**DN 4.0 Dotnet FSE**

**Name: Sparsh Guha**

**Superset ID:6361106**

**Week 2-**

1. **SQL Exercise - Advanced concepts**

**Exercise 1: Ranking and Window Functions**

**Code:**

WITH RankedProducts AS (

SELECT

Category,

ProductName,

Price,

ROW\_NUMBER() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RowNum,

RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS Rank,

DENSE\_RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS DenseRank

FROM Products

)

SELECT

Category,

ProductName,

Price,

RowNum,

Rank,

DenseRank

FROM RankedProducts

WHERE DenseRank <= 3;

**Schema Code:**

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

INSERT INTO Products (ProductID, ProductName, Category, Price) VALUES

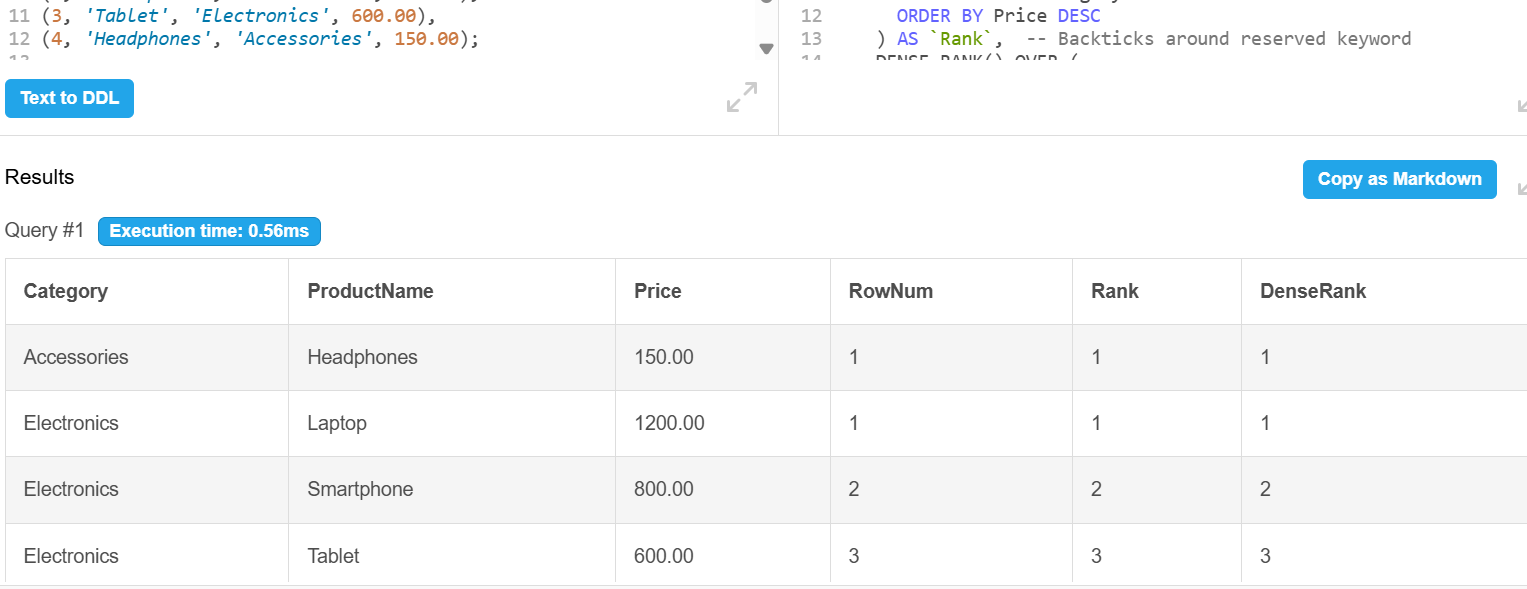
(1, 'Laptop', 'Electronics', 1200.00),

(2, 'Smartphone', 'Electronics', 800.00),

(3, 'Tablet', 'Electronics', 600.00),

(4, 'Headphones', 'Accessories', 150.00);

**Output:**



1. **SQL Exercise - Stored procedure**

**Exercise 1: Create a Stored Procedure**

**Schema Code:**

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,

Salary DECIMAL(10,2),

JoinDate DATE

);

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');

**Part 1:**

## **Get Employees by Department (simulate stored procedure for DepartmentID = 2):**

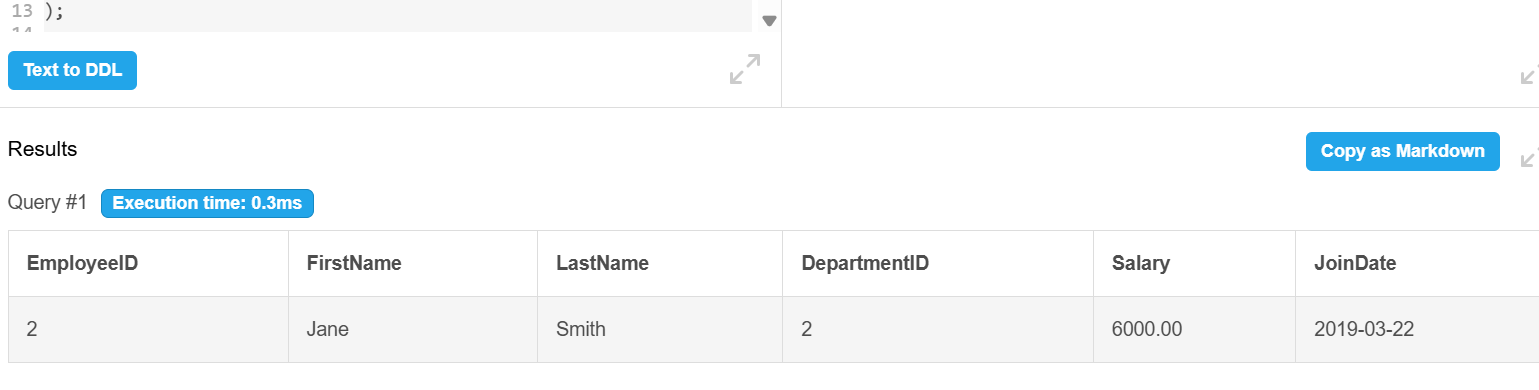
**Query Code:**

SELECT EmployeeID, FirstName, LastName, DepartmentID, Salary, JoinDate

FROM Employees

WHERE DepartmentID = 2;

**Output:**



## Part 2: ****Insert a New Employee****

**CODE:**

**Schema:**

**CREATE TABLE Departments (**

**DepartmentID INT PRIMARY KEY,**

**DepartmentName VARCHAR(100)**

**);**

**CREATE TABLE Employees (**

**EmployeeID INT AUTO\_INCREMENT PRIMARY KEY, -- Added AUTO\_INCREMENT**

**FirstName VARCHAR(50),**

**LastName VARCHAR(50),**

**DepartmentID INT,**

**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');**

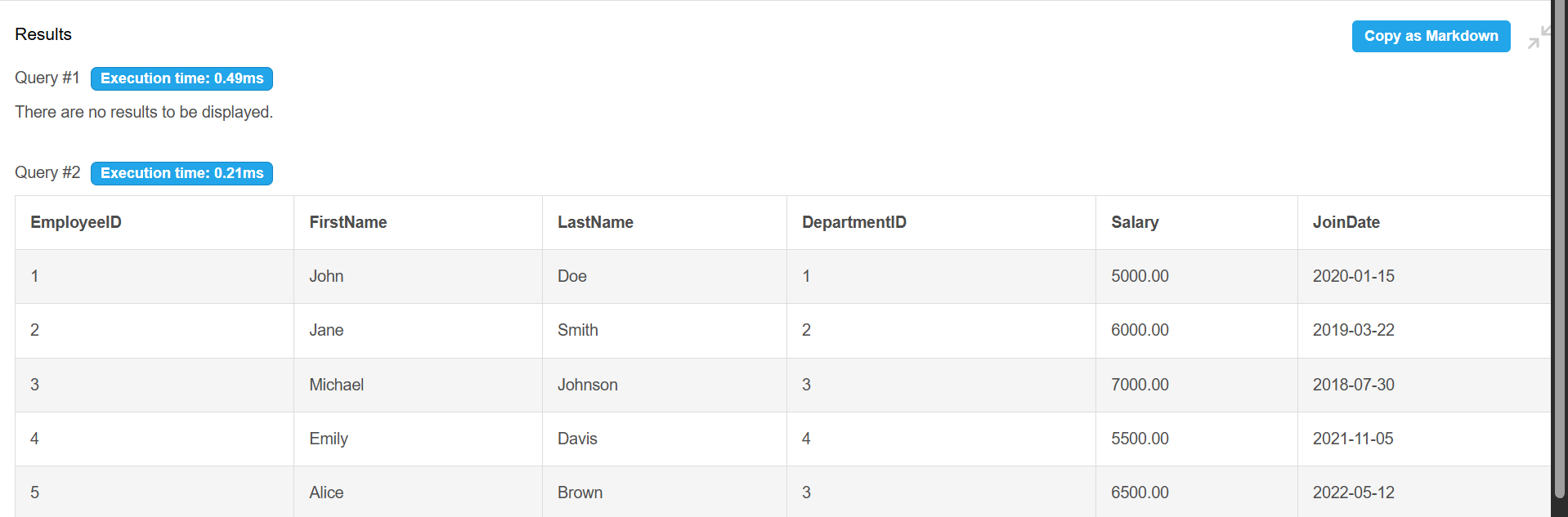
****SQL Query:****

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

**VALUES ('Alice', 'Brown', 3, 6500.00, '2022-05-12');**

**SELECT \* FROM Employees;**

****OUTPUT:****

****

#4 SQL Exercise - Stored procedure

**Exercise 5: Return Data from a Stored Procedure (Using SSMS now)**

**Code:**

USE TestDB;

GO

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Departments')

BEGIN

    CREATE TABLE Departments (

        DepartmentID INT PRIMARY KEY,

        DepartmentName VARCHAR(100)

    );

END

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Employees')

BEGIN

    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

    );

END

GO

IF NOT EXISTS (SELECT 1 FROM Departments)

BEGIN

    INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

    (1, 'HR'),

    (2, 'Finance'),

    (3, 'IT'),

    (4, 'Marketing');

END

IF NOT EXISTS (SELECT 1 FROM Employees)

BEGIN

    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');

END

GO

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

    @DepartmentID INT

AS

BEGIN

    SELECT COUNT(\*) AS EmployeeCount

    FROM Employees

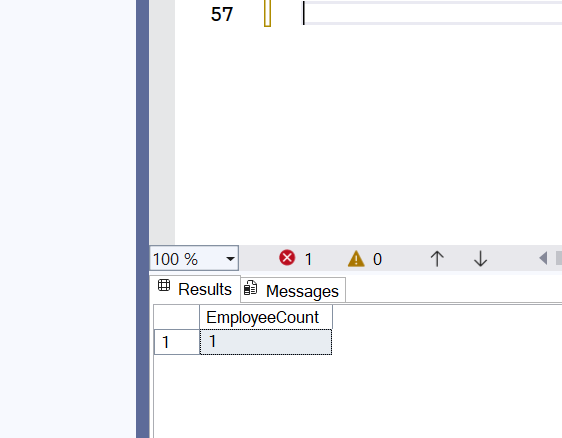
    WHERE DepartmentID = @DepartmentID;

END

GO

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 2;

**OUTPUT:**

****

**Additional Exercises:**

1. **SQL Exercise - Index**

**Code**: (Given)  
  
-- Database Schema

CREATE TABLE Customers (

    CustomerID INT PRIMARY KEY,

    Name VARCHAR(100),

    Region VARCHAR(50)

);

CREATE TABLE Products (

    ProductID INT PRIMARY KEY,

    ProductName VARCHAR(100),

    Category VARCHAR(50),

    Price DECIMAL(10, 2)

);

CREATE TABLE Orders (

    OrderID INT PRIMARY KEY,

    CustomerID INT,

    OrderDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

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)

);

-- Sample Data

INSERT INTO Customers (CustomerID, Name, Region) VALUES

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

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

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

(4, 'David', 'West');

INSERT INTO Products (ProductID, ProductName, Category, Price) VALUES

(1, 'Laptop', 'Electronics', 1200.00),

(2, 'Smartphone', 'Electronics', 800.00),

(3, 'Tablet', 'Electronics', 600.00),

(4, 'Headphones', 'Accessories', 150.00);

INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES

(1, 1, '2023-01-15'),

(2, 2, '2023-02-20'),

(3, 3, '2023-03-25'),

(4, 4, '2023-04-30');

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES

(1, 1, 1, 1),

(2, 2, 2, 2),

(3, 3, 3, 1),

(4, 4, 4, 3);

-- 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.

-- 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

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

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

-- 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.

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

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

-- Step 2: Create a clustered index on OrderDate

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

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

-- 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.

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

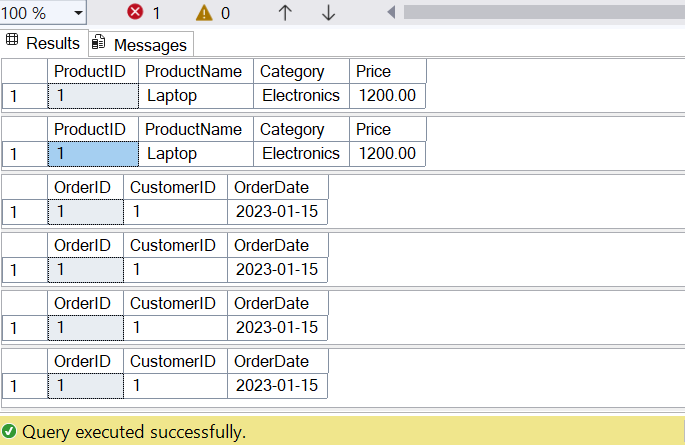
SELECT \* FROM Orders WHERE CustomerID = 1 AND OrderDate = '2023-01-15';

-- Step 2: Create a composite index on CustomerID and OrderDate

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

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

**OUTPUT:**



**#5. SQL Exercise - Functions**

**Exercise 7: Return Data from a Scalar Function**

**Code:**USE TestDB;

GO

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Departments')

BEGIN

    CREATE TABLE Departments (

        DepartmentID INT PRIMARY KEY,

        DepartmentName VARCHAR(100)

    );

END

GO

IF NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'Employees')

BEGIN

    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

    );

END

GO

IF NOT EXISTS (SELECT 1 FROM Departments)

BEGIN

    INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

    (1, 'HR'),

    (2, 'IT'),

    (3, 'Finance');

END

GO

IF NOT EXISTS (SELECT 1 FROM Employees)

BEGIN

    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');

END

GO

IF OBJECT\_ID('dbo.fn\_CalculateAnnualSalary', 'FN')

    DROP FUNCTION dbo.fn\_CalculateAnnualSalary;

GO

CREATE FUNCTION dbo.fn\_CalculateAnnualSalary (@EmployeeID INT)

RETURNS DECIMAL(18,2)

AS

BEGIN

    RETURN (SELECT Salary \* 12 FROM Employees WHERE EmployeeID = @EmployeeID);

END;

GO

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

    DROP PROCEDURE dbo.sp\_GetEmployeeCountByDepartment;

GO

CREATE PROCEDURE dbo.sp\_GetEmployeeCountByDepartment

    @DepartmentID INT

AS

BEGIN

    SELECT COUNT(\*) AS EmployeeCount

    FROM Employees

    WHERE DepartmentID = @DepartmentID;

END;

GO

SELECT

    EmployeeID,

    FirstName,

    LastName,

    dbo.fn\_CalculateAnnualSalary(1) AS AnnualSalary

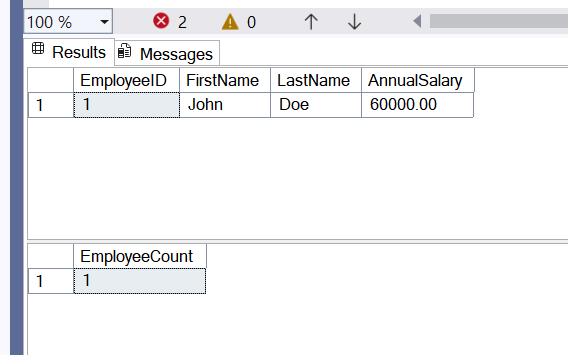
FROM Employees

WHERE EmployeeID = 1;

GO

EXEC dbo.sp\_GetEmployeeCountByDepartment @DepartmentID = 2;

**OUTPUT:**

****

**#4 SQL Exercise - Stored procedure**

**Exercise 4: Execute a Stored Procedure**

**Code:**

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

);

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');

GO

CREATE PROCEDURE sp\_GetEmployeesByDepartment

    @DepartmentID INT

AS

BEGIN

    SELECT

        EmployeeID,

        FirstName,

        LastName,

        DepartmentID,

        Salary,

        JoinDate

    FROM Employees

    WHERE DepartmentID = @DepartmentID;

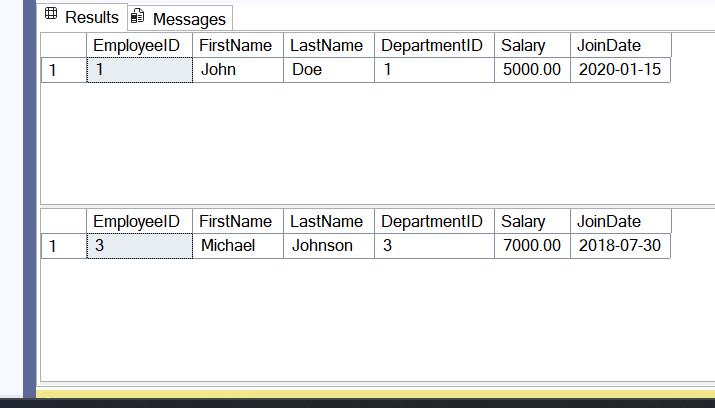
END;

GO

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 1;

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;

**OUTPUT:**

****

1. **NUnit-Handson (VS CODE)**

**INPUT:**

**Calculator.cs**namespace CalcLibrary

{

    public class Calculator

    {

        public int Add(int a, int b) => a + b;

        public int Subtract(int a, int b) => a - b;

        public int Multiply(int a, int b) => a \* b;

        public int Divide(int a, int b)

        {

            if (b == 0)

                throw new DivideByZeroException("Cannot divide by zero.");

            return a / b;

        }

    }

}

CalculatorTests.cs

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

    [TestFixture]

    public class CalculatorTests

    {

        private Calculator \_calc = null!;

        [SetUp]

        public void Setup()

        {

            \_calc = new Calculator();

        }

        [TearDown]

        public void Cleanup()

        {

        }

        [Test]

        [TestCase(2, 3, 5)]

        [TestCase(-1, 1, 0)]

        [TestCase(0, 0, 0)]

        public void Add\_AddsNumbers\_ReturnsExpected(int a, int b, int expected)

        {

            int result = \_calc.Add(a, b);

            Assert.That(result, Is.EqualTo(expected));

        }

        [Test]

        [TestCase(5, 3, 2)]

        [TestCase(10, 7, 3)]

        [TestCase(0, 0, 0)]

        public void Subtract\_SubtractsNumbers\_ReturnsExpected(int a, int b, int expected)

        {

            int result = \_calc.Subtract(a, b);

            Assert.That(result, Is.EqualTo(expected));

        }

        [Test]

        [TestCase(2, 3, 6)]

        [TestCase(-1, 1, -1)]

        [TestCase(0, 5, 0)]

        public void Multiply\_MultipliesNumbers\_ReturnsExpected(int a, int b, int expected)

        {

            int result = \_calc.Multiply(a, b);

            Assert.That(result, Is.EqualTo(expected));

        }

        [Test]

        [TestCase(6, 3, 2)]

        [TestCase(10, 2, 5)]

        public void Divide\_DividesNumbers\_ReturnsExpected(int a, int b, int expected)

        {

            int result = \_calc.Divide(a, b);

            Assert.That(result, Is.EqualTo(expected));

        }

        [Test]

        public void Divide\_ByZero\_ThrowsException()

        {

            Assert.Throws<DivideByZeroException>(() => \_calc.Divide(5, 0));

        }

        [Test]

        [Ignore("Test not implemented yet.")]

        public void Modulo\_NotImplemented\_Ignored()

        {

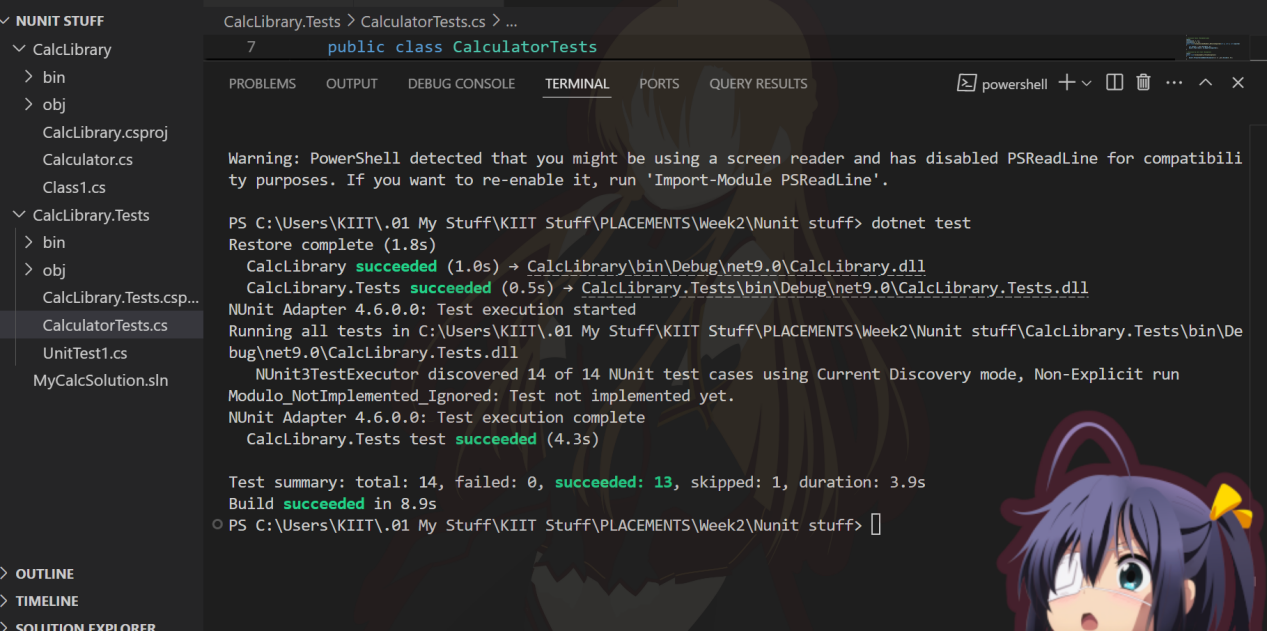
            Assert.Fail("This test is intentionally ignored.");

        }

    }

}

**OUTPUT:**



1. Moq-Handson
2. Write Testable Code with Moq

**INPUT:**

CustomerCommTests.cs

using CustomerCommLib;

using Moq;

using NUnit.Framework;

namespace CustomerComm.Tests

{

    [TestFixture]

    public class CustomerCommTests

    {

        private CustomerComm \_customerComm;

        private Mock<IMailSender> \_mockMailSender;

        [SetUp]

        public void Setup()

        {

            \_mockMailSender = new Mock<IMailSender>();

            \_mockMailSender.Setup(m =>

                m.SendMail(It.IsAny<string>(), It.IsAny<string>())

            ).Returns(true);

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

        }

        [Test]

        public void SendMailToCustomer\_WhenCalled\_ReturnsTrue()

        {

            bool result = \_customerComm.SendMailToCustomer();

            Assert.IsTrue(result);

            \_mockMailSender.Verify(m =>

                m.SendMail(It.IsAny<string>(), It.IsAny<string>()), Times.Once);

        }

    }

}

CustomerComm.cs

namespace CustomerCommLib

{

    public class CustomerComm

    {

        private readonly IMailSender \_mailSender;

        public CustomerComm(IMailSender mailSender)

        {

            \_mailSender = mailSender;

        }

        public bool SendMailToCustomer()

        {

            // Actual implementation would use real data

            return \_mailSender.SendMail("cust123@abc.com", "Welcome Message");

        }

    }

}

IMAILSENDER.CS

using System.Net;

using System.Net.Mail;

namespace CustomerCommLib

{

    public class MailSender : IMailSender

    {

        public bool SendMail(string toAddress, string message)

        {

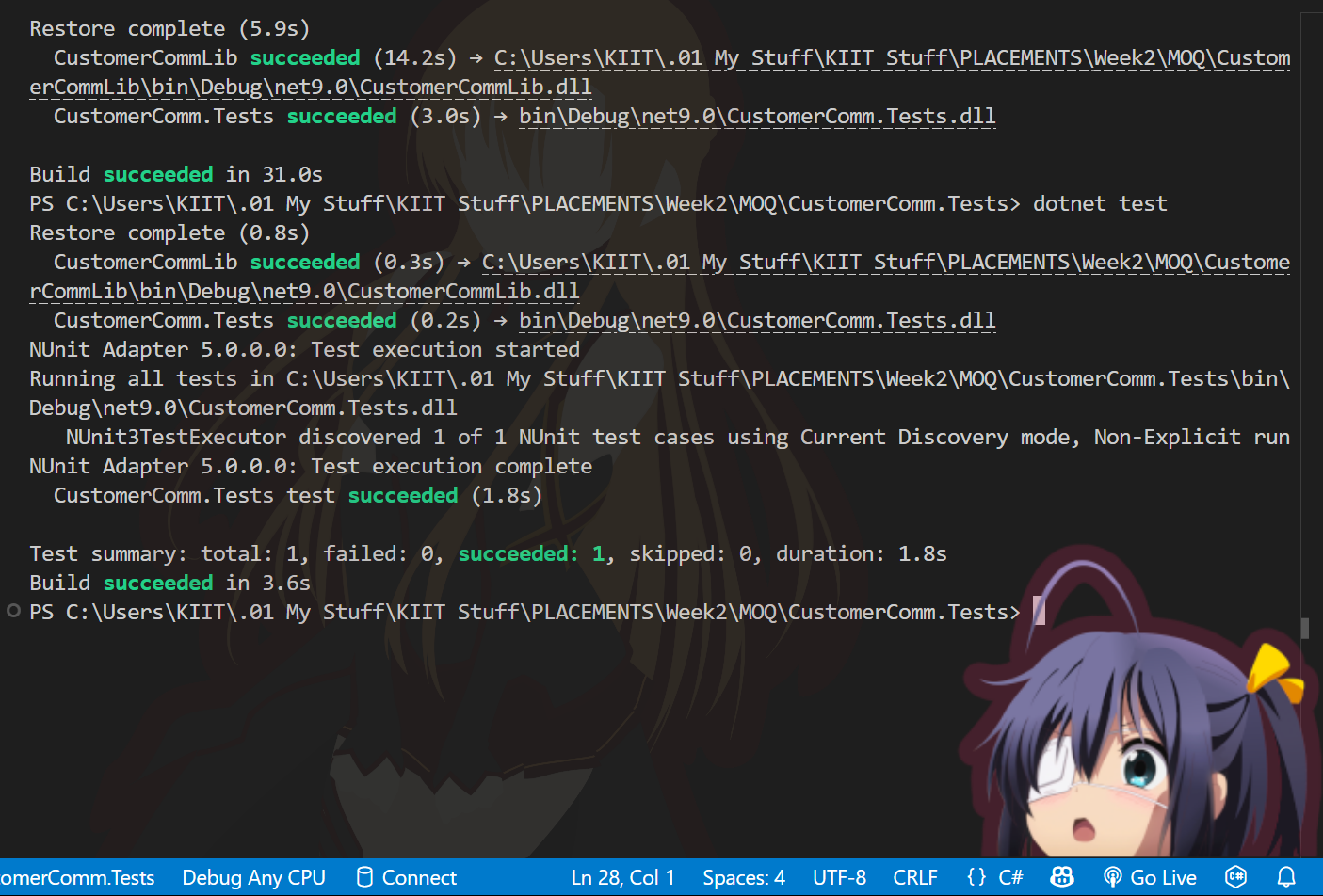
            return true;

        }

    }

}

**OUTPUT:**

****