**ADVANCED SQL SERVER**

1. **Ranking and Window Function**

**Creating and Inserting data in table:**

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', 1000),

(2, 'TV', 'Electronics', 900),

(3, 'Headphones', 'Electronics', 900),

(4, 'Smartphone', 'Electronics', 800);

**Row\_Number()**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

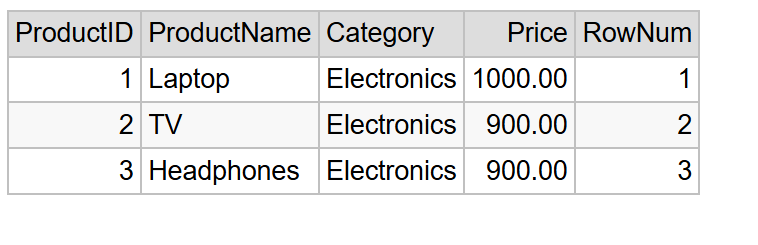
Price,

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

FROM Products

) AS Ranked

WHERE RowNum <= 3;



**Rank()**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

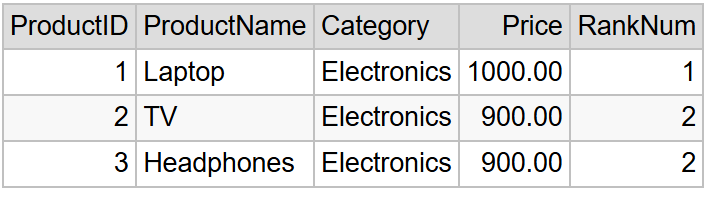
Price,

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

FROM Products

) AS Ranked

WHERE RankNum <= 3;



**Dense\_Rank()**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

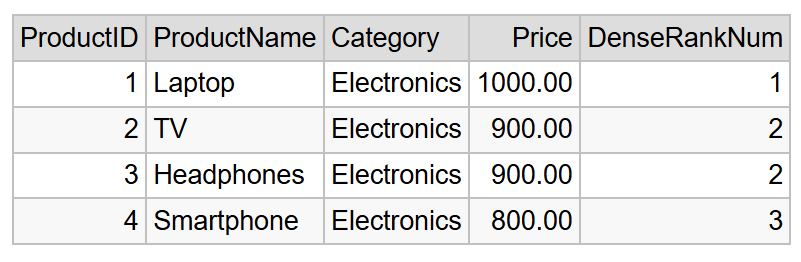
Price,

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

FROM Products

) AS Ranked

WHERE DenseRankNum <= 3;



1. **Storage Procedures**

**Getting Employees by Department**

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID 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 = @DepartmentID;

END;

**Insert new employee**

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;

**Inserting and Retrieving Data**

EXEC sp\_InsertEmployee

@FirstName = 'David',

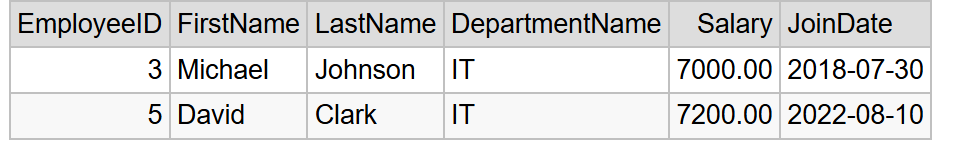
@LastName = 'Clark',

@DepartmentID = 3,

@Salary = 7200.00,

@JoinDate = '2022-08-10';

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;



1. **Return Data from Stored Procedures**

**Defining the Stored Procedure**

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

COUNT(\*) AS EmployeeCount

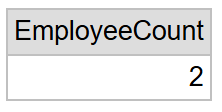
FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

**Calling the Stored Procedure**

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 3;



**NUnit and Moq**

**Unit Testing for a Calculator Application**

**Creating a Test Case**

using NUnit.Framework;

using CalcLib;

namespace calctests

{

    [TestFixture]

    public class CalculatorTests

    {

        private Calculator calc;

        [SetUp]

        public void Setup()

        {

            calc = new Calculator();

        }

        [Test]

        [TestCase(2, 3, 5)]

        [TestCase(-1, -1, -2)]

        [TestCase(0, 0, 0)]

        public void Add\_ShouldReturnCorrectSum(int a, int b, int expected)

        {

            var result = calc.Add(a, b);

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

        }

        [Test]

        [Ignore("This test is intentionally ignored")]

        public void IgnoredTest()

        {

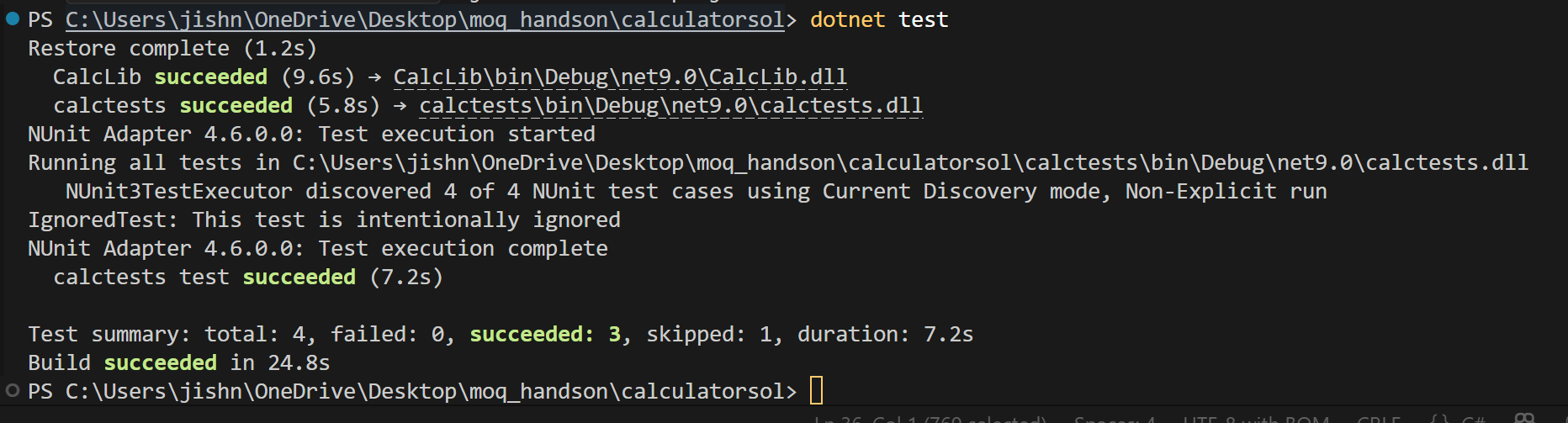
            Assert.Fail("This test should not run");

        }

    }

}

**Output**

****

**Testable Code with Moq**

**MailSender.cs**

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@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;

        }

    }

}

**CustomerComm.cs**

namespace CustomerCommLib

{

    public class CustomerComm

    {

        private readonly IMailSender \_mailSender;

        public CustomerComm(IMailSender mailSender)

        {

            \_mailSender = mailSender;

        }

        public bool SendMailToCustomer()

        {

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

        }

    }

}

**UnitTest1.cs**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerCommTests

{

    public class Tests

    {

        [Test]

        public void SendMailToCustomer\_ShouldReturnTrue\_WhenMailIsSent()

        {

            var mockMailSender = new Mock<IMailSender>();

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

            var customerComm = new CustomerComm(mockMailSender.Object);

            var result = customerComm.SendMailToCustomer();

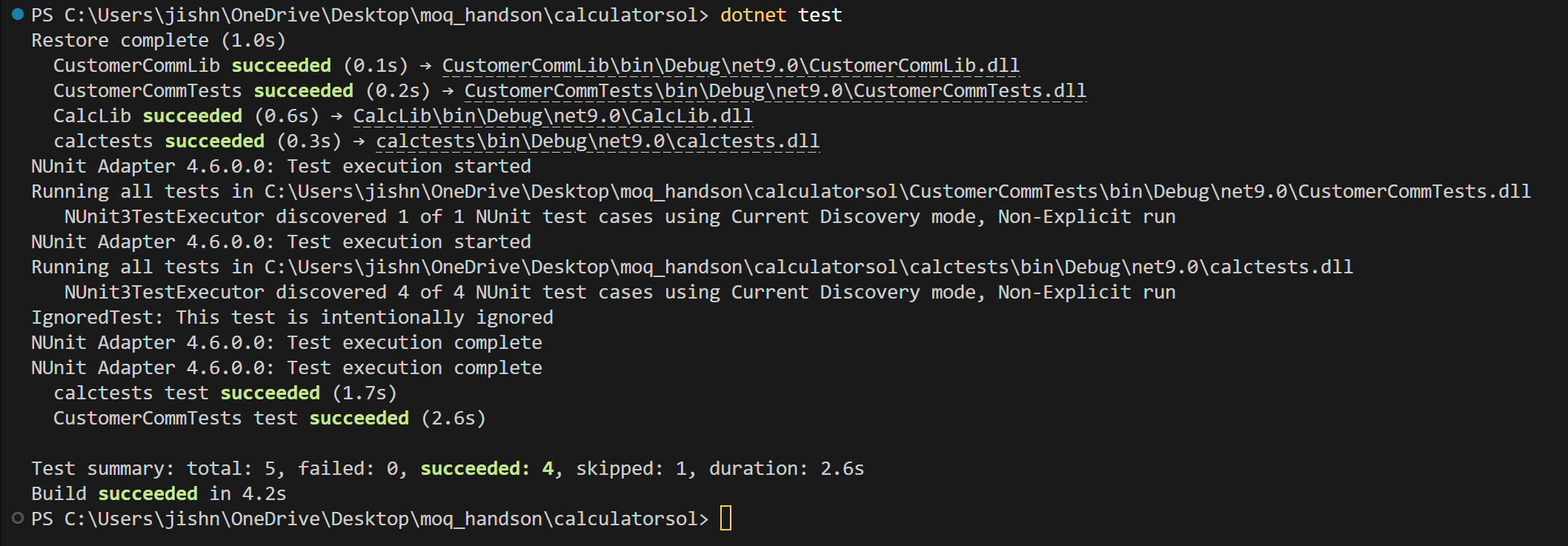
            Assert.That(result, Is.True);

        }

    }

}

**Output**

****