1. :
2. Ranking and Window Functions
3. Using ROW\_NUMBER() – Unique ranking per category

SELECT

ProductID,

ProductName,

Category,

Price,

ROW\_NUMBER() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RowNum

FROM Products;

1. Using RANK() – Ties get same rank, gaps appear

SELECT

ProductID,

ProductName,

Category,

Price,

RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS RankNum

FROM Products;

1. Using DENSE\_RANK() – Ties share rank, no gaps

SELECT

ProductID,

ProductName,

Category,

Price,

DENSE\_RANK() OVER (

PARTITION BY Category

ORDER BY Price DESC

) AS DenseRankNum

FROM Products;

1. : Create a Stored Procedure

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary,

JoinDate

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

1. :Return Data from a Stored Procedure

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

COUNT(\*) AS TotalEmployees

FROM

Employees

WHERE

DepartmentID = @DepartmentID;

END;

1. . Write Testable Code with Moq

**Project1**

**IMailSender.cs**

namespace CustomerCommLib

{

public interface IMailSender

{

bool SendMail(string toAddress, string message);

}

}

MailSender.cs

using System.Net;

using System.Net.Mail;

namespace CustomerCommLib

{

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;

}

}

}

**CustomerComm.cs**

namespace CustomerCommLib

{

public class CustomerComm

{

private readonly IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

string email = "cust123@abc.com";

string message = "Some Message";

return \_mailSender.SendMail(email, message);

}

}

}

**Project2**

**CustomerCommTests.cs**

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()

{

\_mockMailSender = new Mock<IMailSender>();

// Configure mock to return true for any input

\_mockMailSender

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

.Returns(true);

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

}

[Test]

public void SendMailToCustomer\_ShouldReturnTrue()

{

// Act

var result = \_customerComm.SendMailToCustomer();

// Assert

Assert.IsTrue(result);

}

}

}

