1. Write Testable code with MOQ

**Solution:**

using System;

using System.Net;

using System.Net.Mail;

using NUnit.Framework;

using Moq;

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;

}

}

public class CustomerComm

{

IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

string toAddress = "cust123@abc.com";

string message = "Some Message";

return \_mailSender.SendMail(toAddress, message);

}

}

[TestFixture]

public class CustomerCommTests

{

[Test]

public void SendMailToCustomer\_MailSenderCalled\_ReturnsTrue()

{

var mockMailSender = new Mock<IMailSender>();

mockMailSender

.Setup(sender => sender.SendMail(It.IsAny<string>(), It.IsAny<string>()))

.Returns(true);

var customerComm = new CustomerComm(mockMailSender.Object);

var result = customerComm.SendMailToCustomer();

Assert.That(result, Is.True);

mockMailSender.Verify(sender => sender.SendMail("cust123@abc.com", "Some Message"), Times.Once);

}

}

}

**Output:**

2. Test Fixture and Test

**Solution:**

using System;

using NUnit.Framework;

namespace CalcLibrary

{

public class Calculator

{

public int Add(int n1, int n2) => n1 + n2;

public int Subtract(int n1, int n2) => n1 - n2;

public int Multiply(int n1, int n2) => n1 \* n2;

public int Divide(int n1, int n2) => n1 / n2;

}

[TestFixture]

public class CalculatorTests

{

private Calculator calculator;

[SetUp]

public void Setup()

{

calculator = new Calculator();

}

[TearDown]

public void TearDown()

{

calculator = null;

}

[Test]

[TestCase(2, 3, 5)]

[TestCase(5, 10, 15)]

public void TestAdd(int n1, int n2, int expected)

{

Assert.That(calculator.Add(n1, n2), Is.EqualTo(expected));

}

[Test]

[TestCase(5, 3, 2)]

[TestCase(4, 4, 0)]

public void TestSub(int n1, int n2, int expected)

{

Assert.That(calculator.Subtract(n1, n2), Is.EqualTo(expected));

}

[Test]

[TestCase(10, 10, 100)]

[TestCase(4, 5, 20)]

public void TestMultiply(int n1, int n2, int expected)

{

Assert.That(calculator.Multiply(n1, n2), Is.EqualTo(expected));

}

[Test]

[TestCase(100,10,10)]

[TestCase(14,7,2)]

public void TestDivide(int n1, int n2, int expected)

{

Assert.That(calculator.Divide(n1, n2), Is.EqualTo(expected));

}

}

}

**Output:**

3. Ranking and Window Functions

**Solution:**

create table items (PID int , ProductName varchar(50), Category varchar(50), Price float);

insert into items values(1, 'Laptop', 'Electronics', '50000.00');

insert into items values(2, 'Chocolate', 'Snacks', '100.00');

insert into items values(3, 'Chips', 'Snacks', '50.00');

insert into items values(4, 'Mobile Phone', 'Electronics', '15000.00');

insert into items values(5, 'Sofa', 'Furniture', '57000.00');

insert into items values(6, 'Chair', 'Furniture', '5000.00');

insert into items values(7, 'Biscuit', 'Snacks', '30.00');

insert into items values(8, 'Television', 'Electronics', '100000.00');

insert into items values(9, 'Table', 'Furniture', '7000.00');

WITH RankedProducts AS (SELECT PID, ProductName, Category, Price, ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM items)

SELECT \* FROM RankedProducts WHERE RowNum <= 3;

**Output:**

4. Create a Stored Procedure

**Solution:**

CREATE PROCEDURE EmployeesCount

@DepartmentID INT

AS

BEGIN

SELECT EmployeeID, FirstName, LastName, Salary, JoinDate

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

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

5. Return Data from a Stored Procedure

**Solution:**

GO

CREATE PROCEDURE sp\_employeeCount

@DepartmentID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

GO

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 2;

SELECT \* FROM Employees WHERE DepartmentID = 2;

**Output:**