**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.

**CODE:**

-- Database Schema

CREATE TABLE Customers (

    CustomerID INT PRIMARY KEY,

    Name VARCHAR(100),

    Region VARCHAR(50)

);

CREATE TABLE Products (

    ProductID INT PRIMARY KEY,

    ProductName VARCHAR(100),

    Category VARCHAR(50),

    Price DECIMAL(10, 2)

);

CREATE TABLE Orders (

    OrderID INT PRIMARY KEY,

    CustomerID INT,

    OrderDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

    OrderDetailID INT PRIMARY KEY,

    OrderID INT,

    ProductID INT,

    Quantity INT,

    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

-- Sample Data

INSERT INTO Customers (CustomerID, Name, Region) VALUES

(1, 'Alice', 'North'),

(2, 'Bob', 'South'),

(3, 'Charlie', 'East'),

(4, 'David', 'West');

INSERT INTO Products (ProductID, ProductName, Category, Price) VALUES

(1, 'Laptop', 'Electronics', 1200.00),

(2, 'Smartphone', 'Electronics', 800.00),

(3, 'Tablet', 'Electronics', 600.00),

(4, 'Headphones', 'Accessories', 150.00);

INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES

(1, 1, '2023-01-15'),

(2, 2, '2023-02-20'),

(3, 3, '2023-03-25'),

(4, 4, '2023-04-30');

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES

(1, 1, 1, 1),

(2, 2, 2, 2),

(3, 3, 3, 1),

(4, 4, 4, 3);

-- Exercise 1: Creating a Non-Clustered Index

-- Goal: Create a non-clustered index on the ProductName column in the Products table and compare query execution time before and after index creation.

-- Step 1: Query to fetch product details before index creation

SELECT \* FROM Products WHERE ProductName = 'Laptop';

-- Step 2: Create a non-clustered index on ProductName

-- Step 3: Query to fetch product details after index creation

SELECT \* FROM Products WHERE ProductName = 'Laptop';

-- Exercise 2: Creating a Clustered Index

-- Goal: Create a clustered index on the OrderDate column in the Orders table and compare query execution time before and after index creation.

-- Step 1: Query to fetch orders before index creation

SELECT \* FROM Orders WHERE OrderDate = '2023-01-15';

-- Step 2: Create a clustered index on OrderDate

-- Step 3: Query to fetch orders after index creation

SELECT \* FROM Orders WHERE OrderDate = '2023-01-15';

-- Exercise 3: Creating a Composite Index

-- Goal: Create a composite index on the CustomerID and OrderDate columns in the Orders table and compare query execution time before and after index creation.

-- Step 1: Query to fetch orders before index creation

SELECT \* FROM Orders WHERE CustomerID = 1 AND OrderDate = '2023-01-15';

-- Step 2: Create a composite index on CustomerID and OrderDate

-- Step 3: Query to fetch orders after index creation

SELECT \* FROM Orders WHERE CustomerID = 1 AND OrderDate = '2023-01-15';

WITH RankedProducts AS (

    SELECT

        ProductID,

        ProductName,

        Category,

        Price,

        ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum,

        RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankNum,

        DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum

    FROM

        Products

)

SELECT

    ProductID,

    ProductName,

    Category,

    Price,

    RowNum,

    RankNum,

    DenseRankNum

FROM

    RankedProducts

WHERE

    RowNum <= 3 OR RankNum <= 3 OR DenseRankNum <= 3;

**OUTPUT:**

