Advanced SQL Exercises for Online Retail Store

Exercise 1: Ranking and Window Functions

Goal: Use ROW_NUMBER(), RANK(), DENSE_RANK(), OVER(), and PARTITION BY.

Scenario:

Find the top 3 most expensive products in each category using different ranking functions.

Steps:

1. Use ROW_NUMBER() to assign a unique rank within each category.
2. Use RANK() and DENSE_RANK() to compare how ties are handled.
3. Use PARTITION BY Category and ORDER BY Price DESC.

Exercise 2: Aggregation with GROUPING SETS, CUBE, and ROLLUP

Goal: Analyze sales data across multiple dimensions.

Scenario:

Generate a report showing total quantity sold by Region and Category using GROUPING SETS, ROLLUP, an

Steps:

1. Join Orders, OrderDetails, Customers, and Products.
2. Use GROUPING SETS to get totals by Region, Category, and both.
3. Use ROLLUP to get subtotals and grand totals.
4. Use CUBE to get all combinations of Region and Category.

Exercise 3: CTEs and MERGE

Goal: Use WITH, CTEs, Recursive CTEs, and MERGE.

Scenario:

a) Create a recursive CTE to generate a calendar table.

b) Use a MERGE statement to update or insert product prices from a staging table.

Steps:

1. Create a recursive CTE to generate dates from '2025-01-01' to '2025-01-31'.
2. Create a StagingProducts table with updated prices.
3. Use MERGE to update existing products or insert new ones.

Exercise 4: PIVOT and UNPIVOT

Goal: Transform data for reporting.

Scenario:

Show monthly sales quantity per product in a pivoted format, and then unpivot it back.

Steps:

1. Aggregate sales by Product and Month.
2. Use PIVOT to convert rows into columns (one column per month).
3. Use UNPIVOT to convert the pivoted data back into row format.

Exercise 5: Using CTE to Simplify a Query

Goal: Use Common Table Expressions (CTEs) to simplify complex queries.

Scenario:

The business wants to find all customers who have placed more than 3 orders in total.

Steps:

1. Create a CTE that counts the number of orders placed by each customer.
2. Select only those customers who have placed more than 3 orders.

Sample Query:

```
WITH CustomerOrderCounts AS (
    SELECT
        o.CustomerID,
        COUNT(o.OrderID) AS OrderCount
    FROM Orders o
    GROUP BY o.CustomerID
)
SELECT
    c.CustomerID,
    c.Name,
    coc.OrderCount
FROM CustomerOrderCounts coc
JOIN Customers c ON c.CustomerID = coc.CustomerID
WHERE coc.OrderCount > 3;
```