**Week 2\_ Advanced SQL**

***SQL Exercise: Advanced SQL Exercises for Online Retail Store***

***Exercise 1****: Ranking and Window Functions*

Code:

-- Top 3 most expensive products in each category using ROW\_NUMBER

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

) Ranked

WHERE RN <= 3;

-- Using RANK

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

) Ranked

WHERE Rnk <= 3;

-- Using DENSE\_RANK

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

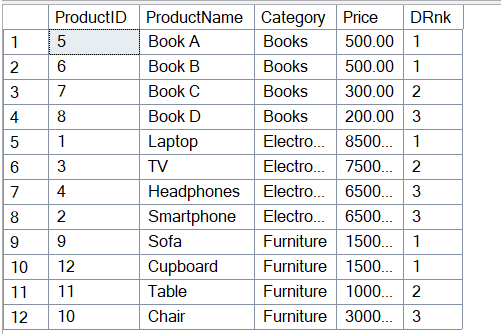
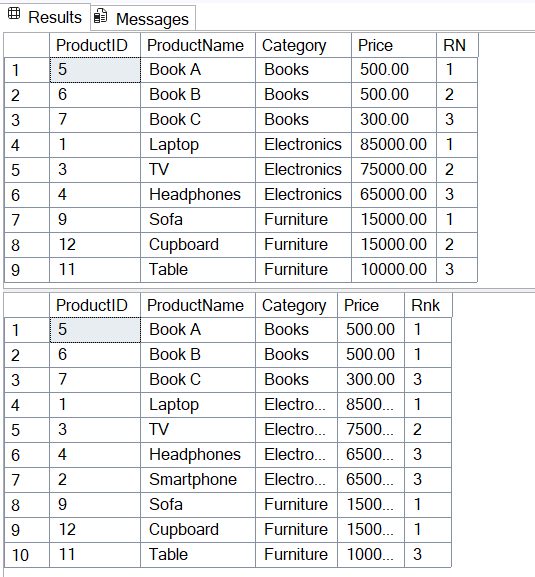
DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DRnk

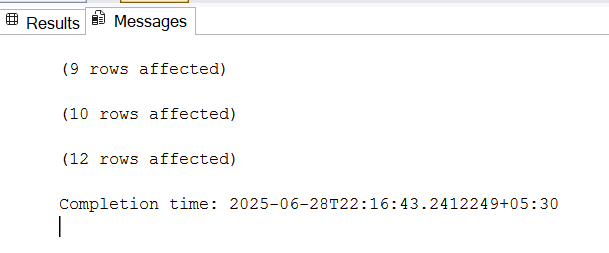
FROM Products

) Ranked

WHERE DRnk <= 3;

Output:





**SQL Exercise : Index**

***-- 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.*

Code:

-- 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

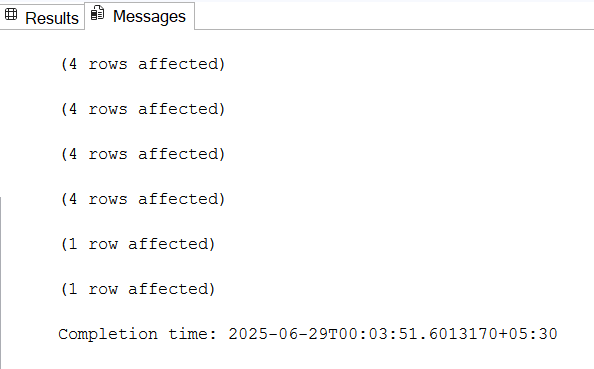
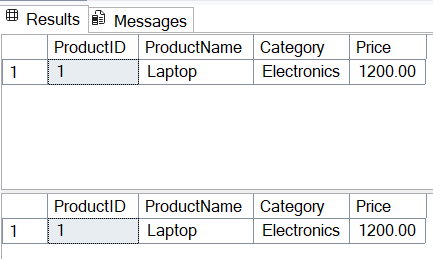
CREATE NONCLUSTERED INDEX IX\_Products\_ProductName

ON Products(ProductName);

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

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

Output:



*--* ***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.*

Code:

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

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

-- Step 2: Create a NON-clustered index on OrderDate (since one clustered index already exists)

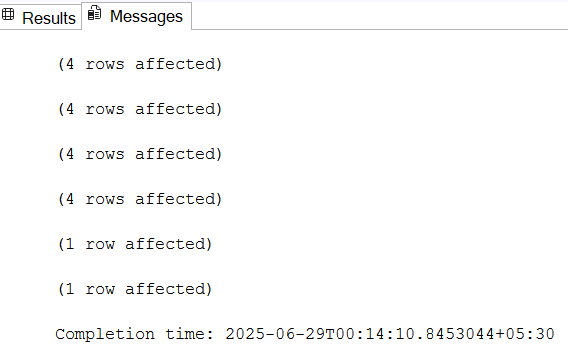
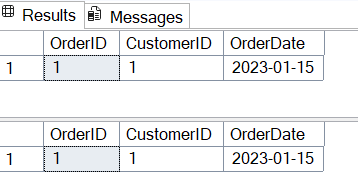
CREATE NONCLUSTERED INDEX IX\_Orders\_OrderDate

ON Orders(OrderDate);

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

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

Output:



-- ***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.*

Code:

-- 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 non-clustered index

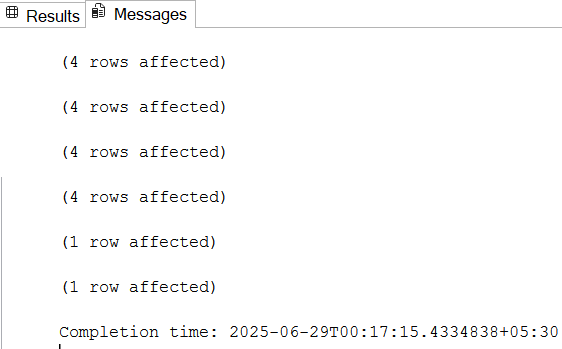
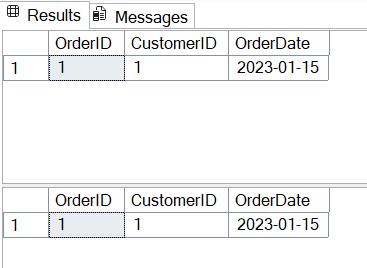
CREATE NONCLUSTERED INDEX IX\_Orders\_CustomerID\_OrderDate

ON Orders(CustomerID, OrderDate);

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

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

Output:



**SQL Exercise : Stored Procedure**

**Employee Management System SQL Exercises**

***Exercise 1****: Create a Stored Procedure*

*Goal: Create a stored procedure to retrieve employee details by department.*

Code:

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

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

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

);

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');

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

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

EXEC sp\_InsertEmployee

@FirstName = 'Alex',

@LastName = 'Reed',

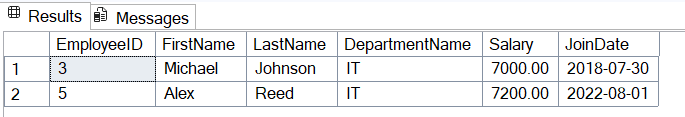
@DepartmentID = 3,

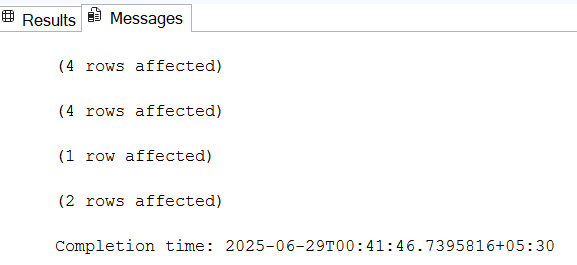
@Salary = 7200.00,

@JoinDate = '2022-08-01';

EXEC sp\_GetEmployeesByDepartment @DeptID = 3;

Output:



****

***Exercise 4****: Execute a Stored Procedure*

*Goal: Execute the stored procedure to retrieve employee details for a specific department.*

Code:-

-- Clean Up: Drop Tables if They Exist

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

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

-

-- Create Tables

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

);

-- Insert Data into Tables

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'),

('Alex', 'Reed', 3, 7200.00, '2022-08-01');

-- Exercise 4: Create and Execute Procedure to Get Employee Details by Department

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;

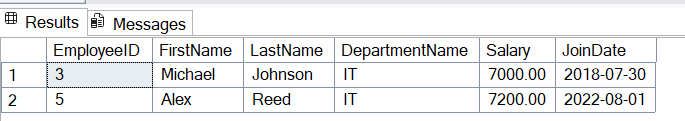
GO

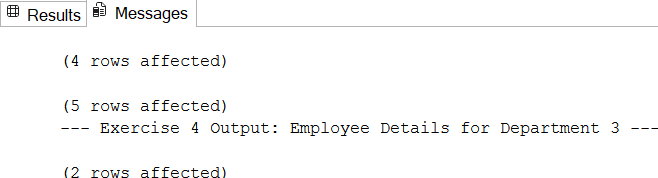
-- Execute Exercise 4 Procedure (Output: Employee details)

PRINT '--- Exercise 4 Output: Employee Details for Department 3 ---';

EXEC sp\_GetEmployeesByDepartment @DeptID = 3;

Output:





***Exercise 5:*** *Return Data from a Stored Procedure Goal: Create a stored procedure that returns the total number of employees in a department.*

Code:

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

DROP PROCEDURE sp\_GetEmployeeCountByDepartment;

GO

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DeptID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DeptID;

END;

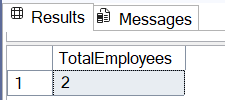
GO

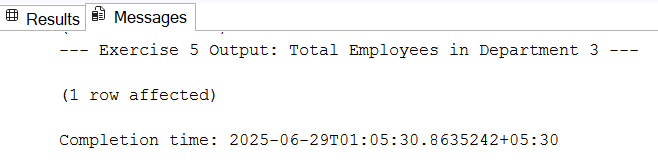
-- Execute Exercise 5 Procedure (Output: Total employee count)

PRINT '--- Exercise 5 Output: Total Employees in Department 3 ---';

EXEC sp\_GetEmployeeCountByDepartment @DeptID = 3;

Output:





| **SQL Exercise - Functions**  **Employee Management System - SQL Exercises** |
| --- |

***Exercise 7****: Return Data from a Scalar Function Goal: Return the annual salary for a specific employee using `fn\_CalculateAnnualSalary`.*

Code:

-- Step 1: Create the database

CREATE DATABASE EmployeeManagementSystem;

GO

-- Step 2: Use the database

USE EmployeeManagementSystem;

GO

-- Step 3: Create Departments table

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

-- Step 4: 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

-- Step 5: Insert sample data into Departments

INSERT INTO Departments (DepartmentID, DepartmentName)

VALUES

(1, 'HR'),

(2, 'IT'),

(3, 'Finance');

GO

-- Step 6: Insert sample data into Employees

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'),

('Bob', 'Johnson', 3, 5500.00, '2021-07-01');

GO

-- Step 7: Create Scalar Function to calculate annual salary

CREATE FUNCTION fn\_CalculateAnnualSalary (@EmpID INT)

RETURNS DECIMAL(12,2)

AS

BEGIN

DECLARE @AnnualSalary DECIMAL(12,2);

SELECT @AnnualSalary = Salary \* 12

FROM Employees

WHERE EmployeeID = @EmpID;

RETURN @AnnualSalary;

END;

GO

-- Step 8: Test the function for EmployeeID = 1

SELECT dbo.fn\_CalculateAnnualSalary(1) AS AnnualSalary;

Output:

