-- Task 1: All orders from Germany

SELECT \* FROM ecommerce\_data

WHERE Country = 'Germany';

|  |
| --- |
|  |

-- Task 2: Top 5 best-selling products

SELECT Description, SUM(Quantity) AS TotalSold

FROM ecommerce\_data

GROUP BY Description

ORDER BY TotalSold DESC

LIMIT 5;

|  |
| --- |
|  |

-- Task 3: Revenue by country

SELECT Country, SUM(Quantity \* UnitPrice) AS Revenue

FROM ecommerce\_data

GROUP BY Country

ORDER BY Revenue DESC;

|  |
| --- |
|  |

**Use \G instead of ; to display output vertically, which can help when columns are wide**

-- Task 4: Invoice-level summary

SELECT InvoiceNo, CustomerID, SUM(Quantity \* UnitPrice) AS InvoiceTotal

FROM ecommerce\_data

GROUP BY InvoiceNo, CustomerID;

|  |
| --- |
|  |

**Here limit for 20 is used because it was showing invoice total for each and every customer, so I used 20 limit to display less number of values.**

-- Task 5: LEFT JOIN-style query (products with or without orders)

SELECT DISTINCT ed.StockCode, ed.Description, ed.Quantity

FROM ecommerce\_data ed

LEFT JOIN (

SELECT StockCode FROM ecommerce\_data WHERE Quantity > 0

) sub ON ed.StockCode = sub.StockCode   
LIMIT 20;

|  |
| --- |
|  |

-- Task 6: RIGHT JOIN simulation (invoices without items)

SELECT DISTINCT ed.InvoiceNo

FROM (

SELECT DISTINCT InvoiceNo FROM ecommerce\_data

) invoices

LEFT JOIN ecommerce\_data ed ON invoices.InvoiceNo = ed.InvoiceNo

WHERE ed.StockCode IS NULL;

|  |
| --- |
|  |

-- Task 7: Customers who spent more than average

SELECT CustomerID

FROM ecommerce\_data

GROUP BY CustomerID

HAVING SUM(Quantity \* UnitPrice) > (

SELECT AVG(totals) FROM (

SELECT SUM(Quantity \* UnitPrice) AS totals

FROM ecommerce\_data

GROUP BY CustomerID

) avg\_sub

) LIMIT 20;

|  |
| --- |
|  |

**Here limit for 20 is used because it was showing invoice total for each and every customer, so I used 20 limit to display less number of values.**

-- Task 8: Aggregates: order count, average, max, min

SELECT

COUNT(DISTINCT InvoiceNo) AS TotalOrders,

AVG(Quantity \* UnitPrice) AS AvgOrderValue,

MAX(Quantity) AS MaxQty,

MIN(Quantity) AS MinQty

FROM ecommerce\_data;

|  |
| --- |
|  |

-- Task 9: Create view for top 10 customers

CREATE VIEW top\_customers AS

SELECT CustomerID, SUM(Quantity \* UnitPrice) AS TotalSpent

FROM ecommerce\_data

GROUP BY CustomerID

ORDER BY TotalSpent DESC

LIMIT 10;

|  |
| --- |
|  |

-- Task 10: Create indexes

CREATE INDEX idx\_invoice\_no ON ecommerce\_data(InvoiceNo);

CREATE INDEX idx\_customer\_id ON ecommerce\_data(CustomerID);

CREATE INDEX idx\_stock\_code ON ecommerce\_data(StockCode);

|  |
| --- |
|  |