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.

SELECT ProductID, ProductName, Category, Price,

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

FROM Products;

SELECT ProductID, ProductName, Category, Price,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS price\_rank

FROM Products;

SELECT ProductID, ProductName, Category, Price,

DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS dense\_price\_rank

FROM Products;

WITH RankedProducts AS (

SELECT ProductID, ProductName, Category, Price,

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

FROM Products

)

SELECT \*

FROM RankedProducts

WHERE row\_num <= 3;

Exercise 2: Create a Stored Procedure Goal: Create a stored procedure to retrieve employee details by department. Steps: 1. Define the stored procedure with a parameter for DepartmentID. 2. Write the SQL query to select employee details based on the DepartmentID. 3. Create a stored procedure named `sp\_InsertEmployee` with the following code:

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT EmployeeID, FirstName, LastName, DepartmentID, Position, HireDate

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 2;

CREATE PROCEDURE sp\_InsertEmployee

@FirstName NVARCHAR(50),

@LastName NVARCHAR(50),

@DepartmentID INT,

@Position NVARCHAR(50),

@HireDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Position, HireDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Position, @HireDate);

END;

EXEC sp\_InsertEmployee

@FirstName = 'Alice',

@LastName = 'Johnson',

@DepartmentID = 3,

@Position = 'Analyst',

@HireDate = '2024-05-10';

Exercise 5: Return Data from a Stored Procedure Goal: Create a stored procedure that returns the total number of employees in a department. Steps: 1. Define the stored procedure with a parameter for DepartmentID. 2. Write the SQL query to count the number of employees in the specified department. 3. Save the stored procedure by executing the Stored procedure content.

We’ll use a parameter to accept the department ID.

Use COUNT(\*) to get the total number of employees in the specified department.

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

-- Create or alter the stored procedure (if updating)

EXEC sp\_CountEmployeesByDepartment @DepartmentID = 2;

If department 2 has 5 employees, the result will be:

| **TotalEmployees** |
| --- |

5