|  |
| --- |
| **1. SQL Exercise - Advanced concepts** |

**Exercise 1: Ranking and Window Functions**

CREATE DATABASE ProductRankingDemo;

GO

USE ProductRankingDemo;

GO

CREATE TABLE Products (

ProductID INT IDENTITY(1,1) PRIMARY KEY,

ProductName NVARCHAR(100),

Category NVARCHAR(50),

Price DECIMAL(10, 2)

);

GO

INSERT INTO Products (ProductName, Category, Price)

VALUES

('Smartphone A', 'Electronics', 899.99),

('Smartphone B', 'Electronics', 799.99),

('Laptop A', 'Electronics', 1199.99),

('Laptop B', 'Electronics', 1199.99), -- tie

('Tablet A', 'Electronics', 499.99),

('Tablet B', 'Electronics', 399.99),

('Sofa A', 'Furniture', 999.99),

('Sofa B', 'Furniture', 999.99), -- tie

('Dining Table', 'Furniture', 799.99),

('Chair Set', 'Furniture', 499.99),

('TV Stand', 'Furniture', 299.99),

('Book A', 'Books', 19.99),

('Book B', 'Books', 24.99),

('Book C', 'Books', 14.99),

('Book D', 'Books', 24.99), -- tie

('Book E', 'Books', 29.99);

GO

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 \*

FROM RankedProducts

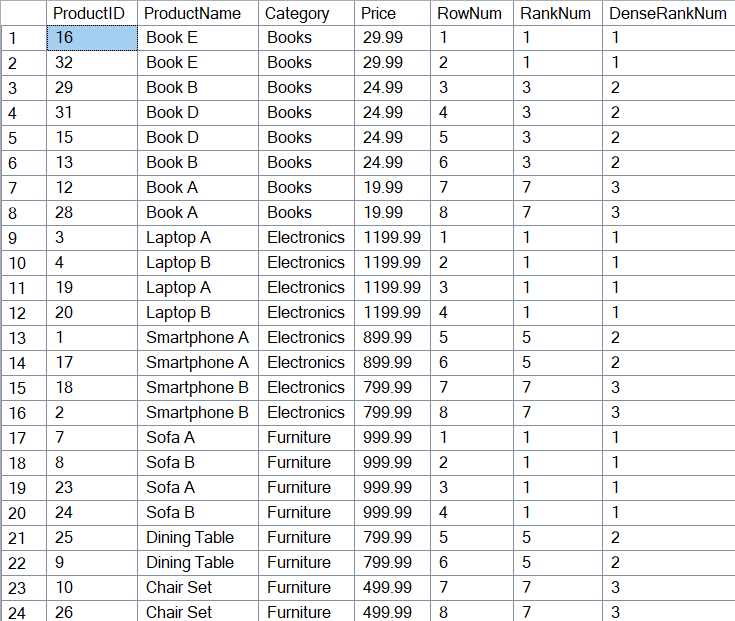
WHERE RowNum <= 3

OR RankNum <= 3

OR DenseRankNum <= 3

ORDER BY Category, Price DESC;

OUTPUT



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

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

CustomerName NVARCHAR(100),

Region NVARCHAR(50)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName NVARCHAR(100),

Category NVARCHAR(50)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),

OrderDate DATE

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY IDENTITY,

OrderID INT FOREIGN KEY REFERENCES Orders(OrderID),

ProductID INT FOREIGN KEY REFERENCES Products(ProductID),

Quantity INT

);

-- Customers

INSERT INTO Customers (CustomerID, CustomerName, Region) VALUES

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

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

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

(4, 'Diana', 'North');

-- Products

INSERT INTO Products (ProductID, ProductName, Category) VALUES

(101, 'Laptop', 'Electronics'),

(102, 'Mouse', 'Electronics'),

(103, 'Desk', 'Furniture'),

(104, 'Chair', 'Furniture');

-- Orders

INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES

(1001, 1, '2024-06-01'),

(1002, 2, '2024-06-02'),

(1003, 3, '2024-06-03'),

(1004, 4, '2024-06-04');

-- OrderDetails

INSERT INTO OrderDetails (OrderID, ProductID, Quantity) VALUES

(1001, 101, 2),

(1001, 103, 1),

(1002, 102, 5),

(1002, 104, 2),

(1003, 103, 3),

(1004, 101, 1),

(1004, 104, 2);

--- Join Orders, OrderDetails, Customers, and Products and Grouping sets

SELECT

c.Region,

p.Category,

SUM(od.Quantity) AS TotalQuantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY GROUPING SETS (

(c.Region),

(p.Category),

(c.Region, p.Category)

)

ORDER BY GROUPING(c.Region), GROUPING(p.Category), c.Region, p.Category;

---ROLLUP Report

SELECT

c.Region,

p.Category,

SUM(od.Quantity) AS TotalQuantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY ROLLUP (c.Region, p.Category)

ORDER BY GROUPING(c.Region), GROUPING(p.Category), c.Region, p.Category;

--- Cube Report

SELECT

c.Region,

p.Category,

SUM(od.Quantity) AS TotalQuantity

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

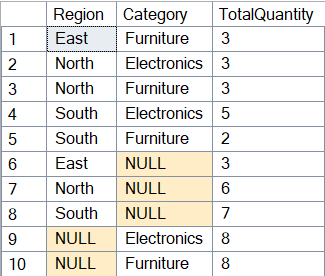
JOIN Customers c ON o.CustomerID = c.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

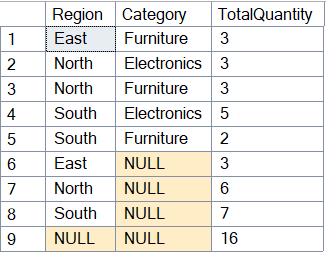
GROUP BY CUBE (c.Region, p.Category)

ORDER BY GROUPING(c.Region), GROUPING(p.Category), c.Region, p.Category;

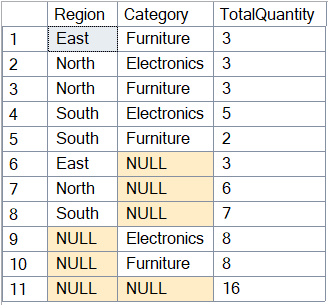
**Joining and Grouping sets**



**Rollup Report**



**Cube Report**



**Exercise 3: CTEs and MERGE**

IF OBJECT\_ID('dbo.OrderDetails', 'U') IS NOT NULL

DROP TABLE dbo.OrderDetails;

IF OBJECT\_ID('dbo.Orders', 'U') IS NOT NULL

DROP TABLE dbo.Orders;

IF OBJECT\_ID('dbo.StagingProducts', 'U') IS NOT NULL

DROP TABLE dbo.StagingProducts;

IF OBJECT\_ID('dbo.Products', 'U') IS NOT NULL

DROP TABLE dbo.Products;

IF OBJECT\_ID('dbo.Calendar', 'U') IS NOT NULL

DROP TABLE dbo.Calendar;

GO

CREATE TABLE dbo.Calendar

(

CalendarDate DATE PRIMARY KEY,

DayOfWeekName AS DATENAME(WEEKDAY, CalendarDate)

);

GO

;WITH CalendarCTE AS

(

SELECT CAST('2025-01-01' AS DATE) AS CalendarDate

UNION ALL

SELECT DATEADD(DAY, 1, CalendarDate)

FROM CalendarCTE

WHERE CalendarDate < '2025-01-31'

)

INSERT INTO dbo.Calendar (CalendarDate)

SELECT CalendarDate

FROM CalendarCTE

OPTION (MAXRECURSION 0);

GO

CREATE TABLE dbo.Products

(

ProductID INT PRIMARY KEY,

ProductName NVARCHAR(100),

Price DECIMAL(10,2)

);

INSERT INTO dbo.Products (ProductID, ProductName, Price) VALUES

(1, 'Laptop', 1200.00),

(2, 'Mouse', 25.00),

(3, 'Desk', 350.00),

(4, 'Office Chair', 150.00);

GO

CREATE TABLE dbo.StagingProducts

(

ProductID INT PRIMARY KEY,

ProductName NVARCHAR(100),

Price DECIMAL(10,2)

);

INSERT INTO dbo.StagingProducts (ProductID, ProductName, Price) VALUES

(2, 'Mouse', 19.99),

(3, 'Desk', 329.00),

(5, 'Keyboard', 49.99);

GO

MERGE dbo.Products AS tgt

USING dbo.StagingProducts AS src

ON tgt.ProductID = src.ProductID

WHEN MATCHED THEN

UPDATE SET

tgt.ProductName = src.ProductName,

tgt.Price = src.Price

WHEN NOT MATCHED BY TARGET THEN

INSERT (ProductID, ProductName, Price)

VALUES (src.ProductID, src.ProductName, src.Price)

OUTPUT

$action AS MergeAction,

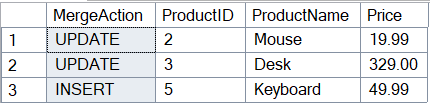
INSERTED.\*;

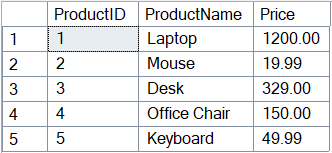
GO

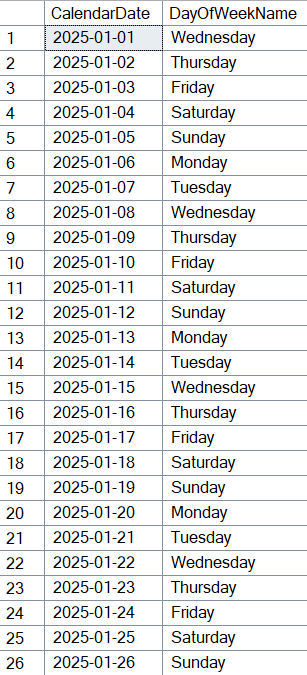
SELECT \* FROM dbo.Products ORDER BY ProductID;

SELECT \* FROM dbo.Calendar ORDER BY CalendarDate;

OUTPUT







**Exercise 4: PIVOT and UNPIVOT**

IF OBJECT\_ID('dbo.Sales', 'U') IS NOT NULL DROP TABLE dbo.Sales;

GO

CREATE TABLE dbo.Sales (

ProductName NVARCHAR(100),

SaleDate DATE,

Quantity INT

);

INSERT INTO dbo.Sales (ProductName, SaleDate, Quantity) VALUES

('Laptop', '2025-01-15', 10),

('Laptop', '2025-02-10', 8),

('Laptop', '2025-03-05', 12),

('Mouse', '2025-01-20', 25),

('Mouse', '2025-02-15', 18),

('Mouse', '2025-03-12', 30),

('Keyboard','2025-01-08', 15),

('Keyboard','2025-03-25', 20);

GO

SELECT

ProductName,

FORMAT(SaleDate, 'MMM') AS Month,

SUM(Quantity) AS TotalQuantity

FROM dbo.Sales

GROUP BY ProductName, FORMAT(SaleDate, 'MMM');

SELECT \*

FROM (

SELECT

ProductName,

FORMAT(SaleDate, 'MMM') AS Month,

Quantity

FROM dbo.Sales

) AS SourceTable

PIVOT (

SUM(Quantity)

FOR Month IN ([Jan], [Feb], [Mar])

) AS PivotTable;

SELECT

ProductName,

Month,

Quantity

FROM (

SELECT

ProductName, [Jan], [Feb], [Mar]

FROM (

SELECT

ProductName,

FORMAT(SaleDate, 'MMM') AS Month,

Quantity

FROM dbo.Sales

) AS SourceTable

PIVOT (

SUM(Quantity)

FOR Month IN ([Jan], [Feb], [Mar])

) AS PivotTable

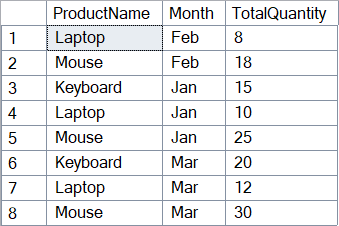
) AS Pivoted

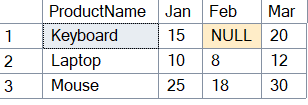
UNPIVOT (

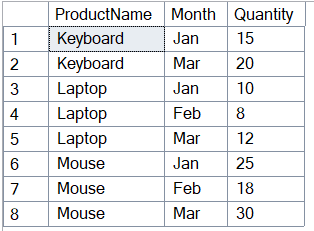
Quantity FOR Month IN ([Jan], [Feb], [Mar])

) AS Unpivoted;

**OUTPUT**

****

****

****

**Exercise 5: Using CTE to Simplify a Query**

IF OBJECT\_ID('dbo.Orders', 'U') IS NOT NULL DROP TABLE dbo.Orders;

IF OBJECT\_ID('dbo.Customers', 'U') IS NOT NULL DROP TABLE dbo.Customers;

GO

CREATE TABLE dbo.Customers (

CustomerID INT PRIMARY KEY,

Name NVARCHAR(100)

);

CREATE TABLE dbo.Orders (

OrderID INT PRIMARY KEY,

CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),

OrderDate DATE

);

INSERT INTO dbo.Customers (CustomerID, Name) VALUES

(1, 'Alice'),

(2, 'Bob'),

(3, 'Charlie'),

(4, 'Diana'),

(5, 'Eve');

INSERT INTO dbo.Orders (OrderID, CustomerID, OrderDate) VALUES

(101, 1, '2025-01-01'),

(102, 1, '2025-01-02'),

(103, 1, '2025-01-03'),

(104, 1, '2025-01-04'),

(105, 2, '2025-01-01'),

(106, 2, '2025-01-03'),

(107, 3, '2025-01-05'),

(108, 4, '2025-01-01'),

(109, 4, '2025-01-02'),

(110, 4, '2025-01-03'),

(111, 4, '2025-01-04');

GO

WITH CustomerOrderCounts AS (

SELECT

o.CustomerID,

COUNT(o.OrderID) AS OrderCount

FROM dbo.Orders o

GROUP BY o.CustomerID

)

SELECT

c.CustomerID,

c.Name,

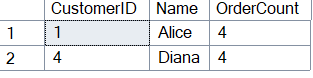
coc.OrderCount

FROM CustomerOrderCounts coc

JOIN dbo.Customers c ON c.CustomerID = coc.CustomerID

WHERE coc.OrderCount > 3;

OUTPUT



|  |
| --- |
| **4. SQL Exercise - Stored procedure** |

**Exercise 1: Create a Stored Procedure**

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

GO

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

GO

INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID, Salary,

JoinDate) VALUES

(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),

(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),

(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');

GO

--- Create Store Procedure

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

d.DepartmentName,

e.Salary,

e.JoinDate

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID

WHERE e.DepartmentID = @DepartmentID;

END;

GO

CREATE PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

END;

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;

Example OUTPUT



**Exercise 5: Return Data from a Stored Procedure**

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

GO

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

GO

INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID, Salary,

JoinDate) VALUES

(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),

(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),

(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');

GO

--- Create Store Procedure

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

d.DepartmentName,

e.Salary,

e.JoinDate

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID

WHERE e.DepartmentID = @DepartmentID;

END;

GO

CREATE PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

END;

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

COUNT(\*) AS EmployeeCount

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 3;

**OUTPUT**

