**Advanced SQL Exercises for Online Retail Store**

Exercise 1. Ranking and Window Functions

Creating the Database:

CREATE DATABASE OnlineRetailDB;

GO

USE OnlineRetailDB;

-- Customers

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name NVARCHAR(100),

Region NVARCHAR(50)

);

-- Products

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName NVARCHAR(100),

Category NVARCHAR(50),

Price DECIMAL(10, 2)

);

-- Orders

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- OrderDetails

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)

);

INSERT INTO Customers VALUES

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

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

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

INSERT INTO Products VALUES

(1, 'Phone', 'Electronics', 500),

(2, 'Shoes', 'Fashion', 80),

(3, 'Laptop', 'Electronics', 1000);

INSERT INTO Orders VALUES

(101, 1, '2025-01-10'),

(102, 2, '2025-01-12'),

(103, 1, '2025-01-14');

INSERT INTO OrderDetails VALUES

(1, 101, 1, 2),

(2, 101, 2, 1),

(3, 102, 3, 1),

(4, 103, 1, 1);

Answer Query:

USE OnlineRetailDB;

GO

-- ROW\_NUMBER

SELECT

Category,

ProductName,

Price,

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

FROM Products;

-- RANK

SELECT

Category,

ProductName,

Price,

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

FROM Products;

-- DENSE\_RANK

SELECT

Category,

ProductName,

Price,

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

FROM Products;

-- Top 3 per category using ROW\_NUMBER

WITH RankedProducts AS (

SELECT \*,

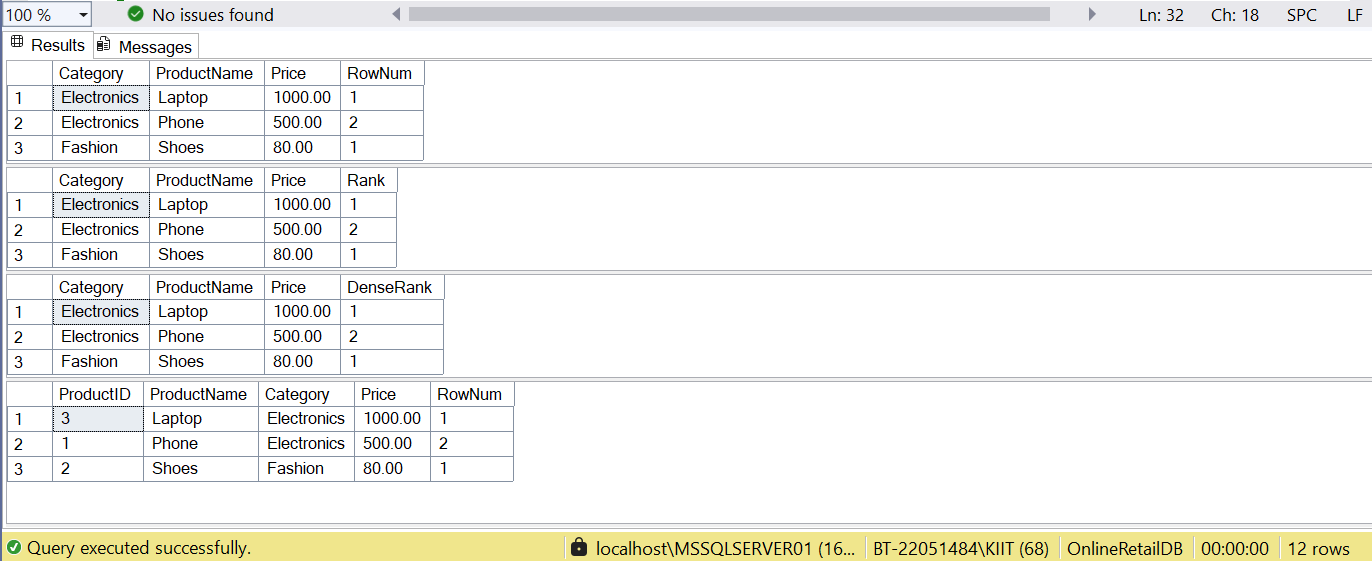
ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM Products

)

SELECT \* FROM RankedProducts WHERE RowNum <= 3;

OUTPUT:



**SQL Exercise - Stored procedure**

Exercise 1: Create a Stored Procedure

-- Step 0: Drop existing tables if they exist

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

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

-- Step 1: Create Schema

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

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

);

-- Step 2: Insert Sample Data

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

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

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

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

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

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

-- Step 3: Stored Procedure to Retrieve Employees by DepartmentID

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

DROP PROCEDURE sp\_GetEmployeesByDepartment;

GO

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DeptID 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 = @DeptID;

END;

GO

-- Step 4: Stored Procedure to Insert New Employee

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

DROP PROCEDURE sp\_InsertEmployee;

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;

GO

-- Step 5: Test Execution

-- Insert a new employee

EXEC sp\_InsertEmployee

@FirstName = 'Alex',

@LastName = 'Reed',

@DepartmentID = 3,

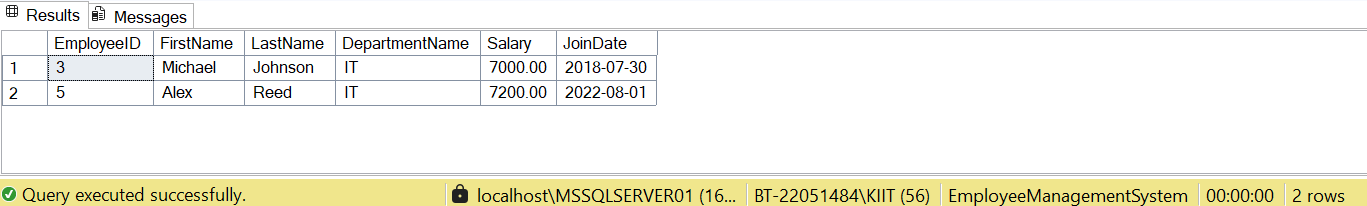
@Salary = 7200.00,

@JoinDate = '2022-08-01';

-- Retrieve employees in IT (DepartmentID = 3)

EXEC sp\_GetEmployeesByDepartment @DeptID = 3;

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

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

-- Step 0: Drop tables if they exist (for isolated execution)

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

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

-- Step 1: Create Tables

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

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

);

-- Step 2: Insert Sample Data

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

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');

-- Step 3: Drop the procedure if it already exists

IF OBJECT\_ID('sp\_CountEmployeesByDepartment') IS NOT NULL

DROP PROCEDURE sp\_CountEmployeesByDepartment;

GO

-- Step 4: Create the Stored Procedure

CREATE PROCEDURE sp\_CountEmployeesByDepartment

@DeptID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DeptID;

END;

GO

-- Step 5: Execute the Procedure (Examples)

PRINT '--- Total Employees in HR (Dept 1) ---';

EXEC sp\_CountEmployeesByDepartment @DeptID = 1;

PRINT '--- Total Employees in IT (Dept 3) ---';

EXEC sp\_CountEmployeesByDepartment @DeptID = 3;

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.