerSpend

GROUP BY Customer\_Segment

WITH ROLLUP;

1. Product cross-sell analysis:

Explanation: This query identifies products that are frequently bought together.

SELECT

p1.P\_Name AS Product1,

p2.P\_Name AS Product2,

COUNT(\*) AS Co\_occurrence

FROM Order\_Product op1

JOIN Order\_Product op2 ON op1.Order\_ID = op2.Order\_ID AND op1.Product\_ID < op2.Product\_ID

JOIN Product p1 ON op1.Product\_ID = p1.Product\_ID

JOIN Product p2 ON op2.Product\_ID = p2.Product\_ID

GROUP BY p1.Product\_ID, p2.Product\_ID

ORDER BY Co\_occurrence DESC

LIMIT 10;

1. Customer lifetime value calculation:

Explanation: This query calculates various metrics to determine customer lifetime value.

SELECT

c.Customer\_ID,

CONCAT(c.First\_Name, ' ', c.Last\_Name) AS Customer\_Name,

COUNT(DISTINCT o.Order\_ID) AS Total\_Orders,

SUM(o.Total\_Amount) AS Total\_Spent,

AVG(o.Total\_Amount) AS Avg\_Order\_Value,

DATEDIFF(MAX(o.Order\_Date), MIN(o.Order\_Date)) / 365.0 AS Years\_Active,

SUM(o.Total\_Amount) / (DATEDIFF(MAX(o.Order\_Date), MIN(o.Order\_Date)) / 365.0) AS Annual\_Value

FROM Customer c

JOIN Orders o ON c.Customer\_ID = o.Customer\_ID

GROUP BY c.Customer\_ID

ORDER BY Annual\_Value DESC

LIMIT 10;

1. Seasonal sales analysis (OLAP query):

Explanation: This OLAP query provides a seasonal analysis of sales with cumulative totals.

SELECT

YEAR(Order\_Date) AS Year,

QUARTER(Order\_Date) AS Quarter,

SUM(Total\_Amount) AS Quarterly\_Sales,

SUM(SUM(Total\_Amount)) OVER (PARTITION BY YEAR(Order\_Date) ORDER BY QUARTER(Order\_Date)) AS Cumulative\_Yearly\_Sales

FROM Orders

GROUP BY Year, Quarter

WITH ROLLUP;

1. Customer purchase frequency distribution:

Explanation: This query categorizes customers based on their purchase frequency.

WITH PurchaseFrequency AS (

SELECT Customer\_ID, COUNT(DISTINCT Order\_ID) AS Order\_Count

FROM Orders

GROUP BY Customer\_ID

)

SELECT

CASE

WHEN Order\_Count = 1 THEN 'One-time'

WHEN Order\_Count BETWEEN 2 AND 5 THEN '2-5 times'

WHEN Order\_Count BETWEEN 6 AND 10 THEN '6-10 times'

ELSE 'More than 10 times'

END AS Purchase\_Frequency,

COUNT(\*) AS Customer\_Count,

AVG(Order\_Count) AS Avg\_Orders

FROM PurchaseFrequency

GROUP BY Purchase\_Frequency

ORDER BY MIN(Order\_Count);

1. Product review sentiment analysis:

Explanation: This query performs a basic sentiment analysis on product reviews.

SELECT

p.Product\_ID,

p.P\_Name,

AVG(r.Rating) AS Avg\_Rating,

COUNT(r.Review\_ID) AS Review\_Count,

SUM(CASE WHEN r.Review\_Text LIKE '%great%' OR r.Review\_Text LIKE '%excellent%' THEN 1 ELSE 0 END) AS Positive\_Reviews,

SUM(CASE WHEN r.Review\_Text LIKE '%poor%' OR r.Review\_Text LIKE '%bad%' THEN 1 ELSE 0 END) AS Negative\_Reviews

FROM Product p

LEFT JOIN Reviews r ON p.Product\_ID = r.Product\_ID

GROUP BY p.Product\_ID, p.P\_Name

HAVING Review\_Count > 0

ORDER BY Avg\_Rating DESC, Review\_Count DESC;