Aman Raj

Superset ID: 6358186  
  
  
 **1: Ranking and Window Functions  
  
Solution:-**

**Using ROW\_NUMBER():**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

) AS RankedProducts

WHERE RowNum <= 3

AND Category IN (

SELECT Category

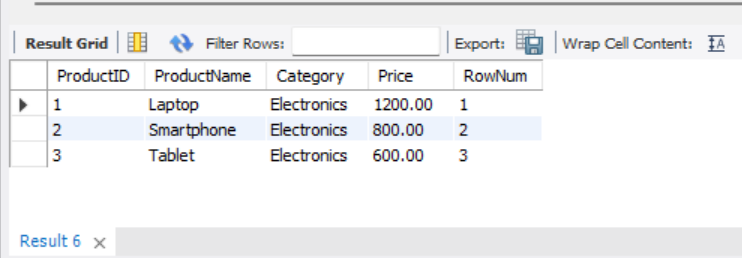
FROM Products

GROUP BY Category

HAVING COUNT(\*) >= 3

)

ORDER BY Category, RowNum;

  
  
  
  
  
**Using RANK():**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

) AS RankedProducts

WHERE RankNum <= 3

AND Category IN (

SELECT Category

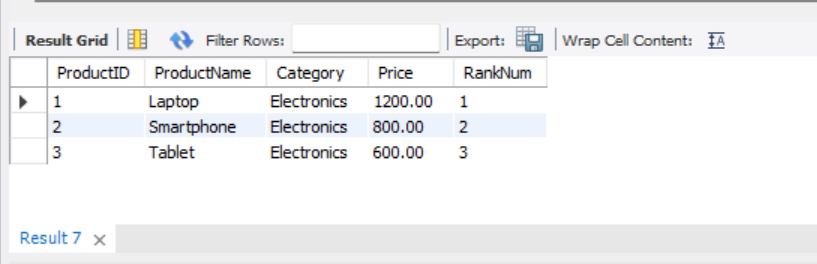
FROM Products

GROUP BY Category

HAVING COUNT(\*) >= 3

)

ORDER BY Category, RankNum;



**Using DENSE\_RANK():**

SELECT \*

FROM (

SELECT

ProductID,

ProductName,

Category,

Price,

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

FROM Products

) AS RankedProducts

WHERE DenseRankNum <= 3

AND Category IN (

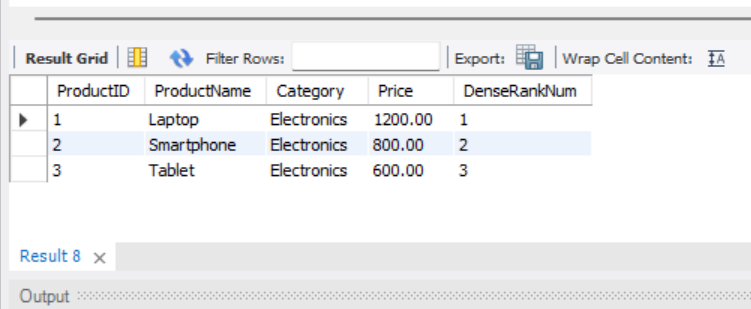
SELECT Category

FROM Products

GROUP BY Category

HAVING COUNT(\*) >= 3

)

ORDER BY Category, DenseRankNum;  
  
  


**2: Create a Stored Procedure**

**Solution:-**

DELIMITER //

CREATE PROCEDURE sp\_InsertEmployee (

IN FirstName VARCHAR(50),

IN LastName VARCHAR(50),

IN DepartmentID INT,

IN Salary DECIMAL(10,2),

IN JoinDate DATE

)

BEGIN

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

VALUES (FirstName, LastName, DepartmentID, Salary, JoinDate);

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE sp\_GetEmployeesByDepartment (

IN DeptID INT

)

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary,

JoinDate

FROM Employees

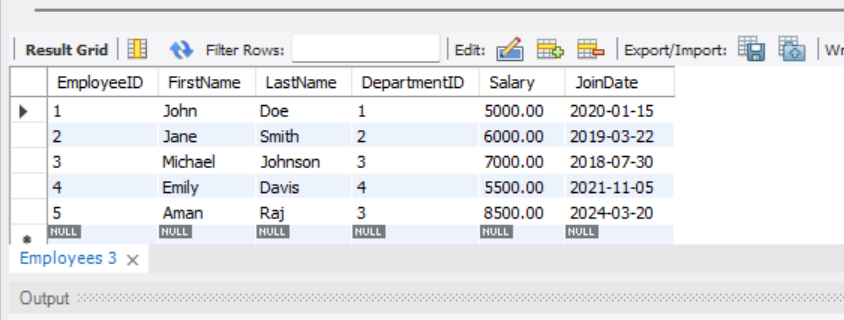
WHERE DepartmentID = DeptID;

END //

DELIMITER ;

CALL sp\_InsertEmployee('Aman', 'Raj', 3, 8500.00, '2024-03-20');

SELECT \* FROM Employees;



**3: Return Data from a Stored Procedure**

**Solution:-**

DELIMITER //

DROP PROCEDURE IF EXISTS sp\_GetEmployeeCountByDepartment;

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment (

IN DeptID INT

)

BEGIN

SELECT COUNT(\*) AS TotalEmployees

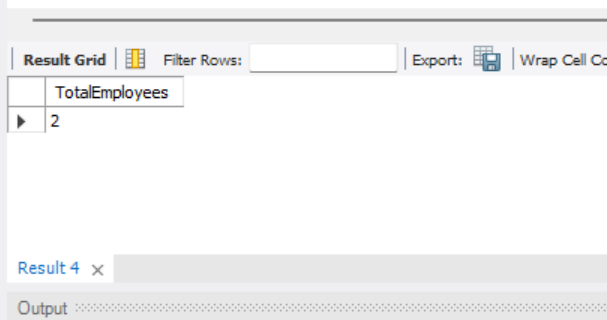
FROM Employees

WHERE DepartmentID = DeptID;

END //

DELIMITER ;

CALL sp\_GetEmployeeCountByDepartment(3);



**4. NUnit-Handson**

**Solution:-**

1. **What is Unit Testing?**

Unit Testing is the process of testing individual units or components of a software (like functions, methods, or classes) in isolation.

* It ensures that a specific section of code (like Add(int a, int b)) works as intended.
* It often uses mocking to simulate dependencies so the unit is tested in isolation

### ****2. Types of Testing:****

**Unit Testing** – Test individual methods or classes.

**Functional Testing** – Test the complete application functionality.

**Automated Testing** – Any test written as code to run automatically.

**Performance Testing** – Test the application under load or stress.

### ****3. Benefits of Automated Testing:****

* Fast and repeatable
* Detect bugs early
* Prevent regression
* Improves code confidence and maintainability
* Enables Continuous Integration / Continuous Deployment (CI/CD)

**Unit Test for Calculator:**

using NUnit.Framework;

using CalcLibrary;

namespace CalcLibrary.Tests

{

[TestFixture]

public class CalculatorTests

{

private SimpleCalculator calc;

[SetUp]

public void SetUp()

{

calc = new SimpleCalculator();

}

[TearDown]

public void TearDown()

{

calc = null;

}

[Test]

[TestCase(2.0, 3.0, 5.0)]

[TestCase(-1.5, 1.5, 0.0)]

[TestCase(0.0, 0.0, 0.0)]

public void Addition\_ValidDoubles\_ReturnsExpectedSum(double a, double b, double expected)

{

double result = calc.Addition(a, b);

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

}

[Test]

[Ignore("Demonstrating the Ignore attribute")]

public void IgnoreExample\_SkippedTest()

{

Assert.Fail("This should be skipped");

}

[Test]

public void Division\_ByZero\_ThrowsArgumentException()

{

Assert.That(

() => calc.Division(5, 0),

Throws.ArgumentException

.With.Message.Contain("Can't be Zero")

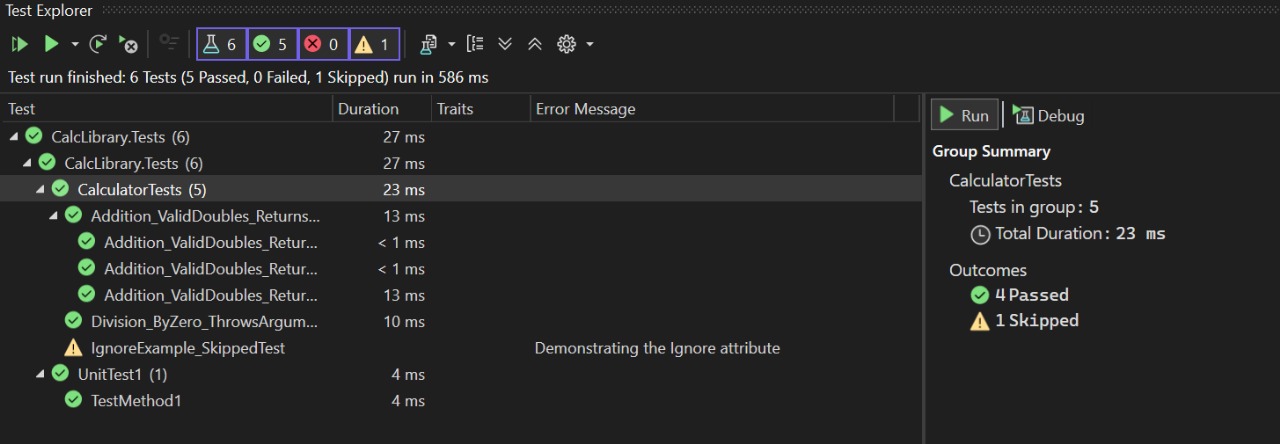
);

}

}

}

**Output:-**

****

1. **Write Testable Code with Moq**

**Solution:-**

**CustomerCommTests.cs**

using Moq;

using NUnit.Framework;

using CustomerCommLib;

namespace CustomerCommLib.Tests

{

[TestFixture]

public class CustomerCommTests

{

private Mock<IMailSender> \_mockMailSender;

private CustomerComm \_customerComm;

[SetUp]

public void SetUp()

{

// Arrange: Create and configure mock before each test

\_mockMailSender = new Mock<IMailSender>();

\_mockMailSender

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

.Returns(true);

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

}

[Test]

public void SendMailToCustomer\_WhenCalled\_ShouldReturnTrueAndInvokeSendMailOnce()

{

// Act

bool result = \_customerComm.SendMailToCustomer();

// Assert

Assert.IsTrue(result, "Expected SendMailToCustomer to return true.");

\_mockMailSender.Verify(

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

Times.Once,

"SendMail should be called exactly once.");

}

}

}

**CustomerComm.cs**

namespace CustomerCommLib

{

public class CustomerComm

{

private readonly IMailSender \_mailSender;

public CustomerComm(IMailSender mailSender)

{

\_mailSender = mailSender;

}

public bool SendMailToCustomer()

{

// In real code you'd build these dynamically

const string to = "cust123@abc.com";

const string msg = "Some Message";

return \_mailSender.SendMail(to, msg);

       }

    }

}

**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)

{

var mail = new MailMessage();

var smtp = new SmtpClient("smtp.gmail.com")

{

Port = 587,

Credentials = new NetworkCredential("username", "password"),

EnableSsl = true

};

mail.From = new MailAddress("your\_email\_address@gmail.com");

mail.To.Add(toAddress);

mail.Subject = "Test Mail";

mail.Body = message;

smtp.Send(mail);

return true;

}

}

}

**Output:-**

