Exercise - 1

Ranking window system:

DROP TABLE IF EXISTS Products;

-- Step 1: Create Products Table

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

-- Step 2: Insert Sample Products

INSERT INTO Products VALUES

(1, 'iPhone 13', 'Mobiles', 799),

(2, 'Samsung Galaxy S21', 'Mobiles', 699),

(3, 'OnePlus 9', 'Mobiles', 729),

(4, 'Google Pixel 6', 'Mobiles', 699),

(5, 'MacBook Pro', 'Laptops', 1299),

(6, 'Dell XPS 13', 'Laptops', 999),

(7, 'HP Spectre x360', 'Laptops', 999),

(8, 'Lenovo Yoga', 'Laptops', 949),

(9, 'Sony WH-1000XM4', 'Accessories', 299),

(10, 'Bose QC 45', 'Accessories', 299),

(11, 'JBL Flip 5', 'Accessories', 149),

(12, 'Anker Soundcore', 'Accessories', 99);

-- Step 3: Use ROW\_NUMBER to get top 3 products per category

SELECT \*

FROM (

SELECT \*,

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

FROM Products

) AS Ranked

WHERE RowNum <= 3;

-- Step 4: Use RANK to get top 3 ranked products per category

SELECT \*

FROM (

SELECT \*,

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

FROM Products

) AS Ranked

WHERE RankNum <= 3;

-- Step 5: Use DENSE\_RANK to get top 3 ranked products per category

SELECT \*

FROM (

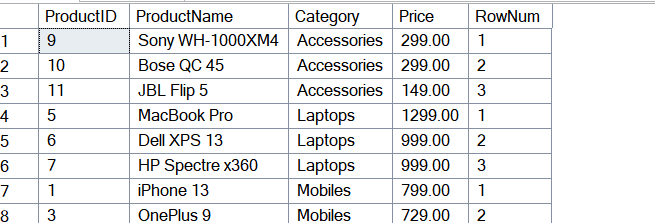
SELECT \*,

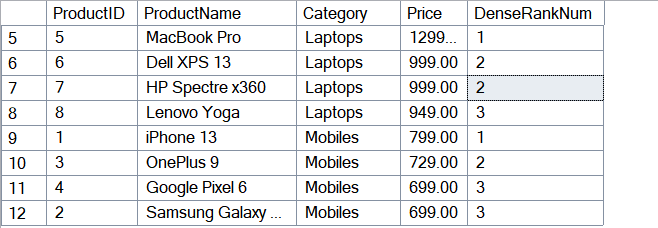
DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum

FROM Products

) AS Ranked

WHERE DenseRankNum <= 3;





USE pratisthadb;

-- Enable statistics to see execution time and IO

SET STATISTICS TIME ON;

SET STATISTICS IO ON;

-- Step 1: Query BEFORE index creation

PRINT 'Query execution BEFORE creating index:';

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

-- Step 2: Create the non-clustered index

PRINT 'Creating non-clustered index...';

CREATE NONCLUSTERED INDEX IX\_Products\_ProductName

ON Products (ProductName);

-- Step 3: Query AFTER index creation

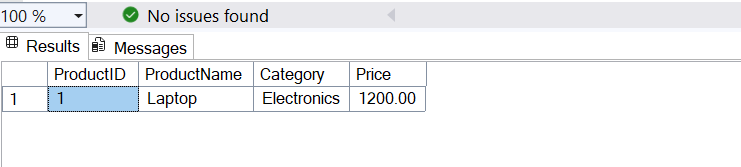
PRINT 'Query execution AFTER creating index:';

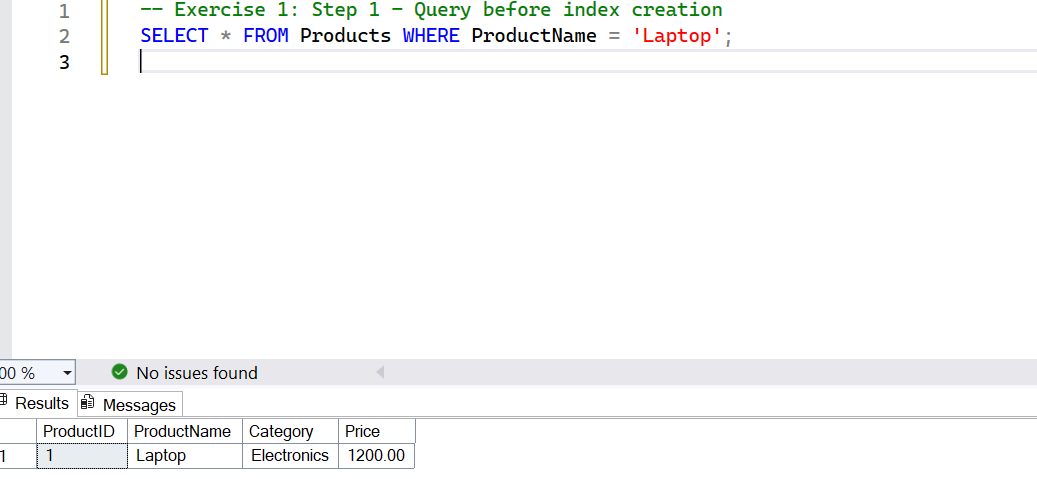
SELECT \* FROM Products WHERE ProductName = 'Laptop';

-- Turn off statistics

SET STATISTICS TIME OFF;

SET STATISTICS IO OFF;





SELECT

ProductID,

ProductName,

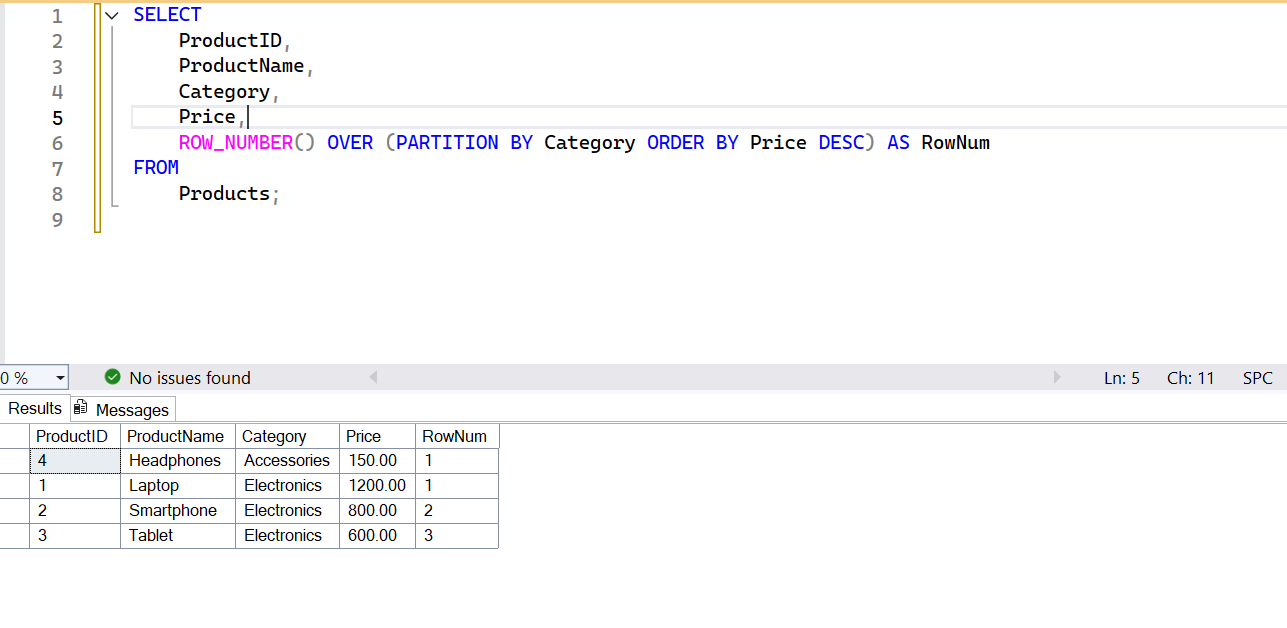
Category,

Price,

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

FROM

Products;



SELECT

ProductID,

ProductName,

Category,

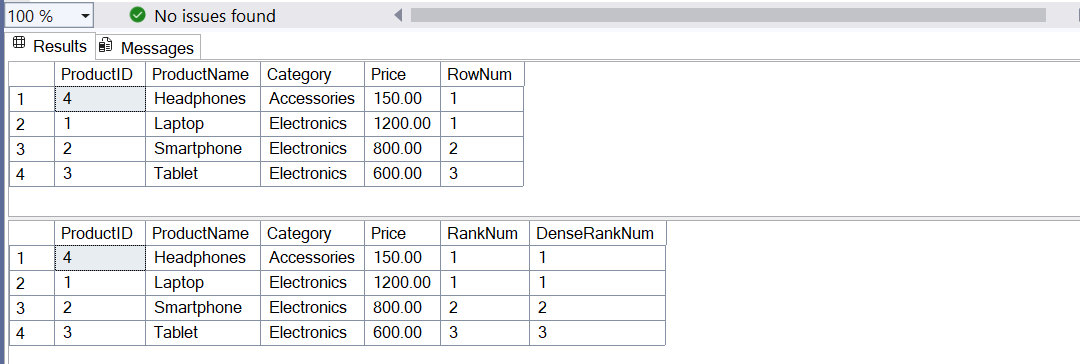
Price,

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

ProductID,

ProductName,

Category,

Price

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

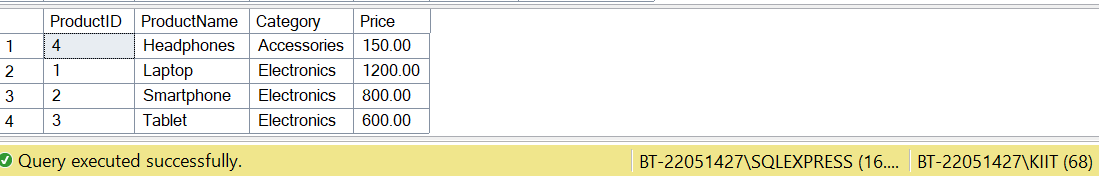
ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM

Products

) AS Ranked

WHERE RowNum <= 3;



Exercise 2:

-- Step 1: Create the Database

CREATE DATABASE EmployeeManagement;

GO

-- Step 2: Use the Database

USE EmployeeManagement;

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,

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, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

GO

-- Step 6: Insert Sample Data into Employees

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

-- Step 7: Create Stored Procedure to Retrieve Employees by Department

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

DROP PROCEDURE sp\_GetEmployeesByDepartment;

GO

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary,

JoinDate

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

GO

-- Step 8: Create Stored Procedure to Insert a New Employee

IF OBJECT\_ID('sp\_InsertEmployee', 'P') 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 9: Test sp\_GetEmployeesByDepartment (Example: IT Department = 3)

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;

GO

-- Step 10: Test sp\_InsertEmployee (Insert new employee into Finance)

EXEC sp\_InsertEmployee

@FirstName = 'Alice',

@LastName = 'Walker',

@DepartmentID = 2,

@Salary = 6200.00,

@JoinDate = '2022-05-01';

GO

-- Step 11: Check updated data

SELECT \* FROM Employees;

GO

-- Drop the procedure if it already exists

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

DROP PROCEDURE sp\_GetEmployeeCountByDepartment;

GO

-- Create the stored procedure

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT COUNT(\*) AS EmployeeCount

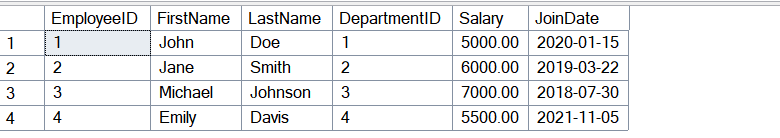
FROM Employees

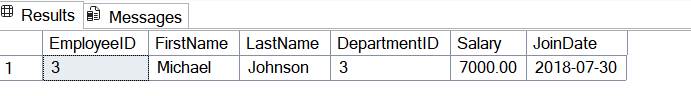
WHERE DepartmentID = @DepartmentID;

END;

GO

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 3;





Exercise 3:

CREATE PROCEDURE dbo.sp\_CheckDepartment

@DepartmentID INT

AS

BEGIN

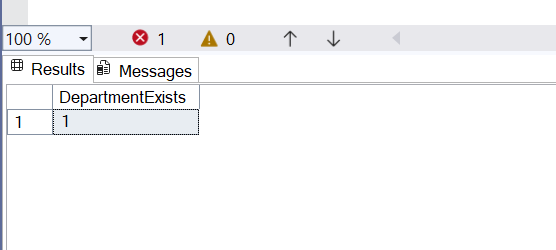
IF EXISTS (SELECT 1 FROM Departments WHERE DepartmentID = @DepartmentID)

RETURN 1; -- Department exists

ELSE

RETURN 0; -- Department does not exist

END;



EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 3;



**MOQ**

**using System.Net;**

**using System.Net.Mail;**

**namespace CustomerCommLib**

**{**

**public interface IMailSender**

**{**

**bool SendMail(string toAddress, string message);**

**}**

**public class MailSender : IMailSender**

**{**

**public bool SendMail(string toAddress, string message)**

**{**

**MailMessage mail = new MailMessage();**

**SmtpClient SmtpServer = new SmtpClient("smtp.gmail.com");**

**mail.From = new MailAddress("your\_email\_address@gmail.com");**

**mail.To.Add(toAddress);**

**mail.Subject = "Test Mail";**

**mail.Body = message;**

**SmtpServer.Port = 587;**

**SmtpServer.Credentials = new NetworkCredential("username", "password");**

**SmtpServer.EnableSsl = true;**

**SmtpServer.Send(mail);**

**return true; // Return true after sending**

**}**

**}**

**public class CustomerComm**

**{**

**IMailSender \_mailSender;**

**public CustomerComm(IMailSender mailSender)**

**{**

**\_mailSender = mailSender;**

**}**

**public bool SendMailToCustomer()**

**{**

**// define message and mail address**

**\_mailSender.SendMail("cust123@abc.com", "Some Message");**

**return true;**

**}**

**}**

**}**

**using NUnit.Framework;**

**using Moq;**

**using CustomerCommLib;**

**namespace CustomerComm.Tests**

**{**

**[TestFixture]**

**public class CustomerCommTests**

**{**

**private Mock<IMailSender> \_mockMailSender;**

**private CustomerComm \_customerComm;**

**[OneTimeSetUp]**

**public void Init()**

**{**

**// Create mock of IMailSender**

**\_mockMailSender = new Mock<IMailSender>();**

**// Configure mock: SendMail accepts any string arguments and returns true**

**\_mockMailSender.Setup(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>())).Returns(true);**

**// Inject mock into CustomerComm**

**\_customerComm = new CustomerComm(\_mockMailSender.Object);**

**}**

**[Test]**

**public void SendMailToCustomer\_ShouldReturnTrue()**

**{**

**// Act**

**var result = \_customerComm.SendMailToCustomer();**

**// Assert**

**Assert.IsTrue(result);**

**// Verify that SendMail was called exactly once with any string parameters**

**\_mockMailSender.Verify(m => m.SendMail(It.IsAny<string>(), It.IsAny<string>()), Times.Once);**

**}**

**}**

**}**