Advanced SQL Exercises for Online Retail Store

**Exercise 1: Ranking and Window Functions**

-- Drop the table if it exists

IF OBJECT\_ID('StoreItems', 'U') IS NOT NULL

DROP TABLE StoreItems;

GO

-- Create a new table for store items

CREATE TABLE StoreItems (

ItemID INT PRIMARY KEY,

ItemName NVARCHAR(100),

Category NVARCHAR(50),

Price DECIMAL(10, 2)

);

GO

-- Insert fresh data

INSERT INTO StoreItems (ItemID, ItemName, Category, Price)

VALUES

(1, 'Espresso Machine', 'Home Appliances', 220.00),

(2, 'Blender', 'Home Appliances', 150.00),

(3, 'Air Purifier', 'Home Appliances', 220.00),

(4, 'Toaster', 'Home Appliances', 60.00),

(5, 'Gaming Chair', 'Furniture', 320.00),

(6, 'Bookshelf', 'Furniture', 180.00),

(7, 'Office Desk', 'Furniture', 320.00),

(8, 'Bean Bag', 'Furniture', 100.00),

(9, 'LED Light Strip', 'Decor', 50.00),

(10, 'Wall Art', 'Decor', 120.00),

(11, 'Smart Mirror', 'Decor', 120.00),

(12, 'Vase', 'Decor', 45.00);

GO

-- Use ROW\_NUMBER to get top 3 most expensive items per category

WITH RowRanked AS (

SELECT \*,

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

FROM StoreItems

)

SELECT 'ROW\_NUMBER()' AS RankType, \*

FROM RowRanked

WHERE RowRank <= 3

ORDER BY Category, RowRank;

GO

-- Use RANK to show how ties introduce gaps

WITH RankRanked AS (

SELECT \*,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankVal

FROM StoreItems

)

SELECT 'RANK()' AS RankType, \*

FROM RankRanked

WHERE RankVal <= 3

ORDER BY Category, RankVal;

GO

-- Use DENSE\_RANK to show how ties do not introduce gaps

WITH DenseRanked AS (

SELECT \*,

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

FROM StoreItems

)

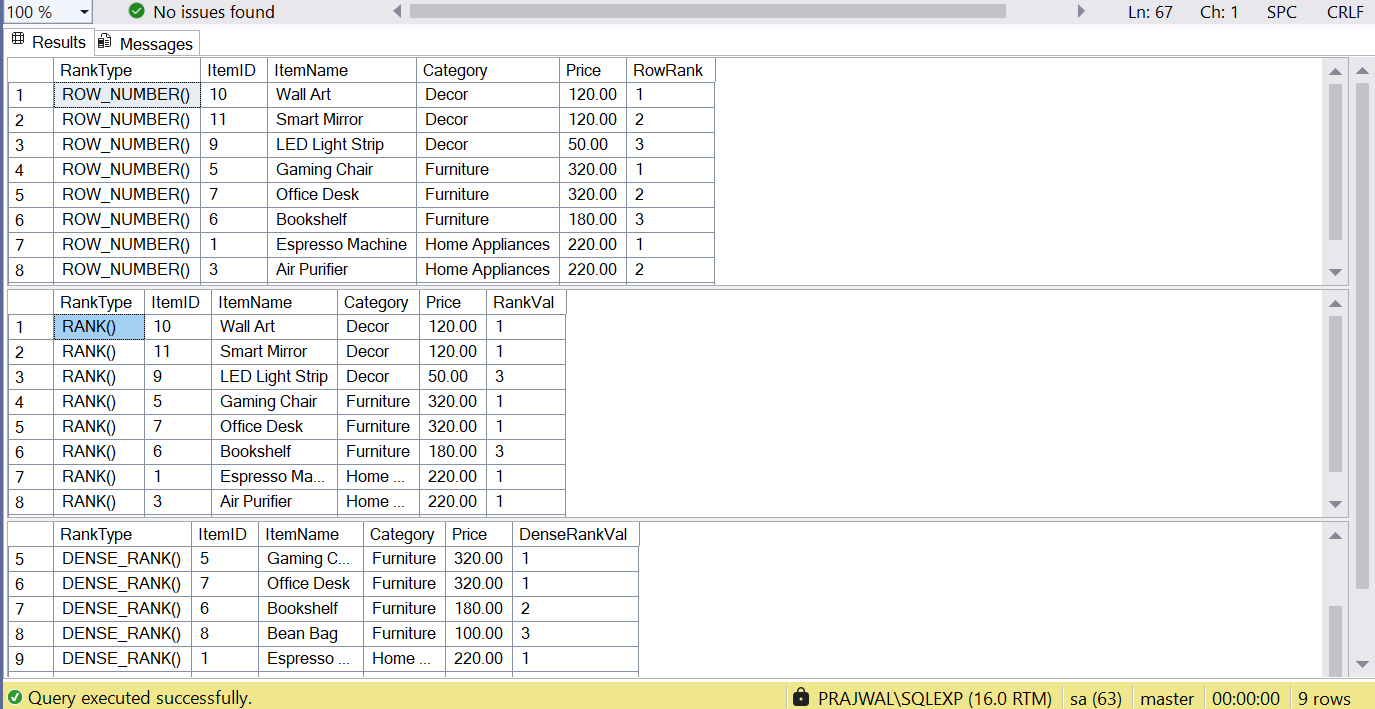
SELECT 'DENSE\_RANK()' AS RankType, \*

FROM DenseRanked

WHERE DenseRankVal <= 3

ORDER BY Category, DenseRankVal;

GO

****

Employee Management System SQL Exercises

## Exercise 1: Create a Stored Procedure

-- Drop tables if they already exist

IF OBJECT\_ID('Employees', 'U') IS NOT NULL DROP TABLE Employees;

IF OBJECT\_ID('Departments', 'U') IS NOT NULL DROP TABLE Departments;

GO

-- Create Departments table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

-- Create Employees table

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY IDENTITY(1,1),

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

GO

-- Insert new Departments

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'Engineering'),

(2, 'Sales'),

(3, 'Support'),

(4, 'Design');

GO

-- Insert new Employees

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

('Aarav', 'Sharma', 1, 7200.00, '2022-03-01'),

('Meera', 'Singh', 2, 5000.00, '2021-06-15'),

('Rohan', 'Verma', 3, 4500.00, '2023-01-10'),

('Isha', 'Patel', 4, 5800.00, '2020-12-05');

GO

-- Drop existing procedure if it exists

IF OBJECT\_ID('sp\_GetEmployeesByDepartment', 'P') IS NOT NULL

DROP PROCEDURE sp\_GetEmployeesByDepartment;

GO

-- Create procedure to get employees by department

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

E.EmployeeID,

E.FirstName,

E.LastName,

D.DepartmentName,

E.Salary,

E.JoinDate

FROM Employees E

INNER JOIN Departments D ON E.DepartmentID = D.DepartmentID

WHERE E.DepartmentID = @DepartmentID;

END;

GO

-- Drop existing procedure if it exists

IF OBJECT\_ID('sp\_InsertEmployee', 'P') IS NOT NULL

DROP PROCEDURE sp\_InsertEmployee;

GO

-- Create procedure to insert new employee

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;

GO

-- Insert new employee using stored procedure

EXEC sp\_InsertEmployee

@FirstName = 'Prajwal',

@LastName = 'Jha',

@DepartmentID = 2,

@Salary = 6200.00,

@JoinDate = '2024-09-01';

GO

-- Get employees from a specific department (e.g., Sales - ID 2)

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 2;

GO

-- Final select: View all employees with department names

SELECT

E.EmployeeID,

E.FirstName,

E.LastName,

D.DepartmentName,

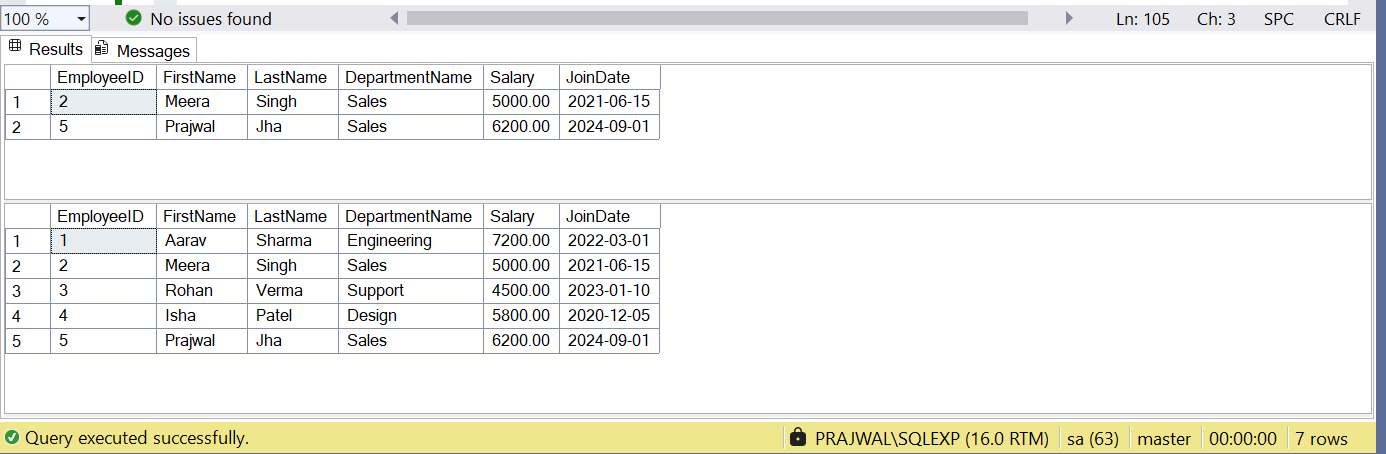
E.Salary,

E.JoinDate

FROM Employees E

INNER JOIN Departments D ON E.DepartmentID = D.DepartmentID;

GO

****

## Exercise 5: Return Data from a Stored Procedure

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 2;