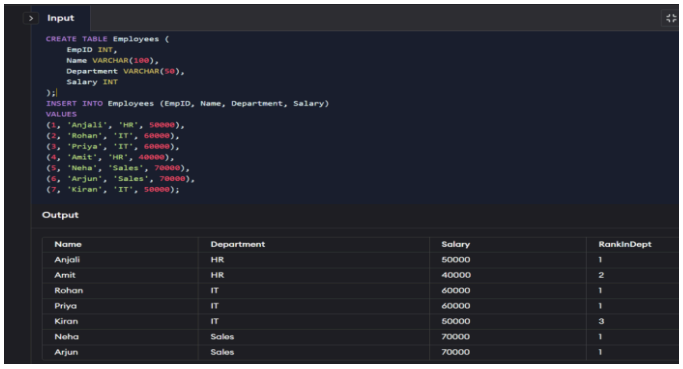
WEEK 2

SUPERSET ID : 6363523
Name : Shalu Kumari
Roll No. : 22052852

Exercise 1: Ranking and Window Functions

```
CREATE TABLE Employees (
    EmpID INT,
    Name VARCHAR(100),
    Department VARCHAR(50),
    Salary INT
);

INSERT INTO Employees VALUES
(1, 'Anjali', 'HR', 50000),
(2, 'Rohan', 'IT', 60000),
(3, 'Priya', 'IT', 60000),
(4, 'Amit', 'HR', 40000),
(5, 'Neha', 'Sales', 70000);
```



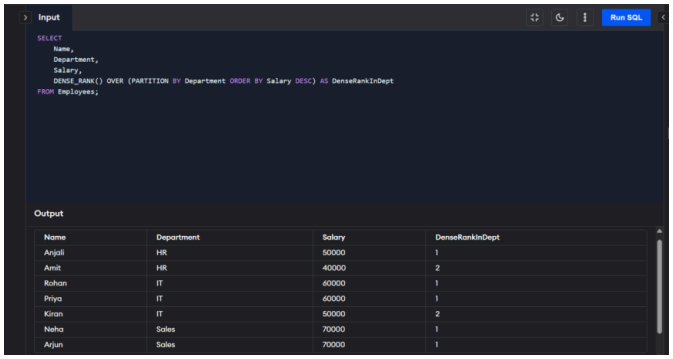
Input

```
CREATE TABLE Employees (
    EmpID INT,
    Name VARCHAR(100),
    Department VARCHAR(50),
    Salary INT
);
INSERT INTO Employees (EmpID, Name, Department, Salary)
VALUES
(1, 'Anjali', 'HR', 50000),
(2, 'Rohan', 'IT', 60000),
(3, 'Priya', 'IT', 60000),
(4, 'Amit', 'HR', 40000),
(5, 'Neha', 'Sales', 70000),
(6, 'Arjun', 'Sales', 70000),
(7, 'Kiran', 'IT', 50000);
```

Output

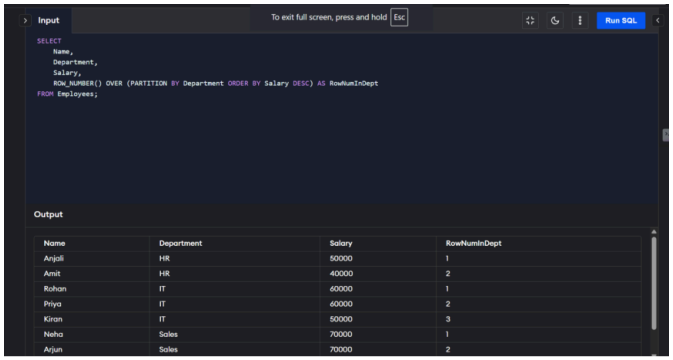
Name	Department	Salary	RankInDept
Anjali	HR	50000	1
Amit	HR	40000	2
Rohan	IT	60000	1
Priya	IT	60000	1
Kiran	IT	50000	3
Neha	Sales	70000	1
Arjun	Sales	70000	1

```
SELECT
    Name,
    Department,
    Salary,
    DENSE_RANK() OVER (PARTITION BY
Department ORDER BY Salary DESC) AS
DenseRankInDept
FROM Employees;
```



Name	Department	Salary	DenseRankInDept
Anjali	HR	50000	1
Amit	HR	40000	2
Rohan	IT	60000	1
Priya	IT	60000	1
Kiran	IT	50000	2
Neha	Sales	70000	1
Arjun	Sales	70000	1

```
SELECT
    Name,
    Department,
    Salary,
    ROW_NUMBER() OVER (PARTITION BY
Department ORDER BY Salary DESC) AS
RowNumInDept
FROM Employees;
```

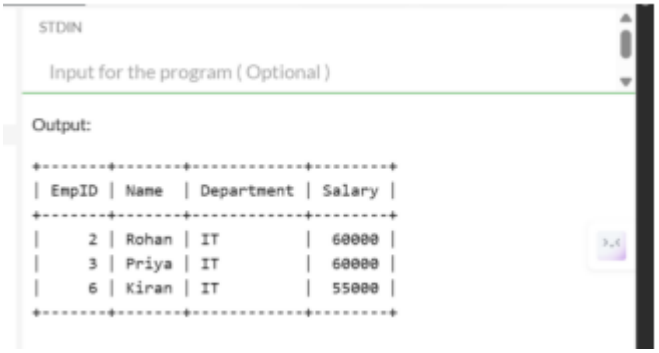


Name	Department	Salary	RowNumInDept
Anjali	HR	50000	1
Amit	HR	40000	2
Rohan	IT	60000	1
Priya	IT	60000	2
Kiran	IT	50000	3
Neha	Sales	70000	1
Arjun	Sales	70000	2

Exercise 2: Create a Stored Procedure

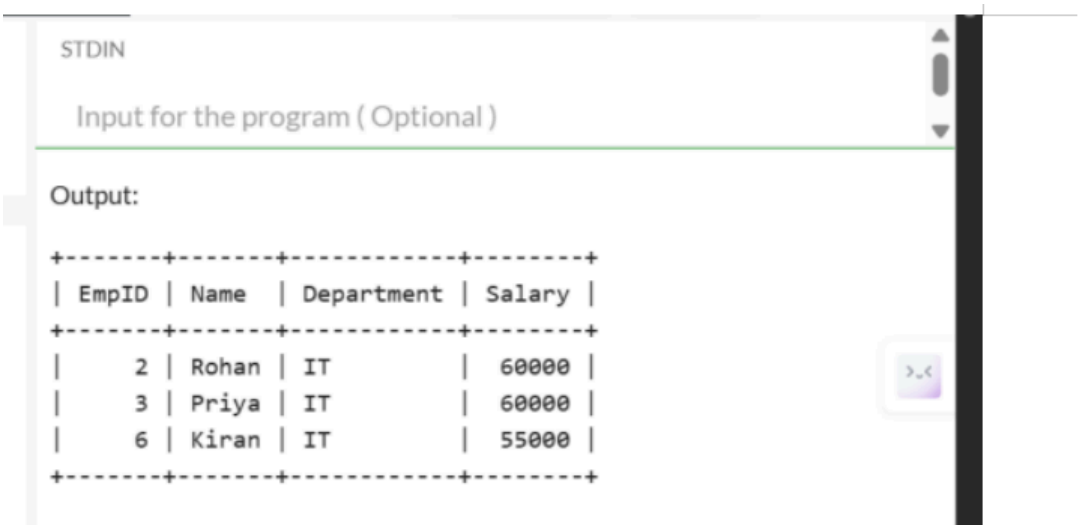
```
DELIMITER //
CREATE PROCEDURE InsertEmployee (
    IN emp_id INT,
    IN emp_name VARCHAR(100),
    IN dept VARCHAR(50),
    IN sal INT
)
BEGIN
    INSERT INTO Employees (EmpID, Name, Department, Salary)
    VALUES (emp_id, emp_name, dept, sal);
```

```
END //
DELIMITER ;

CALL InsertEmployee(6, 'Kiran', 'IT', 55000);
SELECT * FROM Employees;
```



Exercise 3: Return Data from a Stored Procedure

```
DELIMITER //
CREATE PROCEDURE GetEmployeesByDept (
   IN dept_name VARCHAR(50)
)
BEGIN
   SELECT * FROM Employees WHERE Department = dept_name;
END //
DELIMITER ;

CALL GetEmployeesByDept('IT');
```



Exercise 4: NUnit-Handson

Testing

```
using NUnit.Framework;

using CalcLibrary;

using System;


namespace CalcLibrary.Tests

{

    [TestFixture]

    public class CalculatorTests

    {

        private SimpleCalculator calc;


        [SetUp]

        public void SetUp()

        {

            calc = new SimpleCalculator();

        }


        [TearDown]

        public void TearDown()

        {

            calc.AllClear();

        }
```

```
[Test]
[TestCase(2, 3, 5)]
[TestCase(-1, -1, -2)]
[TestCase(0, 0, 0)]
public void TestAddition(double a, double b, double expected)
{
    var result = calc.Addition(a, b);
    Assert.That(result, Is.EqualTo(expected));
}

[Test]
[TestCase(5, 3, 2)]
[TestCase(-1, -2, 1)]
[TestCase(0, 0, 0)]
public void TestSubtraction(double a, double b, double expected)
{
    var result = calc.Subtraction(a, b);
    Assert.That(result, Is.EqualTo(expected));
}

[Test]
[TestCase(2, 3, 6)]
[TestCase(-1, -2, 2)]
[TestCase(0, 5, 0)]
public void TestMultiplication(double a, double b, double expected)
{
```

```
        var result = calc.Multiplication(a, b);
        Assert.That(result, Is.EqualTo(expected));
    }

    [Test]
    [TestCase(6, 3, 2)]
    [TestCase(5, 2, 2.5)]
    public void TestDivision(double a, double b, double expected)
    {
        var result = calc.Division(a, b);
        Assert.That(result, Is.EqualTo(expected));
    }

    [Test]
    public void TestDivisionByZero()
    {
        Assert.Throws<ArgumentException>(() => calc.Division(5, 0));
    }

    [Test]
    [Ignore("Testing ignore attribute")]
    public void IgnoredTest()
    {
        Assert.Fail("This test should be ignored.");
    }
}
```

}

OUTPUT:

Test Name	Status	Message
TestAddition (3 cases)	Passed	
TestSubtraction (3 cases)	Passed	
TestMultiplication (3)	Passed	
TestDivision (2 cases)	Passed	
TestDivisionByZero	Passed	Exception caught as expected

Exercise 5: Write Testable Code with Moq

```
public interface IMailSender
{
    bool SendMail(string toAddress, string message);
}

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;
        }
    }
}
namespace CustomerCommLib
{
    public class CustomerComm
    {
        IMailSender _mailSender;

        public CustomerComm(IMailSender mailSender)
        {
            _mailSender = mailSender;
        }

        public bool SendMailToCustomer()
        {
            _mailSender.SendMail("cust123@abc.com", "Some Message");
            return true;
        }
    }
}
```

```
using Moq;

using NUnit.Framework;

using CustomerCommLib;


namespace CustomerComm.Tests

{

    [TestFixture]

    public class CustomerCommTests

    {

        private Mock<IMailSender> _mailSenderMock;

        private CustomerComm _customerComm;


        [OneTimeSetUp]

        public void Init()

        {

            _mailSenderMock = new Mock<IMailSender>();

            _mailSenderMock.Setup(x => x.SendMail(It.IsAny<string>(),
It.IsAny<string>())).Returns(true);


            _customerComm = new CustomerComm(_mailSenderMock.Object);

        }


        [Test]

        public void SendMailToCustomer_ShouldReturnTrue()

        {

            var result = _customerComm.SendMailToCustomer();

            Assert.IsTrue(result);
```

```
        }
    }
}
```

OUTPUT:

Test Name: SendMailToCustomer_ShouldReturnTrue

Result: Passed