**ADVANCED SQL SERVER**

**1.Ranking and Window Function**

**Creating and Inserting data in table:**

CREATE TABLE Products (

ProductID INT,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10,2)

);

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

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

(2, 'Laptop B', 'Electronics', 1300.00),

(3, 'Laptop C', 'Electronics', 1300.00),

(4, 'Phone A', 'Electronics', 800.00),

(5, 'Chair A', 'Furniture', 150.00),

(6, 'Chair B', 'Furniture', 120.00),

(7, 'Sofa A', 'Furniture', 300.00),

(8, 'Sofa B', 'Furniture', 300.00),

(9, 'Table A', 'Furniture', 200.00),

**Row\_Number()**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

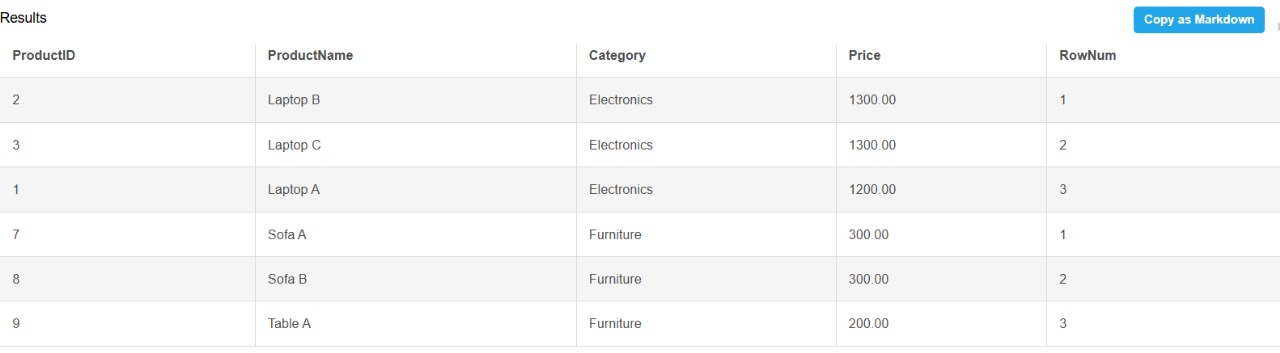
Price,

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,

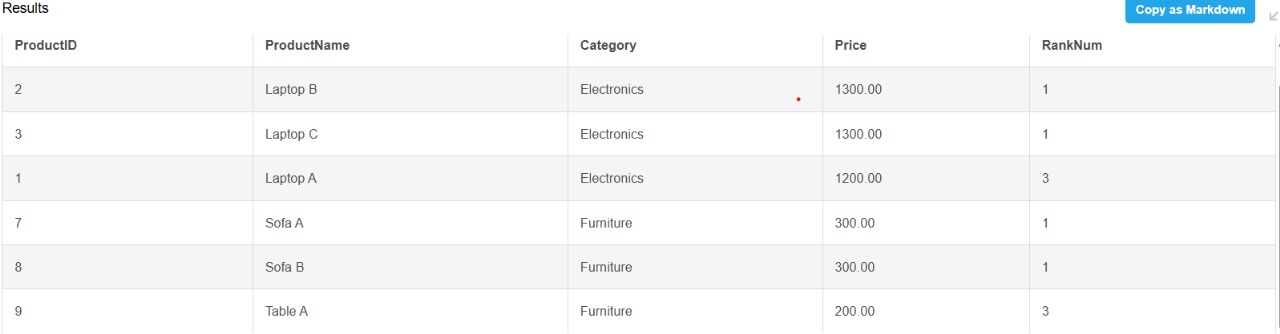
Price,

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

FROM Products

) AS Ranked

WHERE RowNum <= 3;



**Dense\_Rank()**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

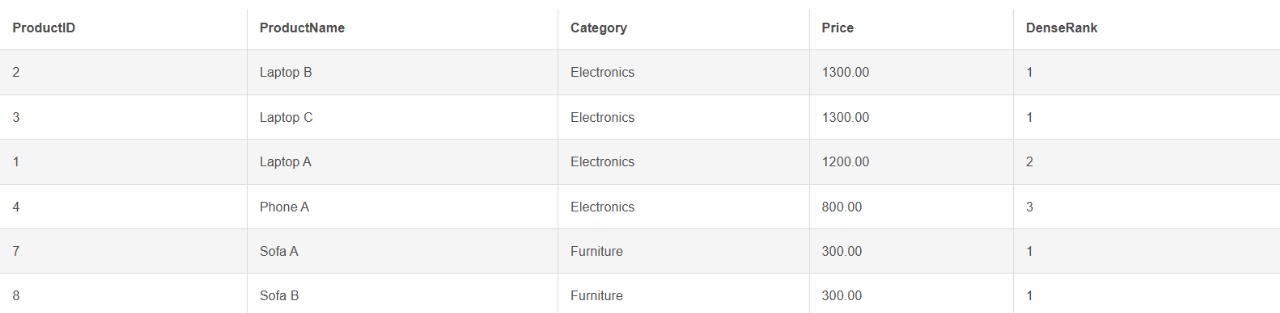
Price,

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

FROM Products

) AS Ranked

WHERE DenseRank <= 3;



**2.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 = &#39;David&#39;,

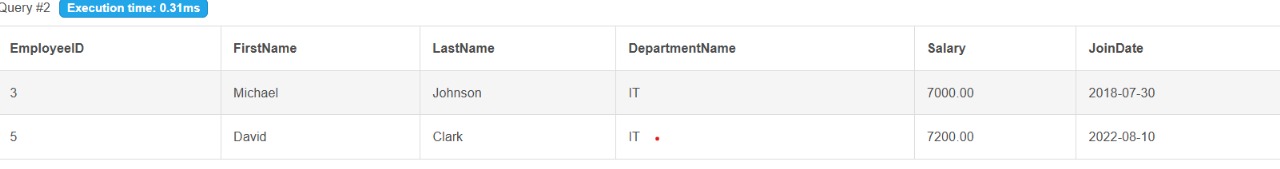
@LastName = &#39;Clark&#39;,

@DepartmentID = 3,

@Salary = 7200.00,

@JoinDate = &#39;2022-08-10&#39;;

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;



**3.Return Data from Stored Procedures**

**Defining the Stored Procedure**

EXEC sp\_InsertEmployee

@FirstName = &#39;David&#39;,

@LastName = &#39;Clark&#39;,

@DepartmentID = 3,

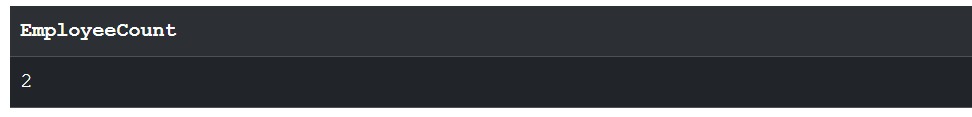
@Salary = 7200.00,

@JoinDate = &#39;2022-08-10&#39;;

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 3;

**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(&quot;This test is intentionally ignored&quot;)]

        public void IgnoredTest()

        {

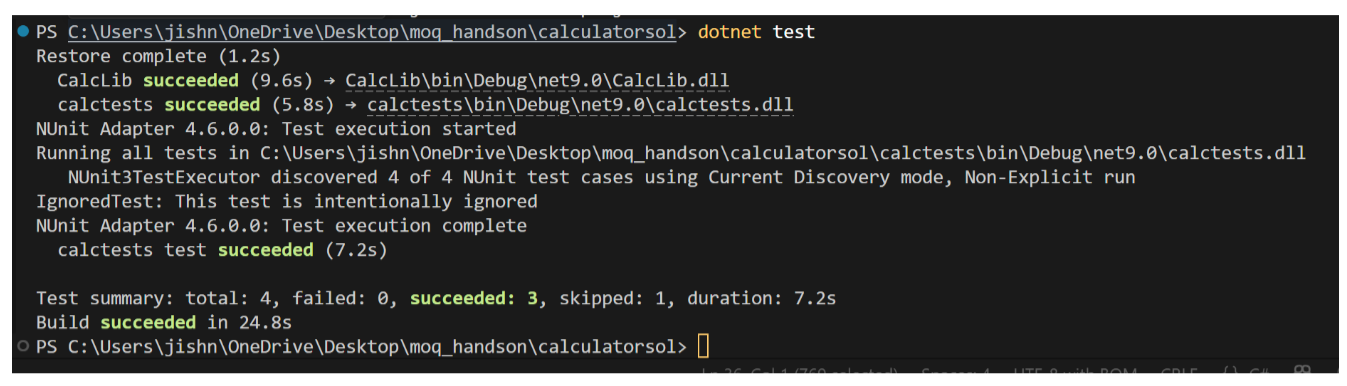
            Assert.Fail(&quot;This test should not run&quot;);

        }

    }

}

**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(&quot;smtp.gmail.com&quot;);

            mail.From = new MailAddress(&quot;your\_email@gmail.com&quot;);

            mail.To.Add(toAddress);

            mail.Subject = &quot;Test Mail&quot;;

            mail.Body = message;

            SmtpServer.Port = 587;

            SmtpServer.Credentials = new NetworkCredential(&quot;username&quot;, &quot;password&quot;);

            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(&quot;cust123@abc.com&quot;, &quot;Some Message&quot;);

        }

    }

}

**UnitTest1.cs**

using NUnit.Framework;

using Moq;

using CustomerCommLib;

namespace CustomerCommTests

{

    public class Tests

    {

        [Test]

        public void SendMailToCustomer\_ShouldReturnTrue\_WhenMailIsSent()

        {

            var mockMailSender = new Mock&lt;IMailSender&gt;();

            mockMailSender.Setup(m =&gt; m.SendMail(It.IsAny&lt;string&gt;(),

It.IsAny&lt;string&gt;())).Returns(true);

            var customerComm = new CustomerComm(mockMailSender.Object);

            var result = customerComm.SendMailToCustomer();

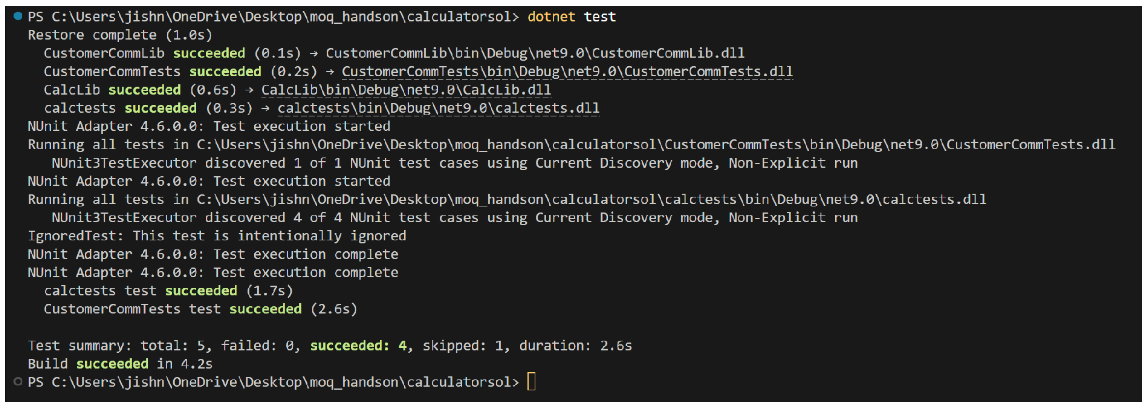
            Assert.That(result, Is.True);

        }

    }

}

**Output**

****