**Week-2**

**Nunit**

**Ex-1**

**Code:-**

using System;

using CalcLibrary;

using NUnit.Framework;

namespace code

{

    public class CalculatorTest

    {

        private SimpleCalculator calculator;

        [SetUp]

        public void Init()

        {

            calculator = new SimpleCalculator();

        }

        [TearDown]

        public void Cleanup()

        {

            calculator.AllClear();

        }

        [TestCase(10,5,15)]

        [TestCase(-2, 3, 1)]

        [TestCase(0, 0, 0)]

        public void TestAddition(double a, double b, double expected)

        {

            double actual = calculator.Addition(a, b);

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

        }

        [TestCase(15,5,10)]

        [TestCase(10,2,8)]

        [TestCase(8,8,0)]

        public void TestSubtraction(double a, double b, double expected)

        {

            double result = calculator.Subtraction(a, b);

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

        }

        [TestCase(2, 2, 4)]

        [TestCase(10, 0, 0)]

        [TestCase(8, 8, 64)]

        public void TestMultiplication(double a, double b, double expected)

        {

            double result = calculator.Multiplication(a, b);

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

        }

        //[TestCase(2, 2, 1)]

        //[TestCase(10, 5, 2)]

        //[TestCase(8, 4, 2)]

        //public void TestDivision(double a, double b, double expected)

        //{

        //    double result = calculator.Division(a, b);

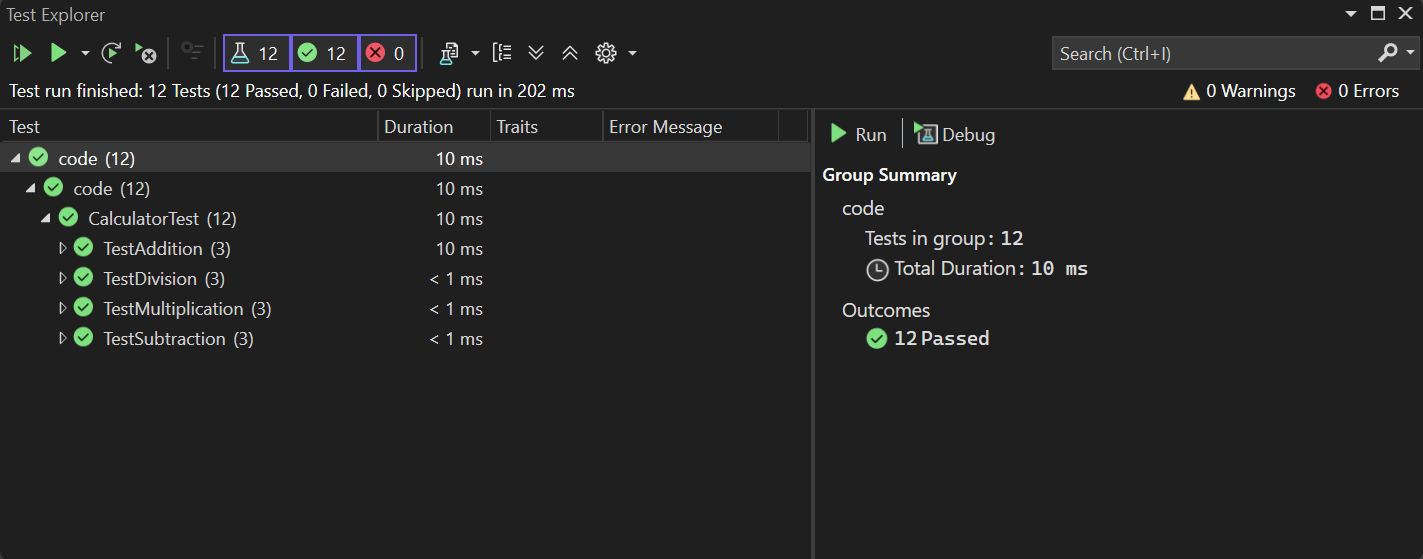
        //    Assert.That(result, Is.EqualTo(expected));

        //}

    }

}

Output:-



**EX-2**

**Code:-**

using System;

using CalcLibrary;

namespace code

{

    [TestFixture]

    public class CalculatorTest

    {

        public SimpleCalculator Calculator { get; set; }

        [SetUp]

        public void Init()

        {

            Calculator = new SimpleCalculator();

        }

        [TearDown]

        public void TearDown()

        {

            Calculator.AllClear();

        }

        [TestCase(5,5,0)]

        [TestCase(5,4,1)]

        [TestCase(9,5,4)]

        public void TestSubtraction(double a,double b,double expected)

        {

            double result = Calculator.Subtraction(a,b);

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

        }

        [TestCase(5, 5, 25)]

        [TestCase(5, 4, 20)]

        [TestCase(9, 4, 36)]

        public void TestMultiplication(double a, double b, double expected)

        {

            double result = Calculator.Multiplication(a,b);

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

        }

        [TestCase(5, 5, 1)]

        [TestCase(5, 2, 2.5)]

        [TestCase(9, 0, 3)]

        public void TestDivision(double a, double b, double expected)

        {

            try

            {

                double result = Calculator.Division(a, b);

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

            }

            catch(ArgumentException ex)

            {

                Assert.That(ex.Message, Is.EqualTo("Second Parameter Can't be Zero"));

                Assert.Fail("Division By Zero");

            }

        }

            [Test]

            public void TestAddAndClear()

            {

                Calculator.Addition(5, 5);

                Assert.That(10, Is.EqualTo(Calculator.GetResult));

                Calculator.AllClear();

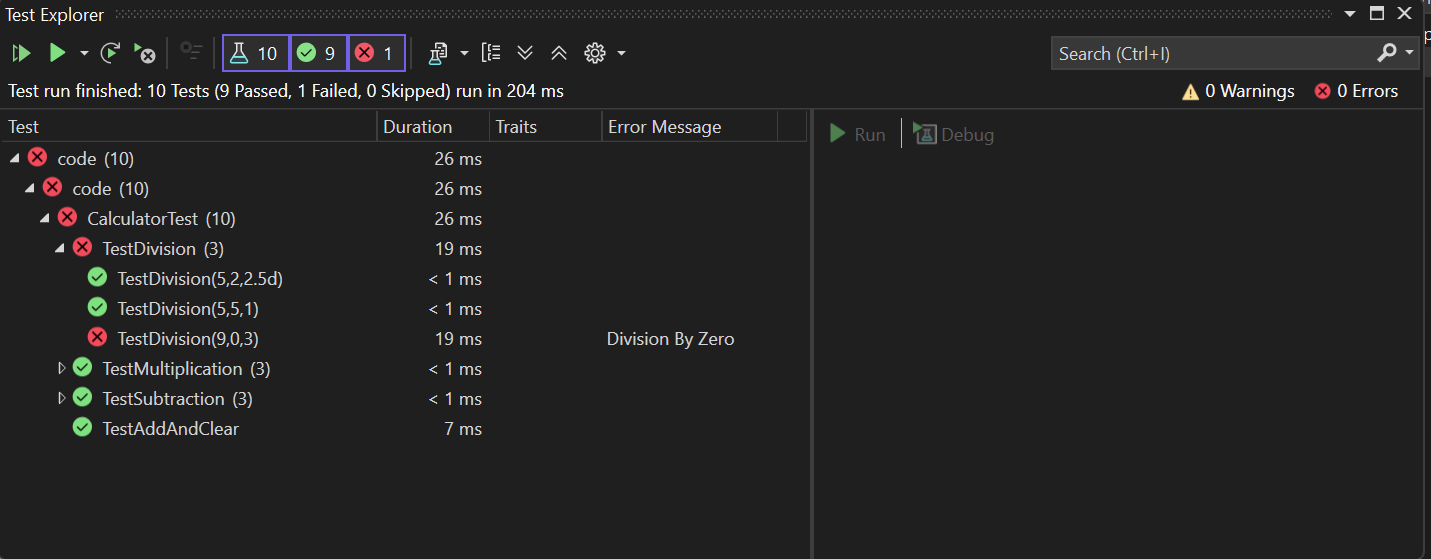
                Assert.That(0, Is.EqualTo(Calculator.GetResult));

            }

    }

}

**Output:-**

****

**Ex-3**

**Code:-**

**using NUnit.Framework;**

**using UtilLib;**

**namespace url.Tests**

**{**

**[TestFixture]**

**public class UrlHostNameParserTests**

**{**

**private UrlHostNameParser parser;**

**[SetUp]**

**public void Setup()**

**{**

**parser = new UrlHostNameParser();**

**}**

**[Test]**

**public void ParseHostName\_HttpUrl\_ReturnsHostName()**

**{**

**string url = "http://example.com/page";**

**string result = parser.ParseHostName(url);**

**Assert.That(result, Is.EqualTo("example.com"));**

**}**

**[Test]**

**public void ParseHostName\_HttpsUrl\_ReturnsHostName()**

**{**

**string url = "https://openai.com/index.html";**

**string result = parser.ParseHostName(url);**

**Assert.That(result, Is.EqualTo("openai.com"));**

**}**

**[Test]**

**public void ParseHostName\_InvalidProtocol\_ThrowsFormatException()**

**{**

**string url = "ftp://fileserver.com/downloads";**

**try**

**{**

**parser.ParseHostName(url);**

**Assert.Fail("Expected FormatException was not thrown.");**

**}**

**catch (FormatException ex)**

**{**

**// Check that the exception message is correct**

**Assert.That(ex.Message, Is.EqualTo("Url is not in correct format"));**

**}**

**}**

**[Test]**

**public void ParseHostName\_EmptyString\_ThrowsIndexOutOfRange()**

**{**

**string url = "";**

**try**

**{**

**parser.ParseHostName(url);**

**Assert.Fail("Expected IndexOutOfRangeException was not thrown.");**

**}**

**catch (IndexOutOfRangeException ex)**

**{**

**Assert.Pass("Caught expected IndexOutOfRangeException");**

**}**

**}**

**[Test]**

**public void ParseHostName\_MissingProtocol\_ThrowsException()**

**{**

**string url = "openai.com/index.html";**

**try**

**{**

**parser.ParseHostName(url);**

**Assert.Fail("Expected IndexOutOfRangeException was not thrown.");**

**}**

**catch (IndexOutOfRangeException ex)**

**{**

**Assert.Pass("Caught expected IndexOutOfRangeException");**

**}**

**}**

**[Test]**

**public void ParseHostName\_NullInput\_ThrowsArgumentNullException()**

**{**

**string url = null;**

**try**

**{**

**parser.ParseHostName(url);**

**Assert.Fail("Expected ArgumentNullException was not thrown.");**

**}**

**catch (ArgumentNullException ex)**

**{**

**Assert.Pass("Caught expected ArgumentNullException");**

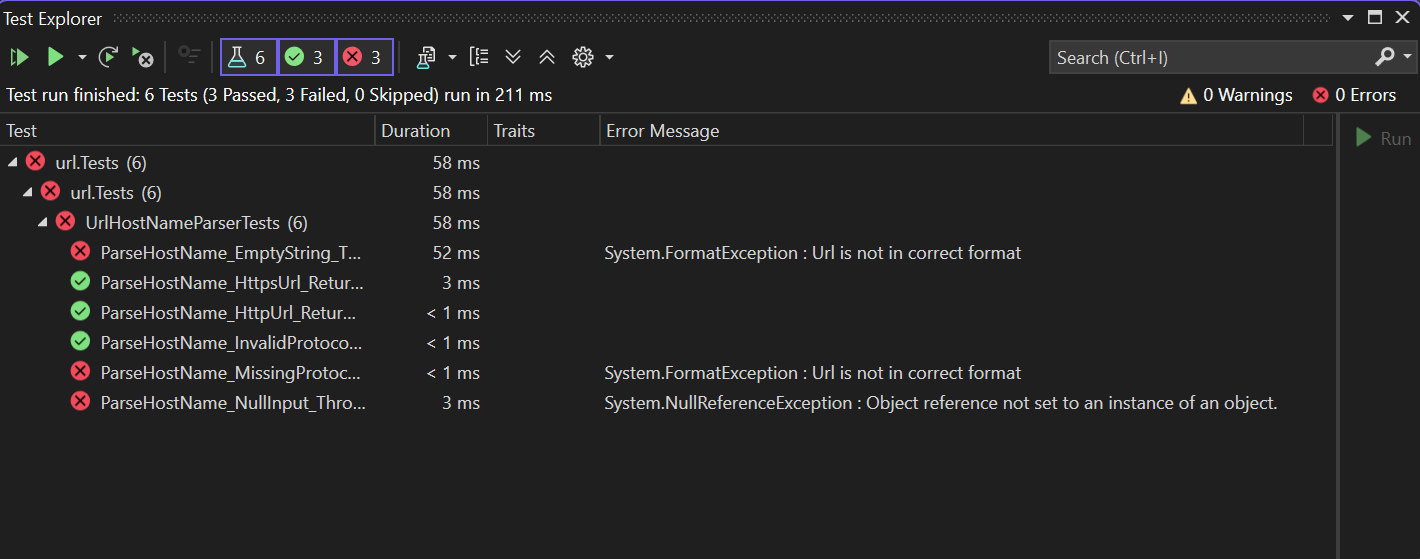
**}**

**}**

**}**

**}**

**Output:-**

****

**Ex4:-**

**Code:**

using System;

using AccountsManagerLib;

namespace code

{

    public class AccountManagerTests

    {

        private AccountsManager Manager { get; set; }

        [SetUp]

        public void Setup()

        {

            Manager = new AccountsManager();

        }

        [Test]

        public void ValidateUser\_ValidCredentials\_RetunsWelcomeMessage()

        {

            string output = Manager.ValidateUser("user\_11", "secret@user11");

            Assert.That("Welcome user\_11!!!",Is.EqualTo(output));

        }

        [TestCase("user\_11", "secret@user11", "Welcome user\_11!!!")]

        [TestCase("user\_22", "secret@user22", "Welcome user\_22!!!")]

        public void ValidateUser\_ValidateAllUsers\_ReturnsCorrectWelcome(string username,string password,string expected)

        {

            string output = Manager.ValidateUser(username, password);

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

        }

        [Test]

        public void ValidateUser\_EmptyInput\_ReturnsFormatException()

        {

            try

            {

                string output = Manager.ValidateUser("", "secret@user11");

                Assert.Fail("Empty String Passed");

                Assert.That(output, Is.EqualTo("Welcome user\_11!!!"));

            }

            catch (FormatException ex)

            {

                Assert.That("Both user id and password are mandatory",Is.EqualTo(ex.Message));

                //Assert.Fail("Empty String Passed");

            }

        }

        [Test]

        public void ValidateUser\_InvalidCredentials\_ReturnsInvalidMessage()

        {

            string output = Manager.ValidateUser("user33", "secret@user11");

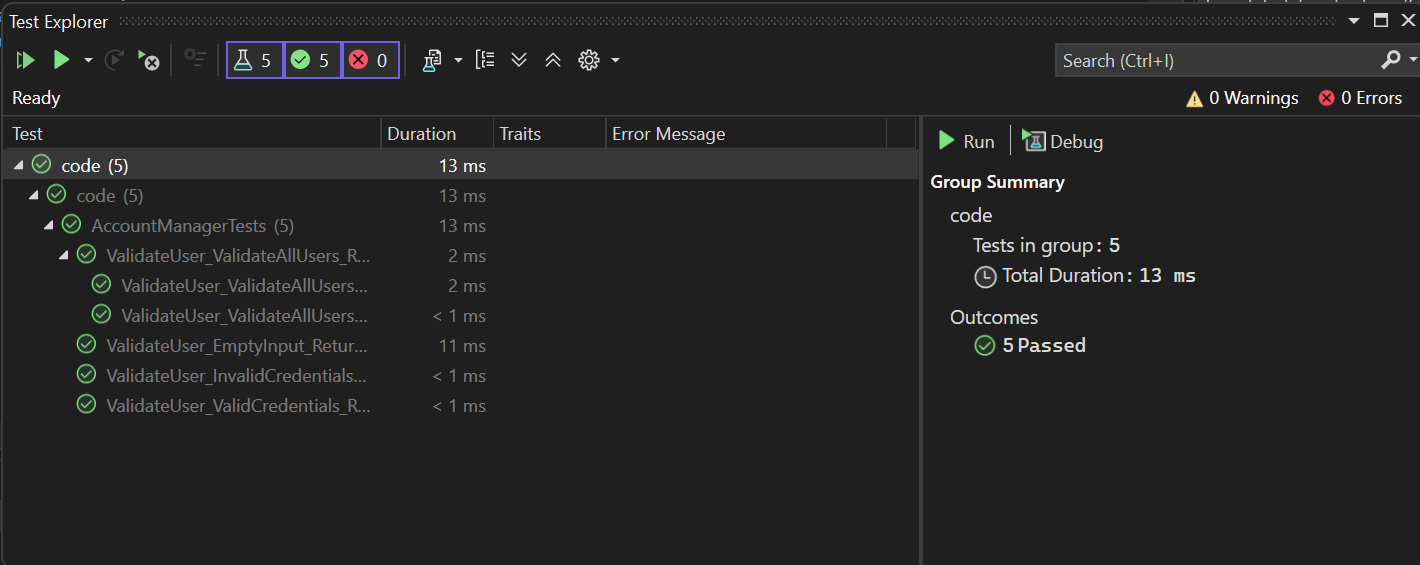
            Assert.That("Invalid user id/password", Is.EqualTo(output));

        }

    }

}

**Output:-**

****

**Moq:-**

**Ex1**

**Code:-**

using code;

using Moq;

using NUnit.Framework;

namespace CustomerCommTests

{

    [TestFixture]

    public class CustomerCommTests

    {

        private Mock<IMailSender> \_mockMailSender;

        private CustomerComm \_customerComm;

        [OneTimeSetUp]

        public void Setup()

        {

            // Arrange: mock IMailSender to always return true

            \_mockMailSender = new Mock<IMailSender>();

            \_mockMailSender

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

                .Returns(true);

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

        }

        [TestCase]

        public void SendMailToCustomer\_ReturnsTrue\_WhenMailIsSent()

        {

            // Act

            bool result = \_customerComm.SendMailToCustomer();

            // Assert

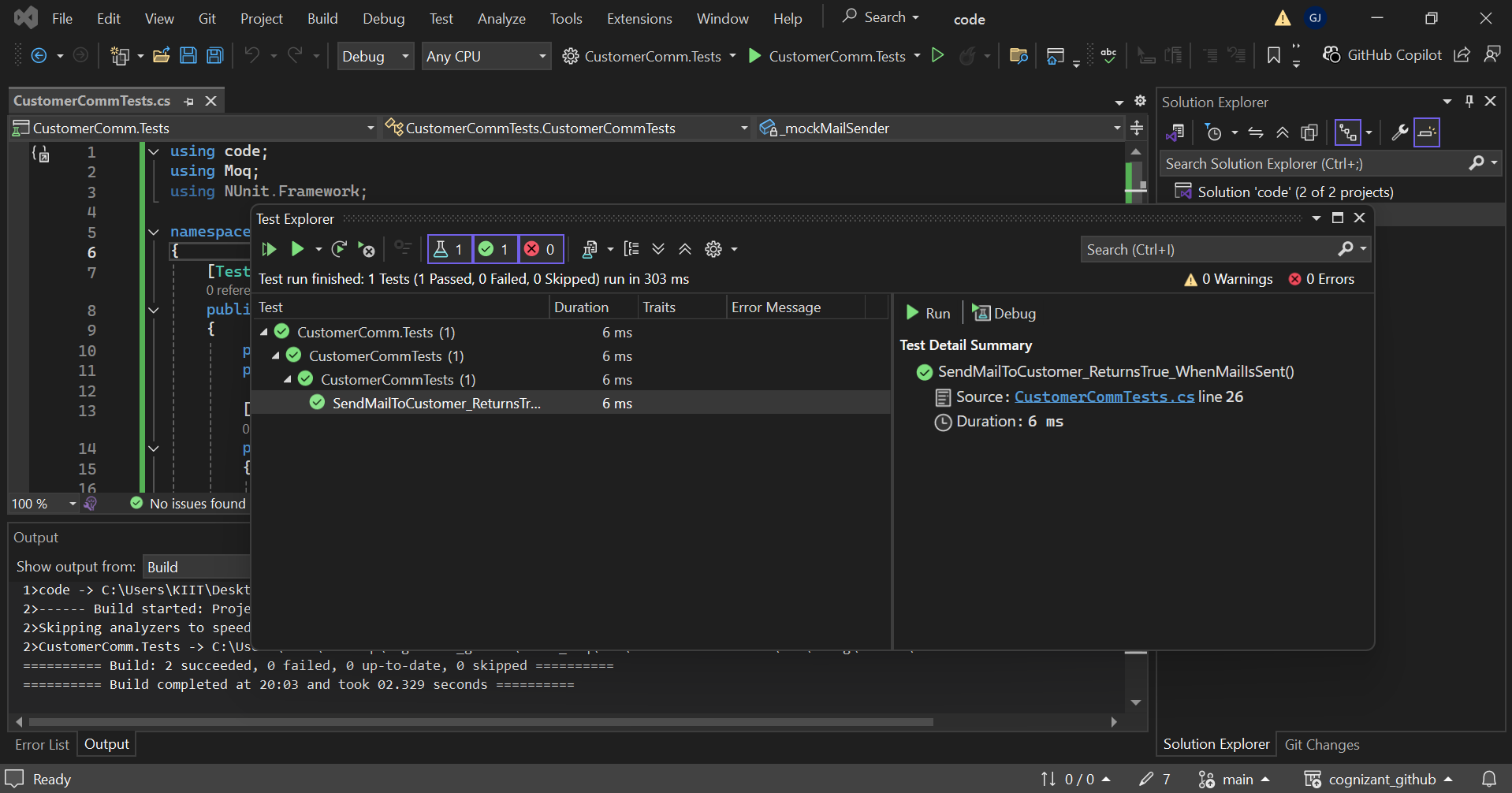
            Assert.IsTrue(result);

        }

    }

}

**Output:-**



**SQL:-**

**Ex1 Advance Sql**

**EX1.1**

**Code:-**

SELECT \* FROM(

    SELECT

    p.ProductID,

    p.ProductName,

    p.CategoryID,

    c.CategoryName,

    p.Price,

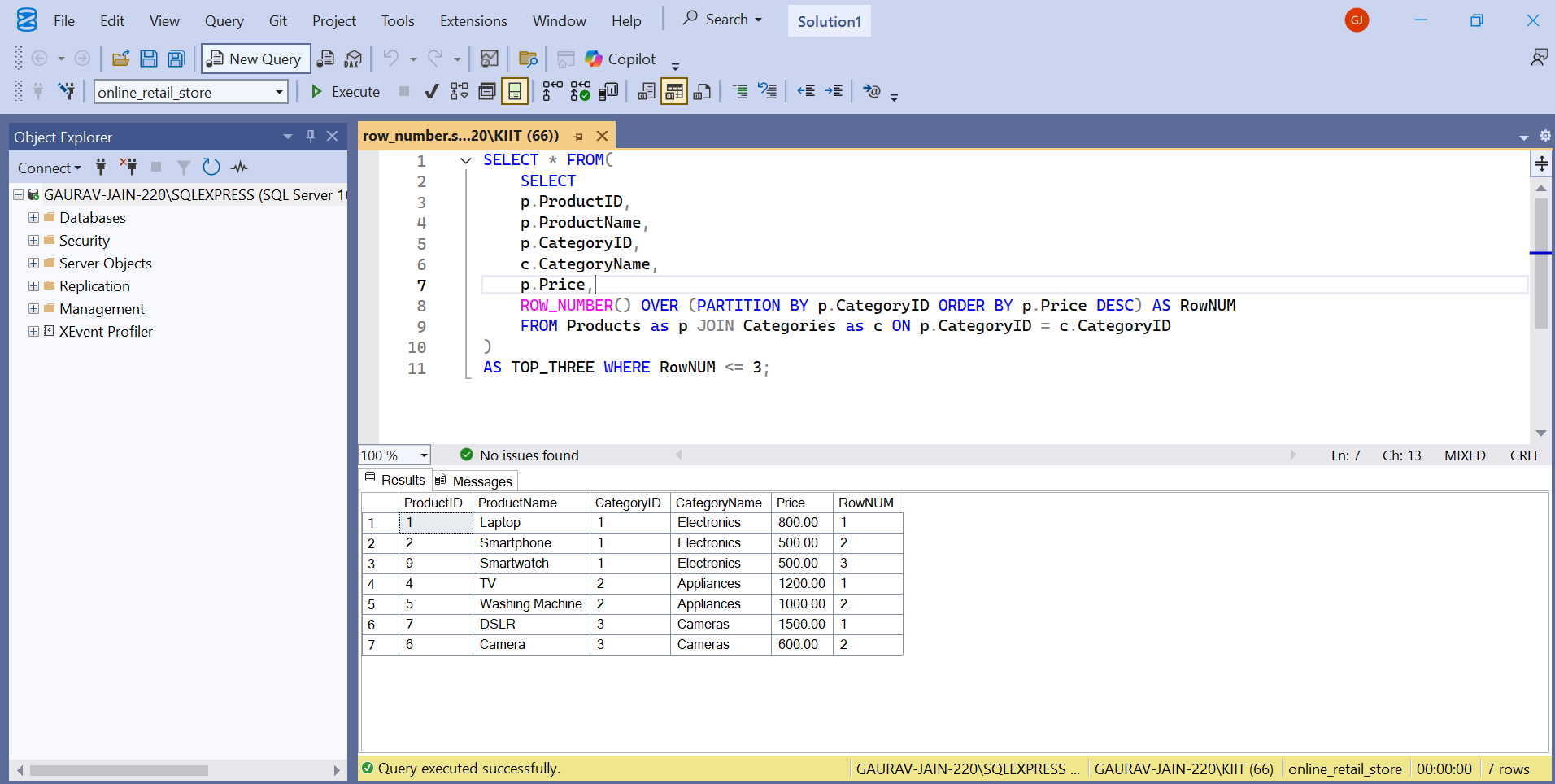
    ROW\_NUMBER() OVER (PARTITION BY p.CategoryID ORDER BY p.Price DESC) AS RowNUM

    FROM Products as p JOIN Categories as c ON p.CategoryID = c.CategoryID

)

AS TOP\_THREE WHERE RowNUM <= 3;

**Output:-**



**Code:-**

SELECT \* FROM(

    SELECT

    p.ProductID,

    p.ProductName,

    p.CategoryID,

    c.CategoryName,

    p.Price,

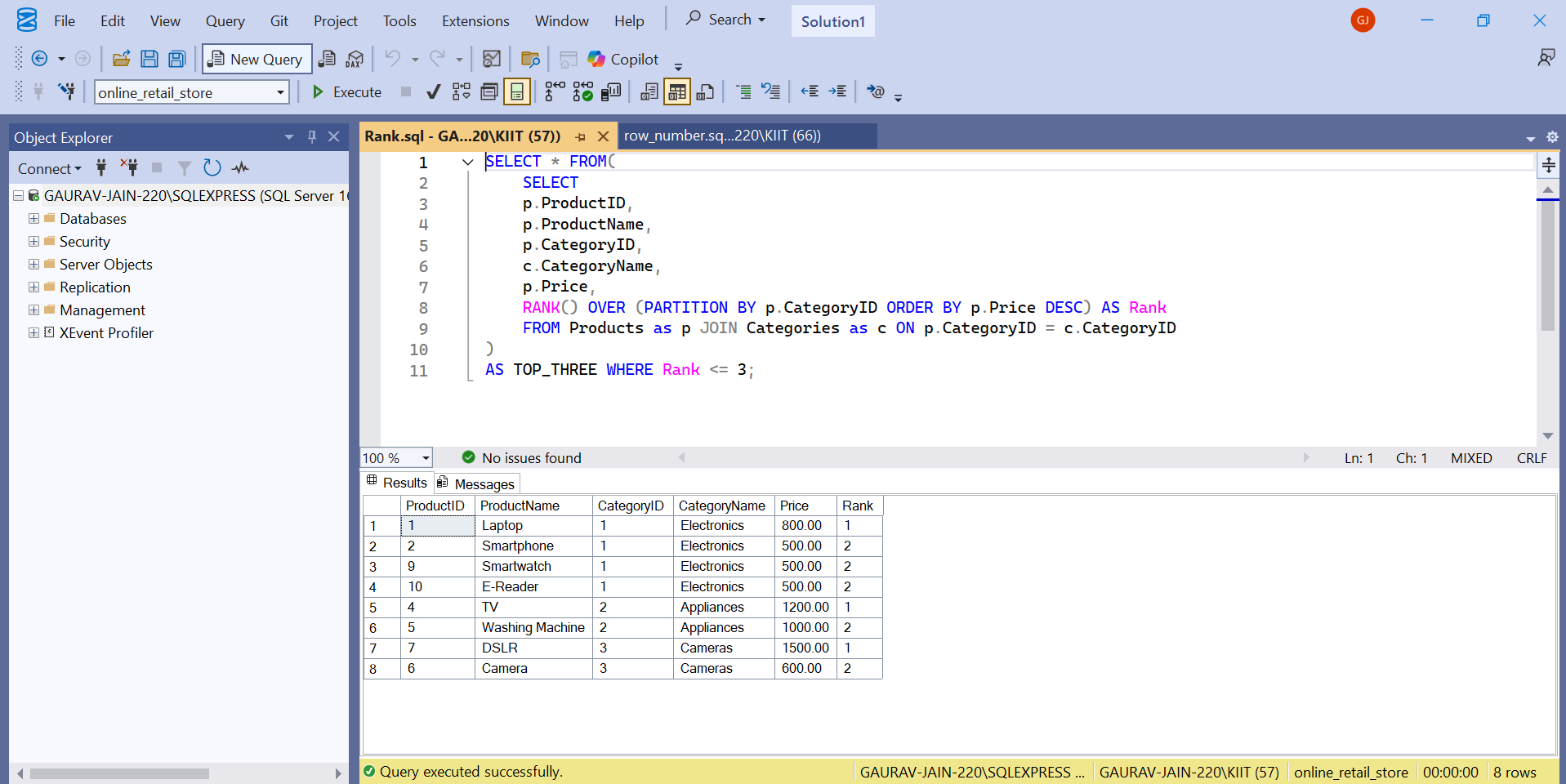
    RANK() OVER (PARTITION BY p.CategoryID ORDER BY p.Price DESC) AS Rank

    FROM Products as p JOIN Categories as c ON p.CategoryID = c.CategoryID

)

AS TOP\_THREE WHERE Rank <= 3;

**Output:-**



**Code:-**

SELECT \* FROM(

    SELECT

    p.ProductID,

    p.ProductName,

    p.CategoryID,

    c.CategoryName,

    p.Price,

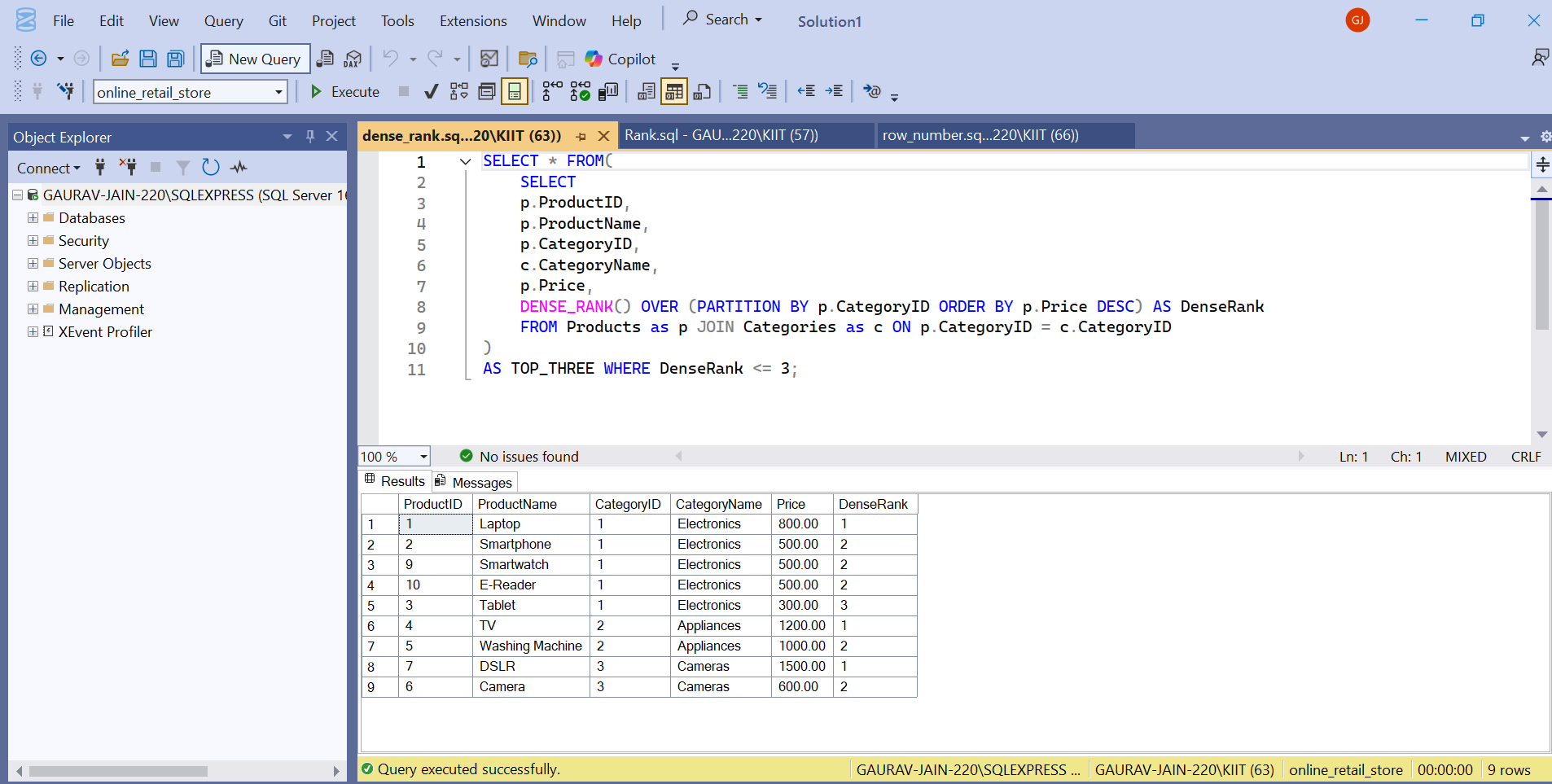
    DENSE\_RANK() OVER (PARTITION BY p.CategoryID ORDER BY p.Price DESC) AS DenseRank

    FROM Products as p JOIN Categories as c ON p.CategoryID = c.CategoryID

)

AS TOP\_THREE WHERE DenseRank <= 3;

**Output:**



**Ex-1.2**

**Code:-**

SELECT c.Region, cat.CategoryName,SUM(od.Quantity) AS Total\_Quantity from OrderDetails od

JOIN Orders o ON od.OrderID = o.OrderID

JOIN Customers  c ON c.CustomerID = o.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

JOIN Categories cat ON p.CategoryID = cat.CategoryID

GROUP BY GROUPING SETS(

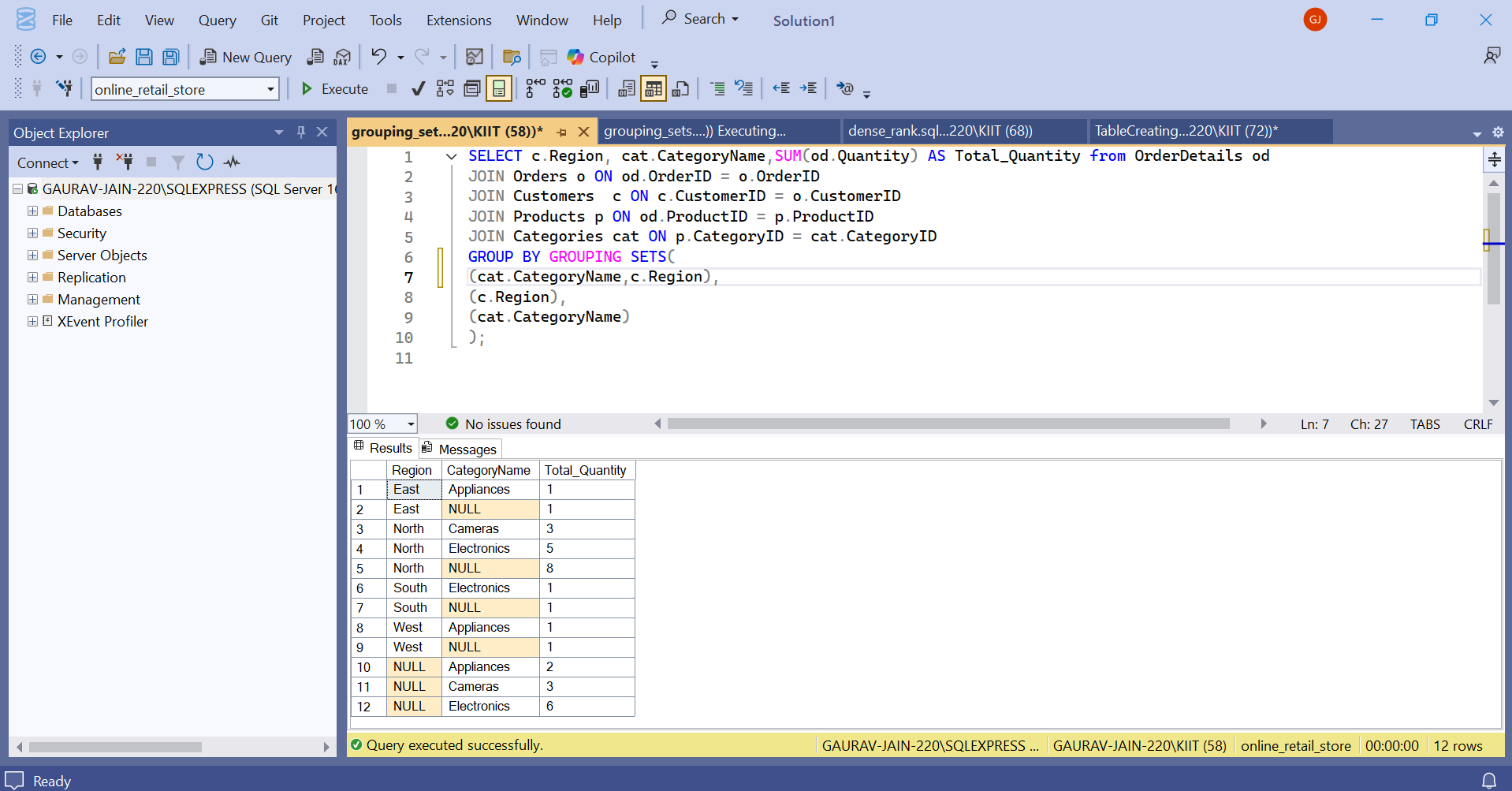
(cat.CategoryName,c.Region),

(c.Region),

(cat.CategoryName)

);

**Output:**



Code:-

SELECT c.Region, cat.CategoryName,SUM(od.Quantity) AS Total\_Quantity from OrderDetails od

JOIN Orders o ON od.OrderID = o.OrderID

JOIN Customers  c ON c.CustomerID = o.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

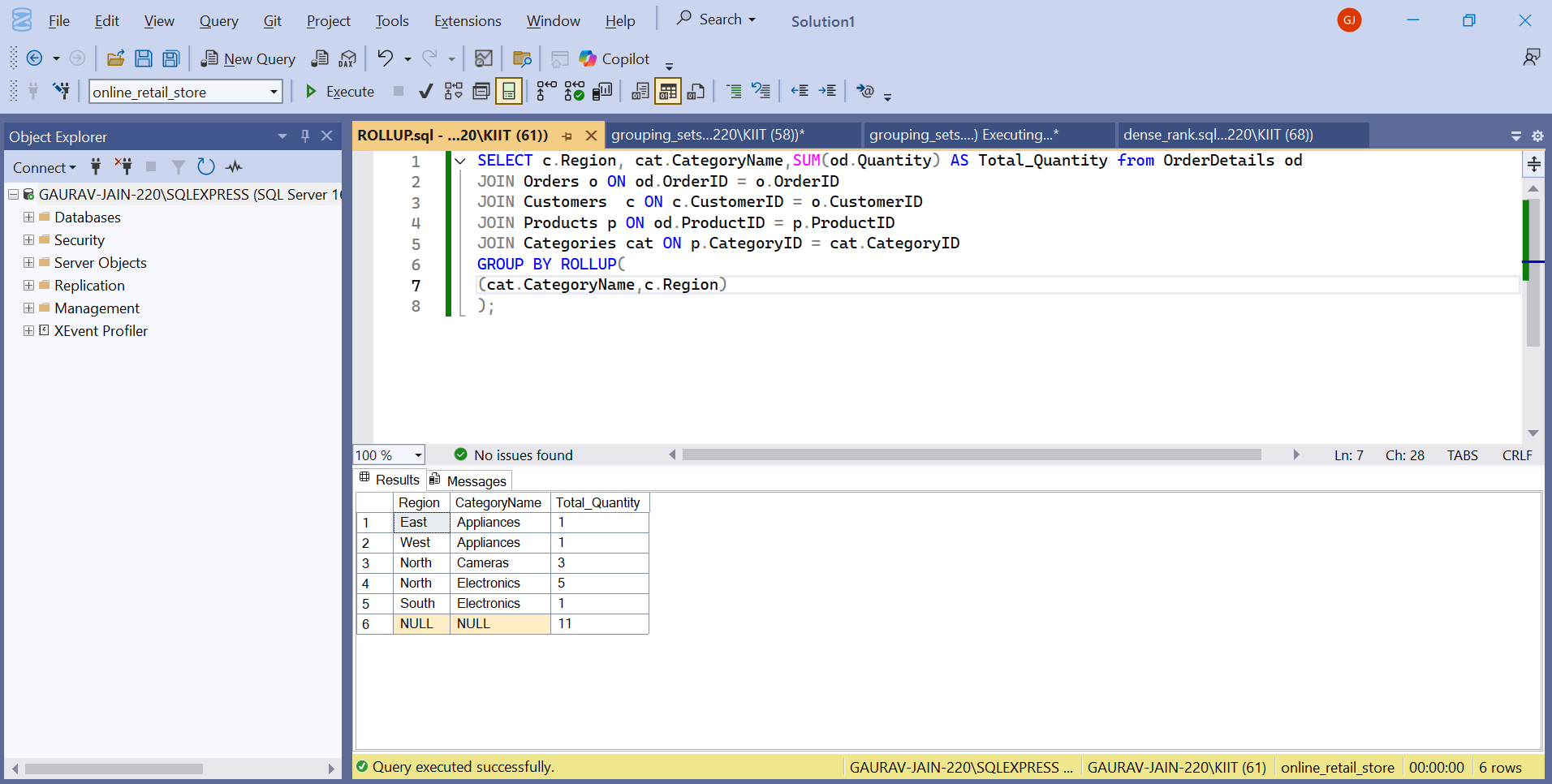
JOIN Categories cat ON p.CategoryID = cat.CategoryID

GROUP BY ROLLUP(

(cat.CategoryName,c.Region)

);

**Output:**



**Code:**

SELECT c.Region, cat.CategoryName,SUM(od.Quantity) AS Total\_Quantity from OrderDetails od

JOIN Orders o ON od.OrderID = o.OrderID

JOIN Customers  c ON c.CustomerID = o.CustomerID

JOIN Products p ON od.ProductID = p.ProductID

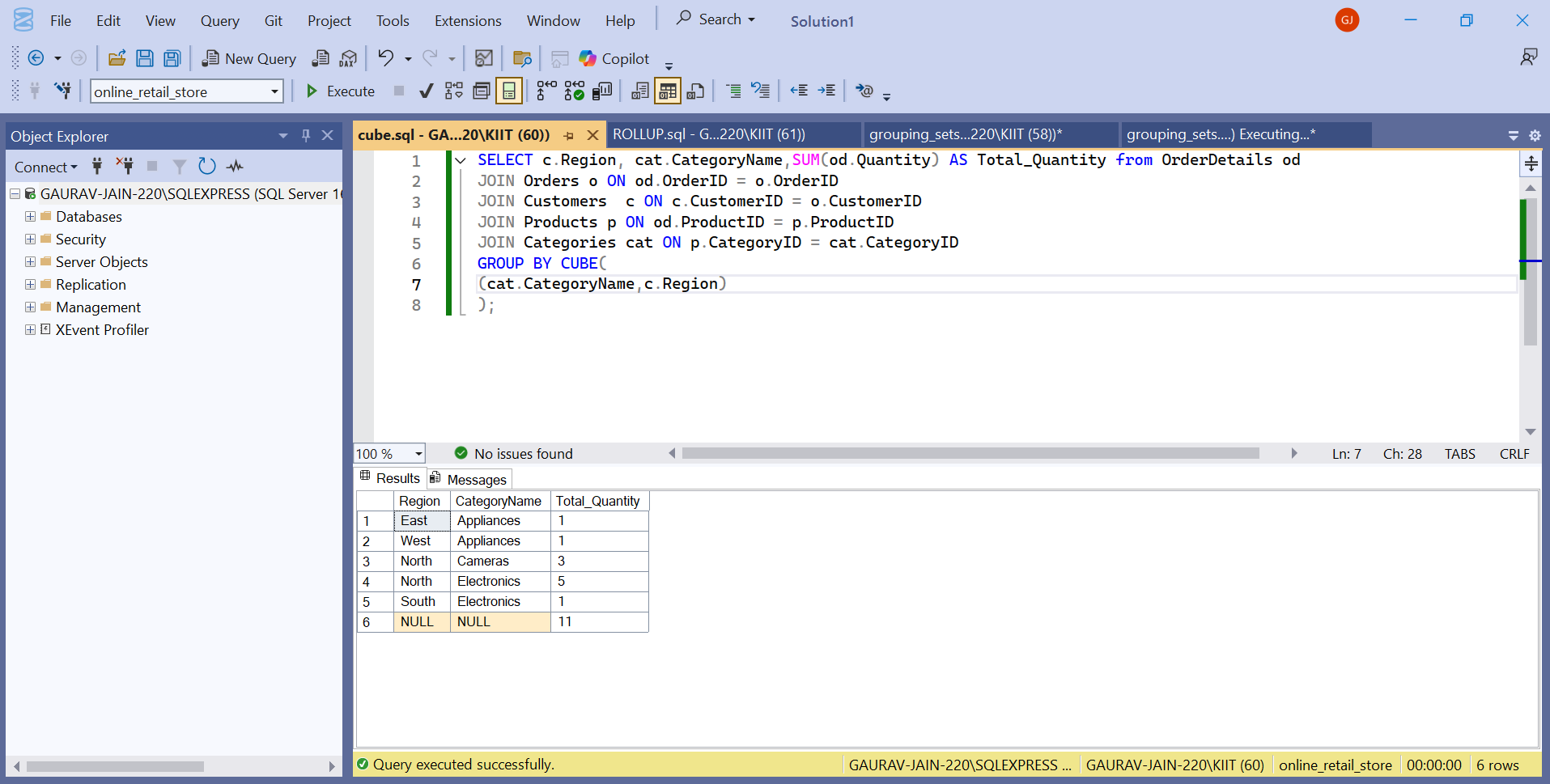
JOIN Categories cat ON p.CategoryID = cat.CategoryID

GROUP BY CUBE(

(cat.CategoryName,c.Region)

);

**Output:**



**Ex-2 Index**

-- Exercise 1: Creating a Non-Clustered Index

-- Goal: Create a non-clustered index on the ProductName column in the Products table and compare query execution time before and after index creation.

-- Step 1: Query to fetch product details before index creation

SELECT \* FROM Products WHERE ProductName = 'Laptop';

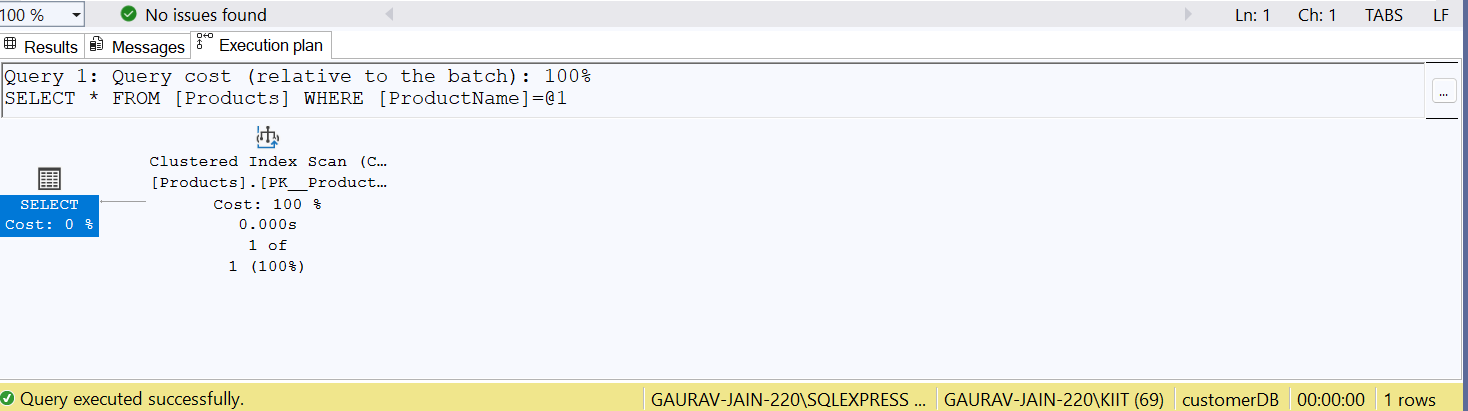
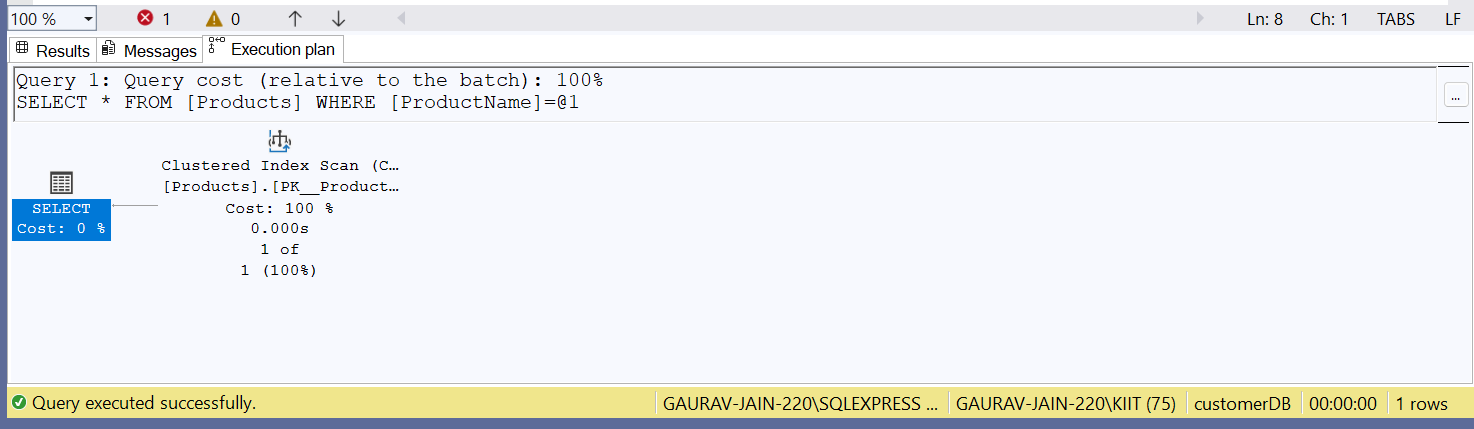
-- Step 2: Create a non-clustered index on ProductName

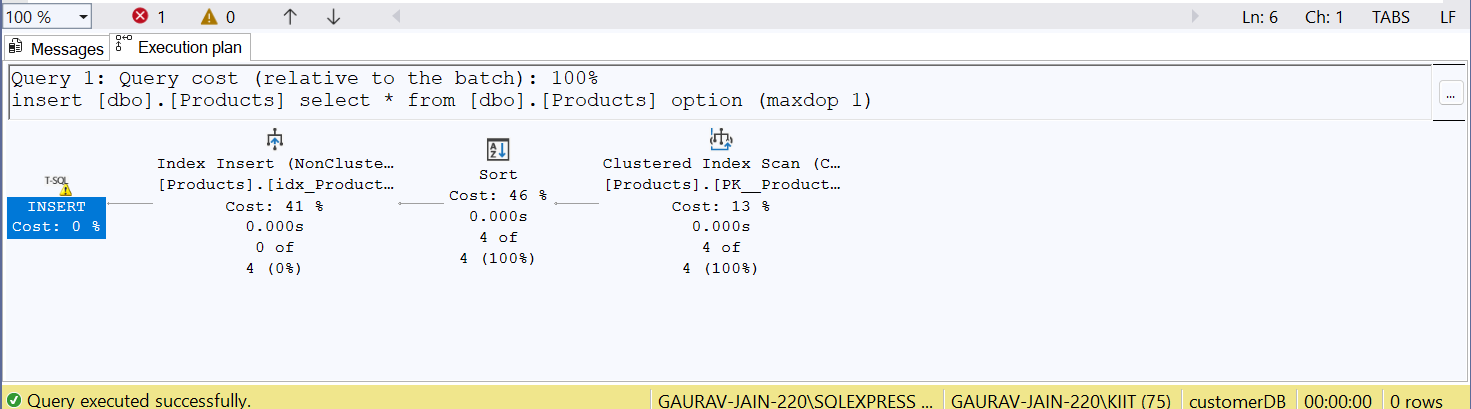
CREATE NONCLUSTERED INDEX idx\_ProductName ON Products(ProductName);

-- Step 3: Query to fetch product details after index creation

SELECT \* FROM Products WHERE ProductName = 'Laptop';

**Output:**





-- Exercise 2: Creating a Clustered Index

-- Goal: Create a clustered index on the OrderDate column in the Orders table and compare query execution time before and after index creation.

-- Step 1: Query to fetch orders before index creation

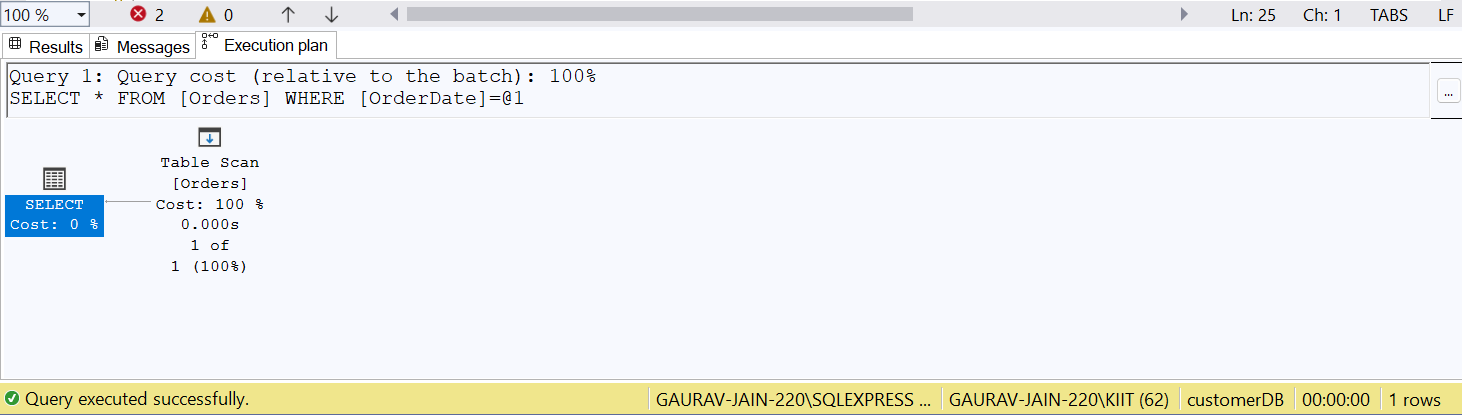
SELECT \* FROM Orders WHERE OrderDate = '2023-01-15';

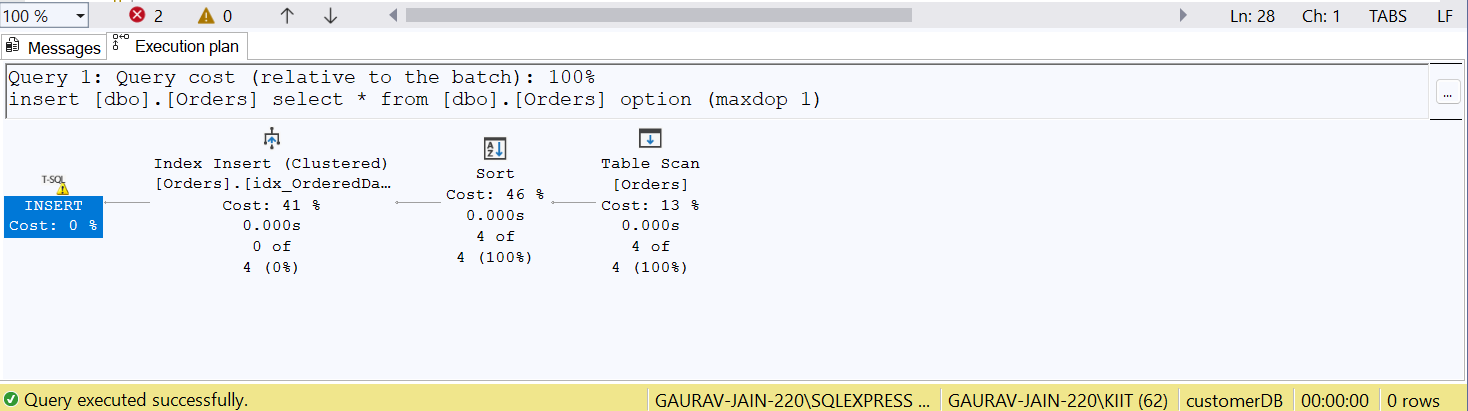
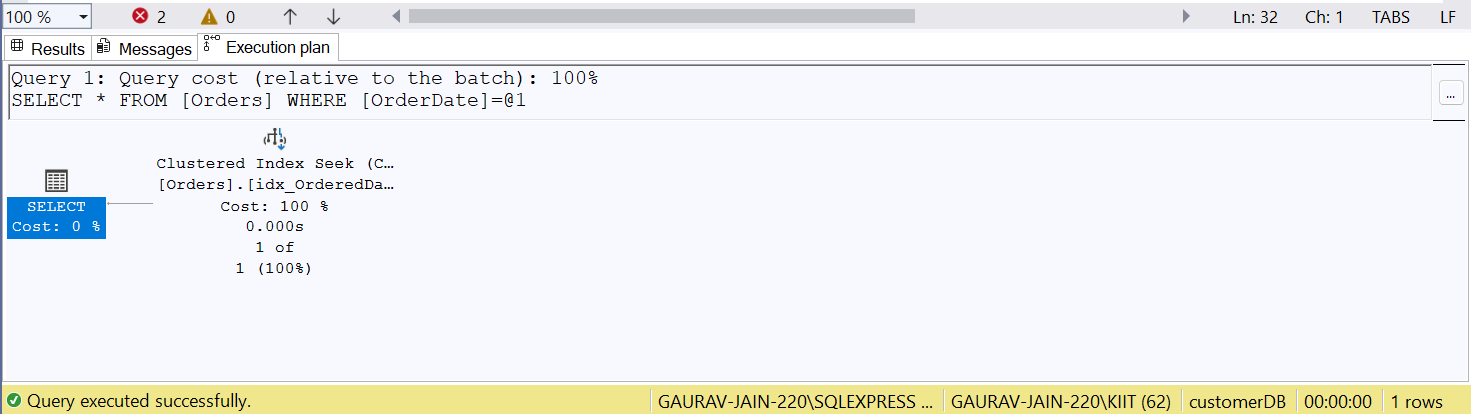
-- Step 2: Create a clustered index on OrderDate

CREATE CLUSTERED INDEX idx\_OrderedDate ON Orders(OrderDate);

-- Step 3: Query to fetch orders after index creation

SELECT \* FROM Orders WHERE OrderDate = '2023-01-15';

**Output:-** 



-- Exercise 3: Creating a Composite Index

-- Goal: Create a composite index on the CustomerID and OrderDate columns in the Orders table and compare query execution time before and after index creation.

-- Step 1: Query to fetch orders before index creation

SELECT \* FROM Orders WHERE CustomerID = 1 AND OrderDate = '2023-01-15';

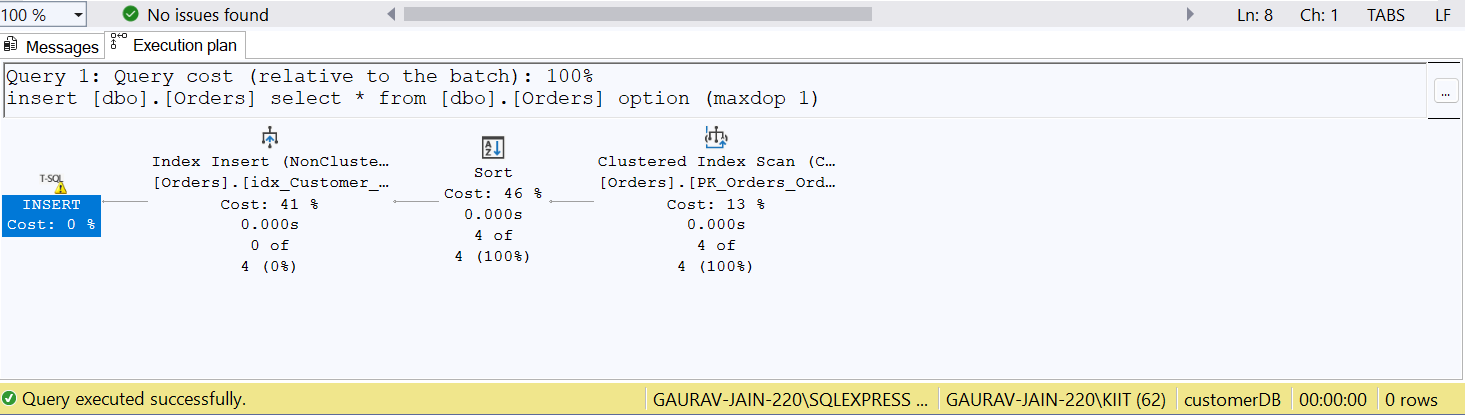
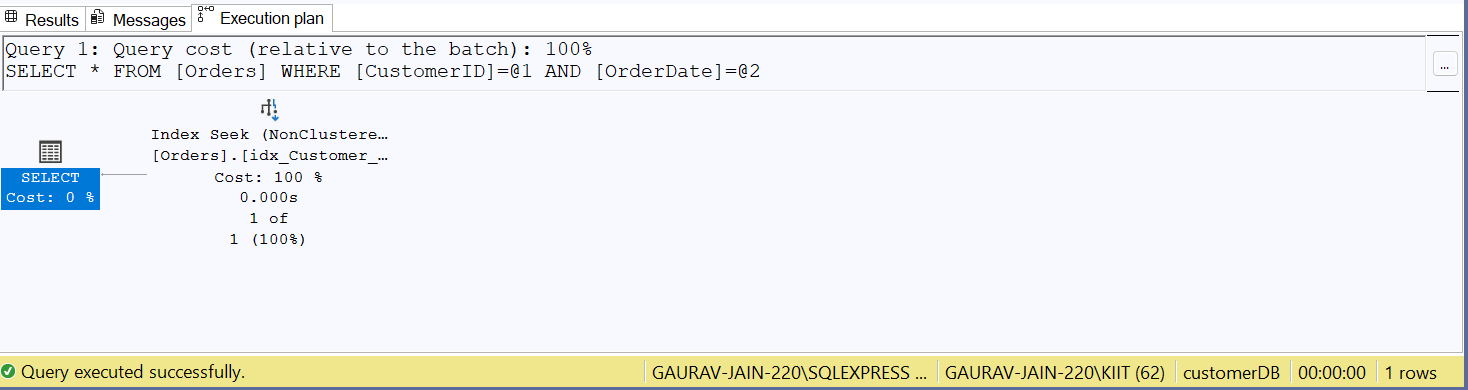
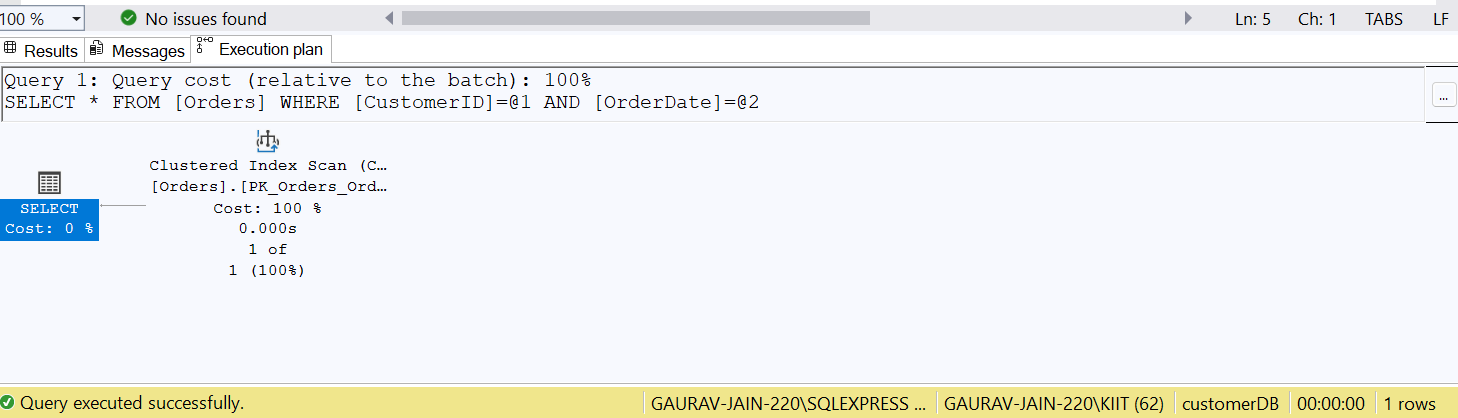
-- Step 2: Create a composite index on CustomerID and OrderDate

CREATE NONCLUSTERED INDEX idx\_Customer\_OrderDate ON Orders (CustomerID, OrderDate);

-- Step 3: Query to fetch orders after index creation

SELECT \* FROM Orders WHERE CustomerID = 1 AND OrderDate = '2023-01-15';

**Output**



**Ex4 Stored Procedures**

**Ex4.1**

**Code:**

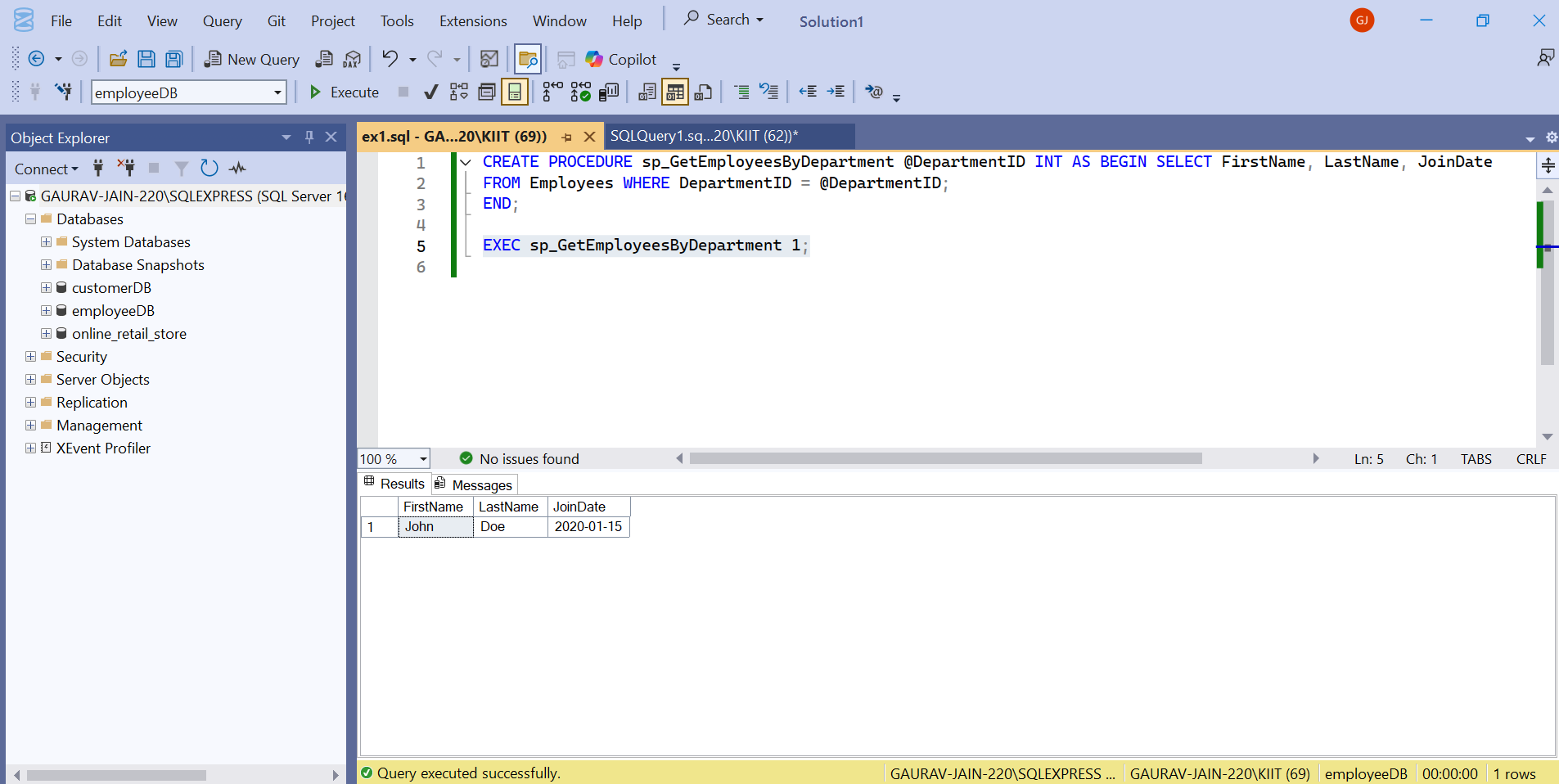
CREATE PROCEDURE sp\_GetEmployeesByDepartment @DepartmentID INT AS BEGIN SELECT FirstName, LastName, JoinDate

FROM Employees WHERE DepartmentID = @DepartmentID;

END;

EXEC sp\_GetEmployeesByDepartment 1;

**Output:**



**Ex4.2**

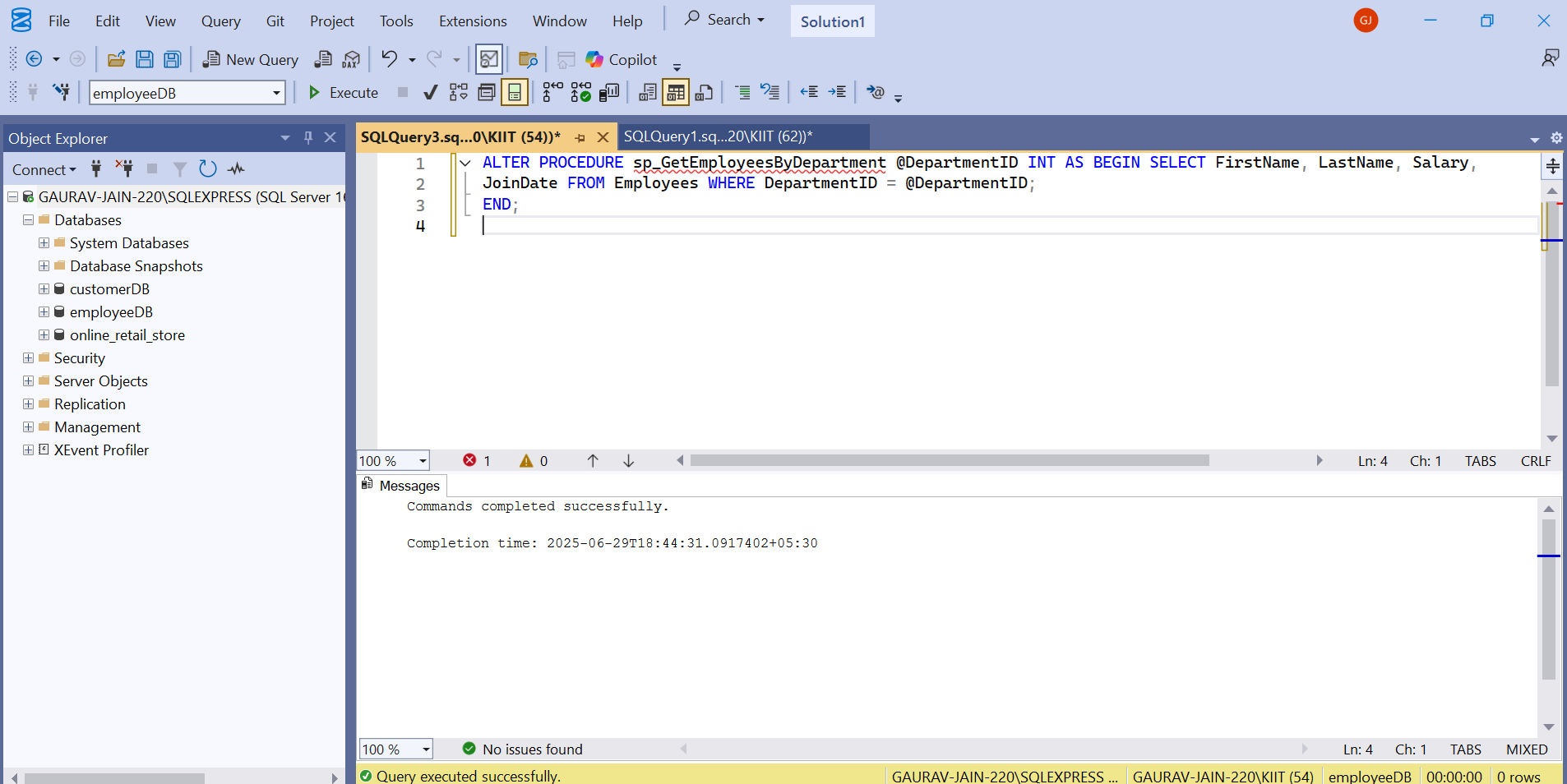
**Code**

ALTER PROCEDURE sp\_GetEmployeesByDepartment @DepartmentID INT AS BEGIN SELECT FirstName, LastName, Salary,

JoinDate FROM Employees WHERE DepartmentID = @DepartmentID;

END;

**Output:**

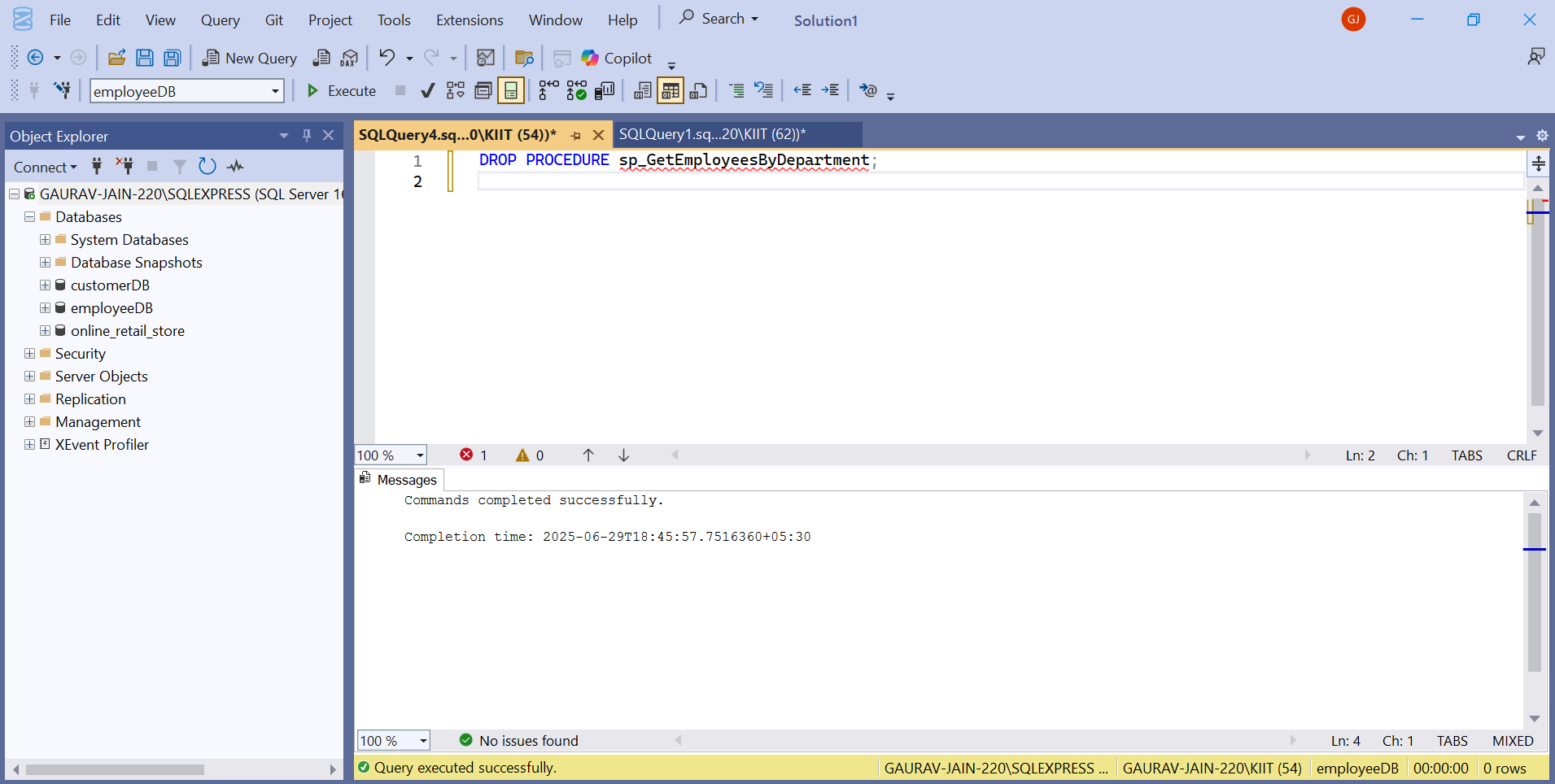


**Ex 4.3**

**Code:**

DROP PROCEDURE sp\_GetEmployeesByDepartment;

**Output:**



**Ex4.4**

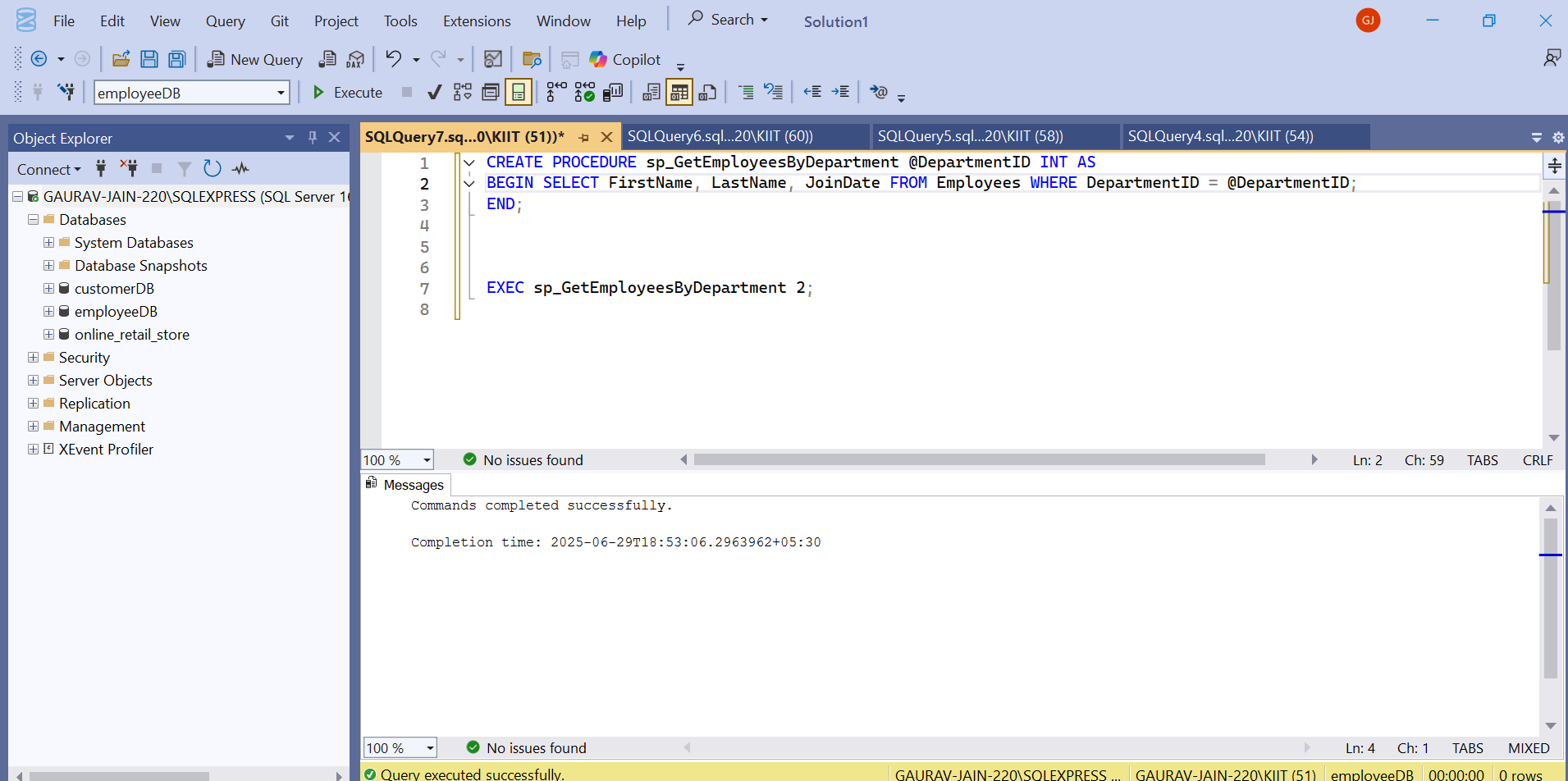
**Code:**

CREATE PROCEDURE sp\_GetEmployeesByDepartment @DepartmentID INT AS

BEGIN SELECT FirstName, LastName, JoinDate FROM Employees WHERE DepartmentID = @DepartmentID;

END;

EXEC sp\_GetEmployeesByDepartment 2;

**Output:****Ex4.5**

**Code**

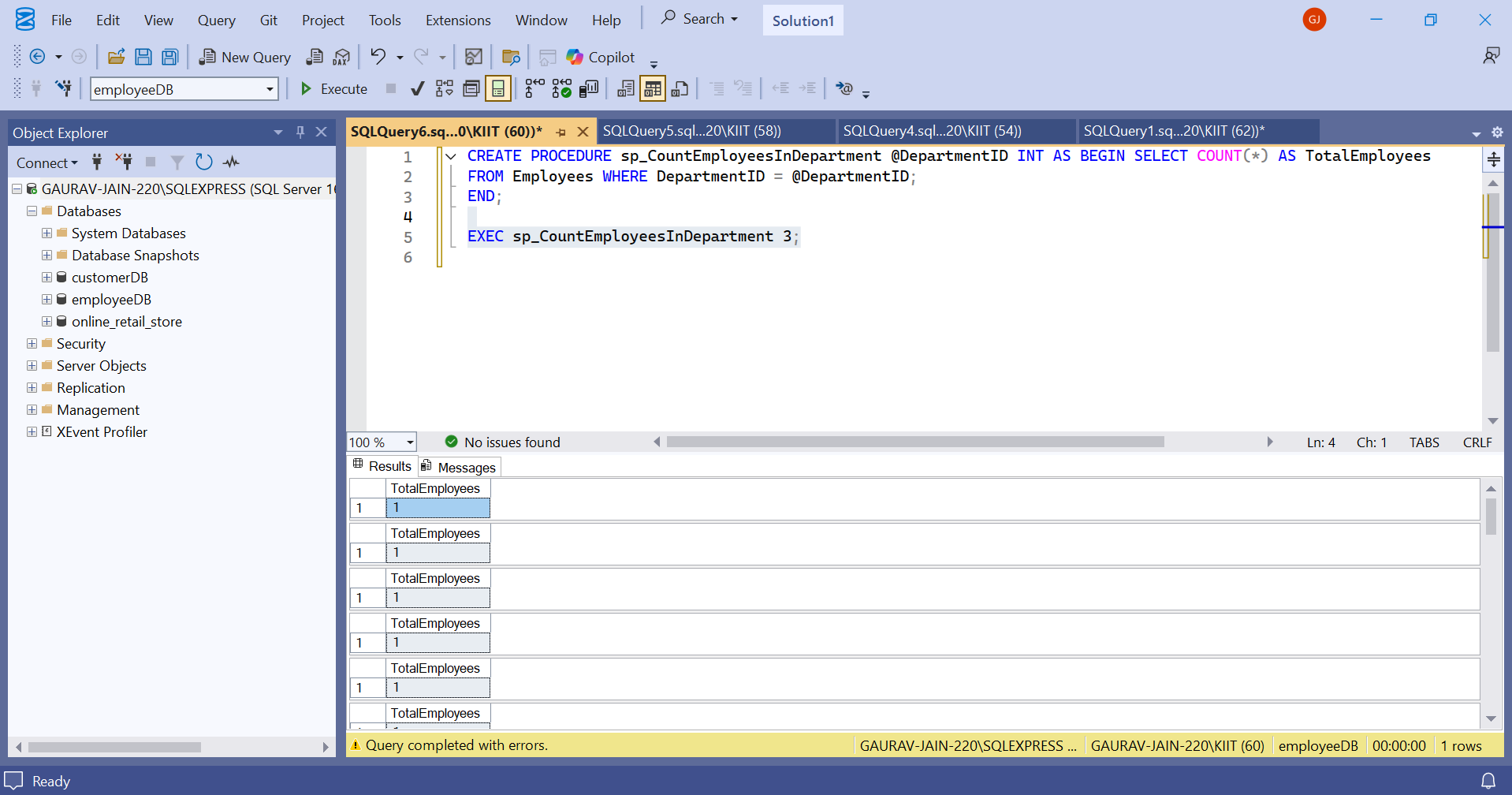
CREATE PROCEDURE sp\_CountEmployeesInDepartment @DepartmentID INT AS BEGIN SELECT COUNT(\*) AS TotalEmployees

FROM Employees WHERE DepartmentID = @DepartmentID;

END;

EXEC sp\_CountEmployeesInDepartment 3;

**Output**



**Ex4.6**

**Code**

CREATE PROCEDURE sp\_TotalSalaryByDepartment @DepartmentID INT,@TotalSalary DECIMAL(10,2) OUTPUT

AS BEGIN

SELECT @TotalSalary = SUM(Salary) FROM Employees WHERE DepartmentID = @DepartmentID;

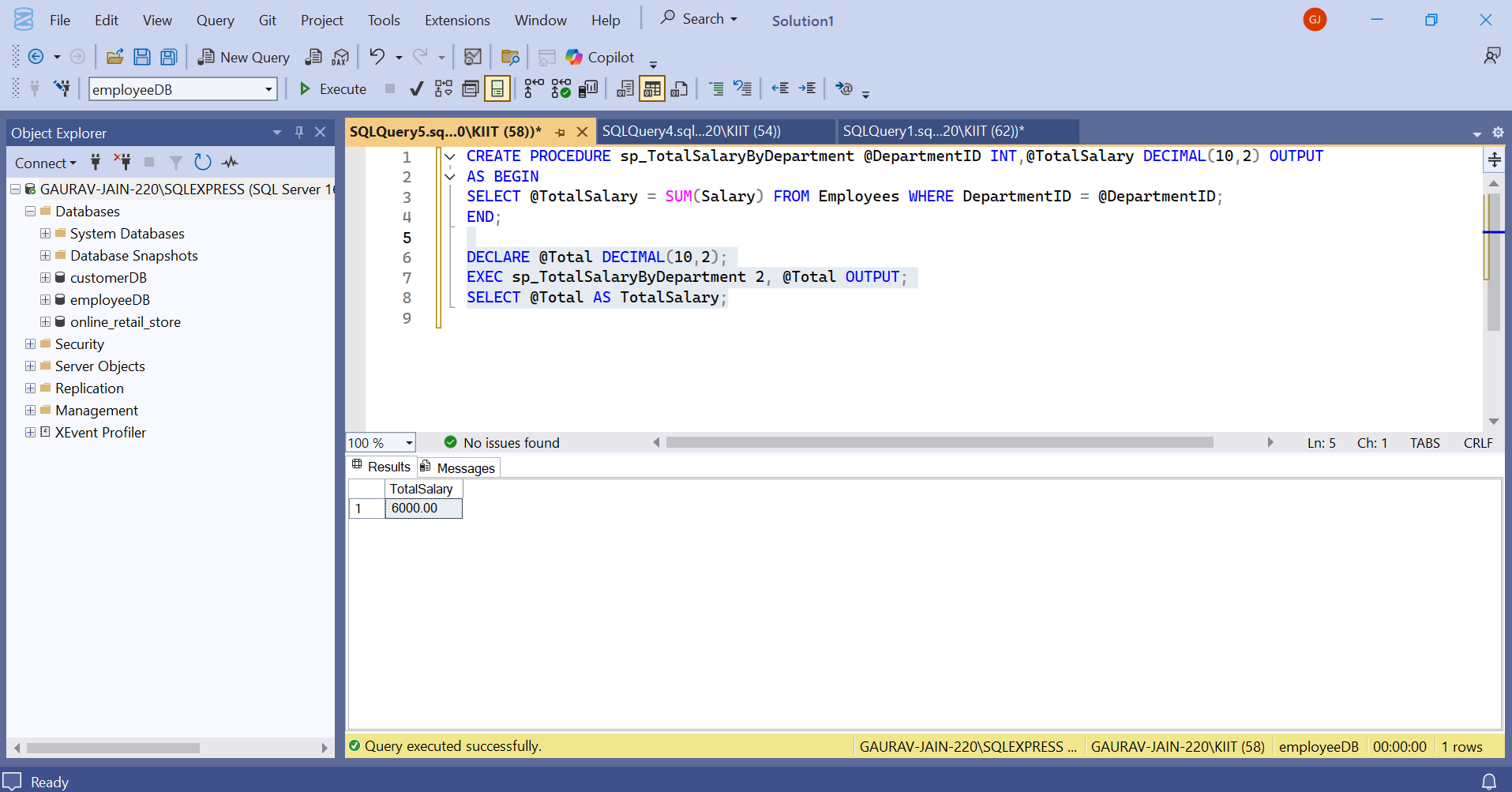
END;

DECLARE @Total DECIMAL(10,2);

EXEC sp\_TotalSalaryByDepartment 2, @Total OUTPUT;

SELECT @Total AS TotalSalary;

**Output:**



**Ex5 Functions**

**Ex5.1**

**Code**

**CREATE FUNCTION fn\_CalculateAnnualSalary**

**(**

**@Salary DECIMAL(10,2)**

**)**

**RETURNS DECIMAL(10,2)**

**AS**

**BEGIN**

**RETURN @Salary \* 12;**

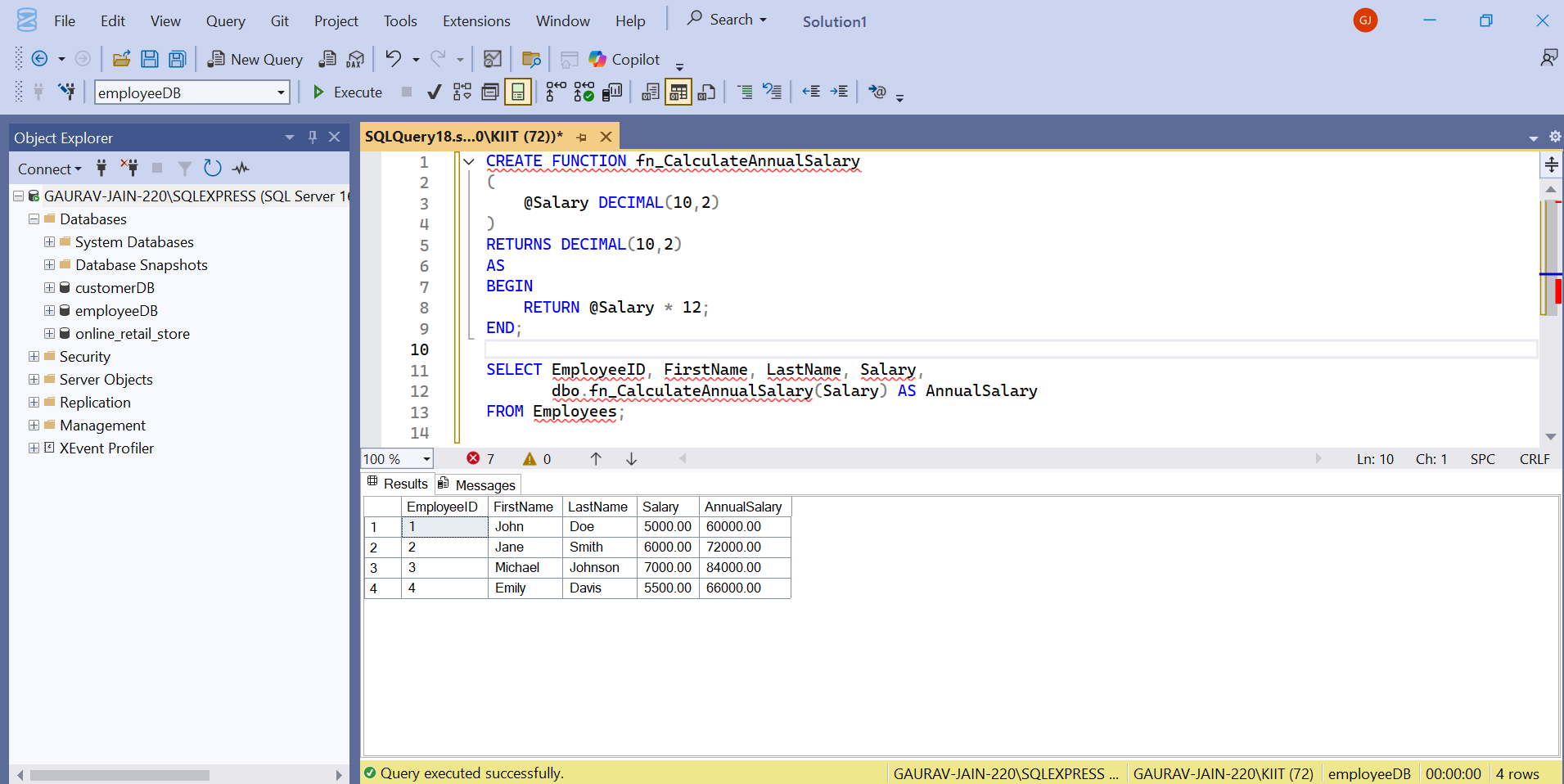
**END;**

**SELECT EmployeeID, FirstName, LastName, Salary,**

**dbo.fn\_CalculateAnnualSalary(Salary) AS AnnualSalary**

**FROM Employees;**

**Output:-**



**Ex5.2**

**Code:**

**CREATE FUNCTION fn\_GetEmployeesByDepartment**

**(**

**@DeptID INT**

**)**

**RETURNS TABLE**

**AS**

**RETURN**

**(**

**SELECT EmployeeID, FirstName, LastName, Salary, JoinDate**

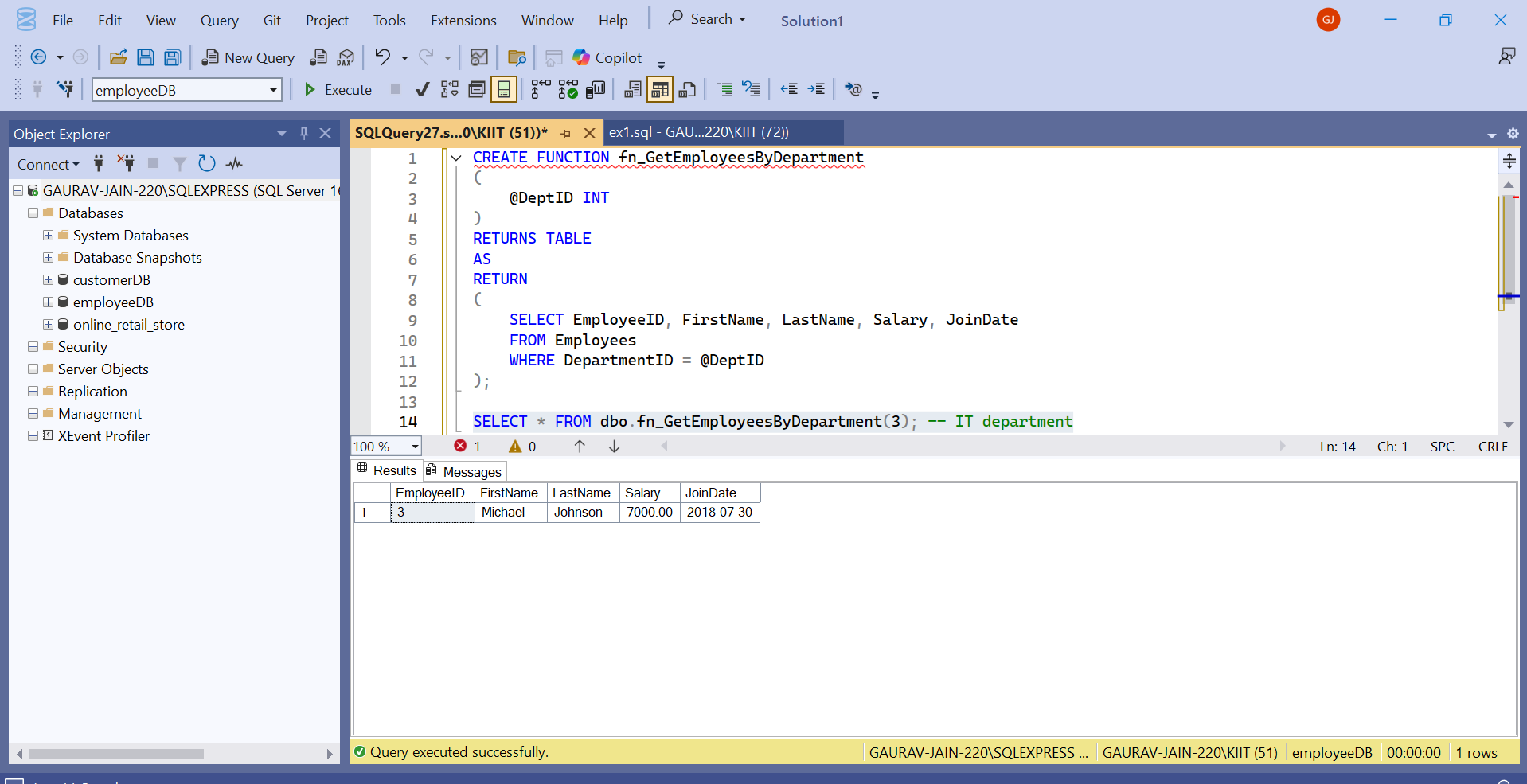
**FROM Employees**

**WHERE DepartmentID = @DeptID**

**);**

**SELECT \* FROM dbo.fn\_GetEmployeesByDepartment(3);**

**Output**



**Ex 5.3**

**Code:**

**CREATE FUNCTION fn\_CalculateBonus**

**(**

**@Salary DECIMAL(10,2)**

**)**

**RETURNS DECIMAL(10,2)**

**AS**

**BEGIN**

**RETURN @Salary \* 0.10;**

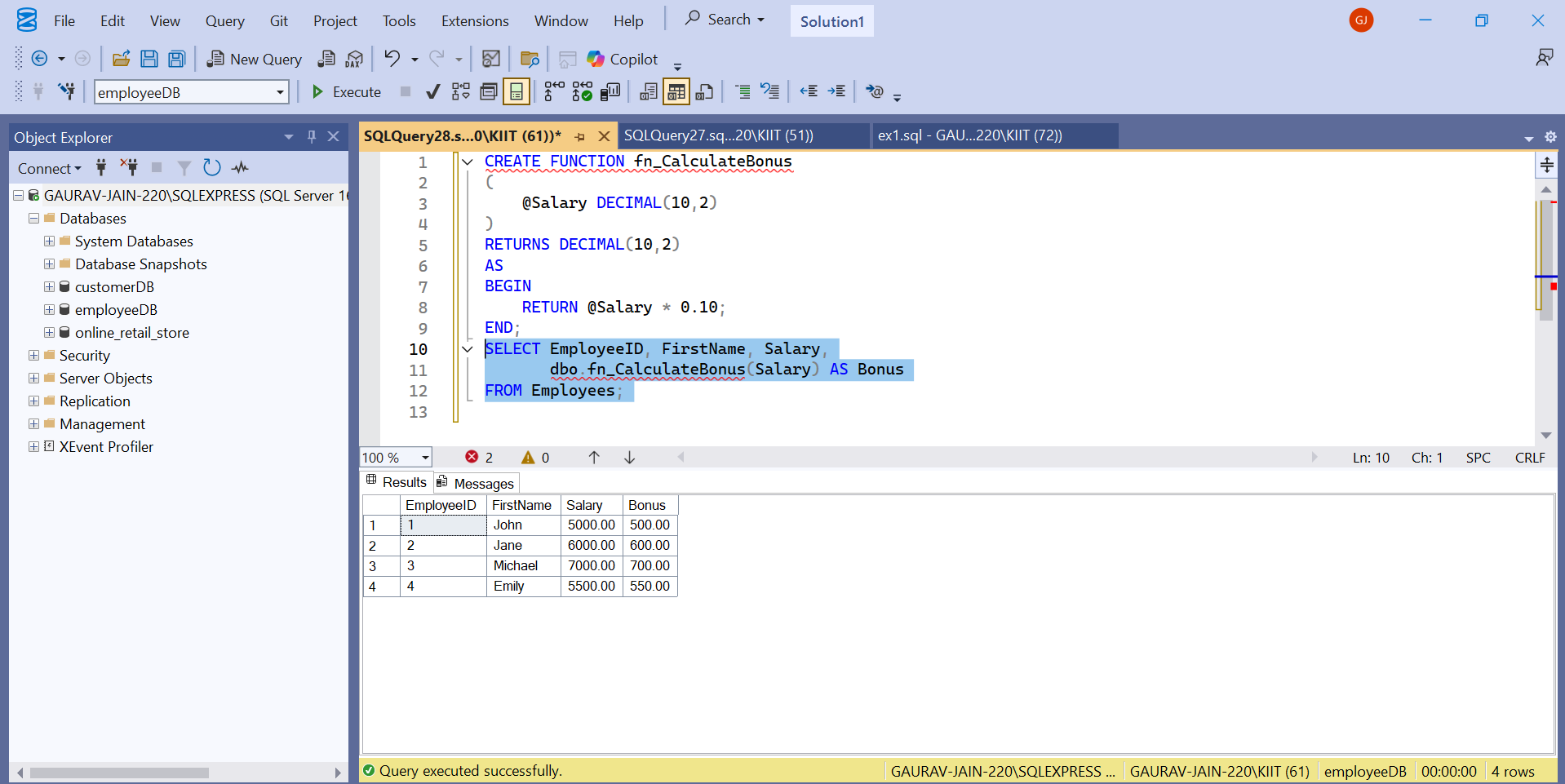
**END;**

**SELECT EmployeeID, FirstName, Salary,**

**dbo.fn\_CalculateBonus(Salary) AS Bonus**

**FROM Employees;**

**Output:**



**Ex5.4**

**Code**

**ALTER FUNCTION fn\_CalculateBonus**

**(**

**@Salary DECIMAL(10,2)**

**)**

**RETURNS DECIMAL(10,2)**

**AS**

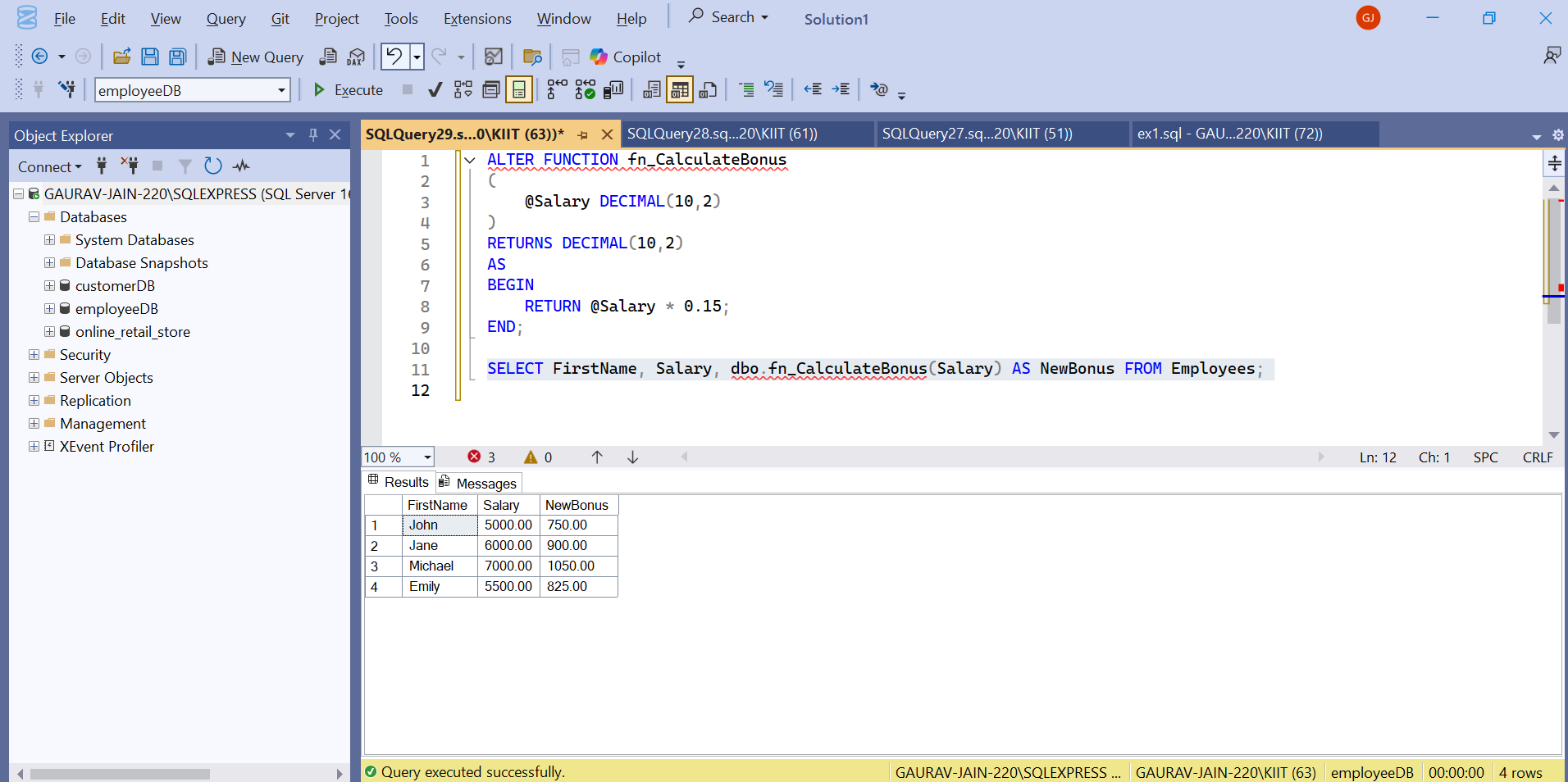
**BEGIN**

**RETURN @Salary \* 0.15;**

**END;**

**SELECT FirstName, Salary, dbo.fn\_CalculateBonus(Salary) AS NewBonus FROM Employees;**

**Output:**



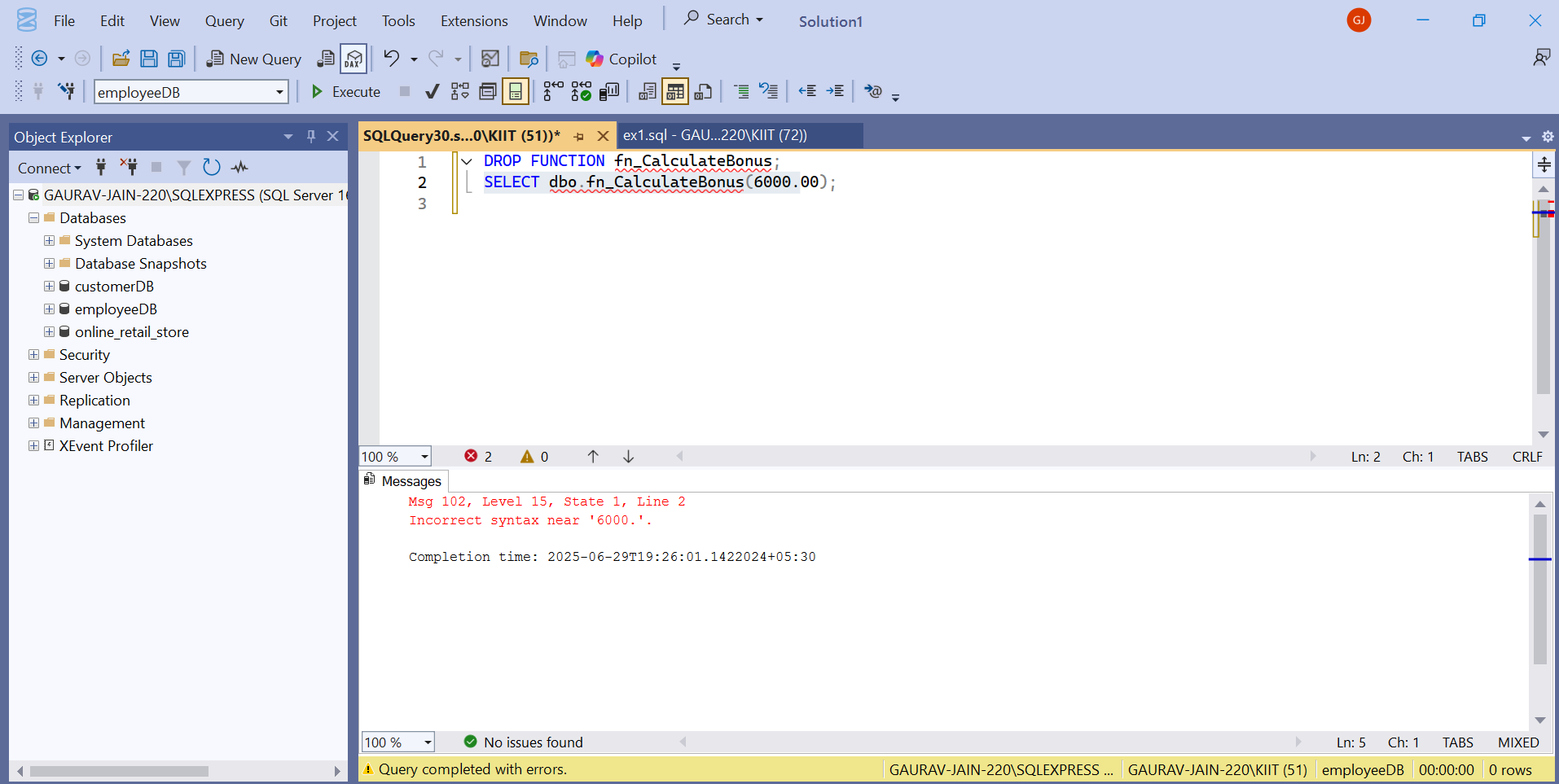
**Ex5.5**

**Code:**

**DROP FUNCTION fn\_CalculateBonus;**

**SELECT dbo.fn\_CalculateBonus(6000.00);**

**Output:**



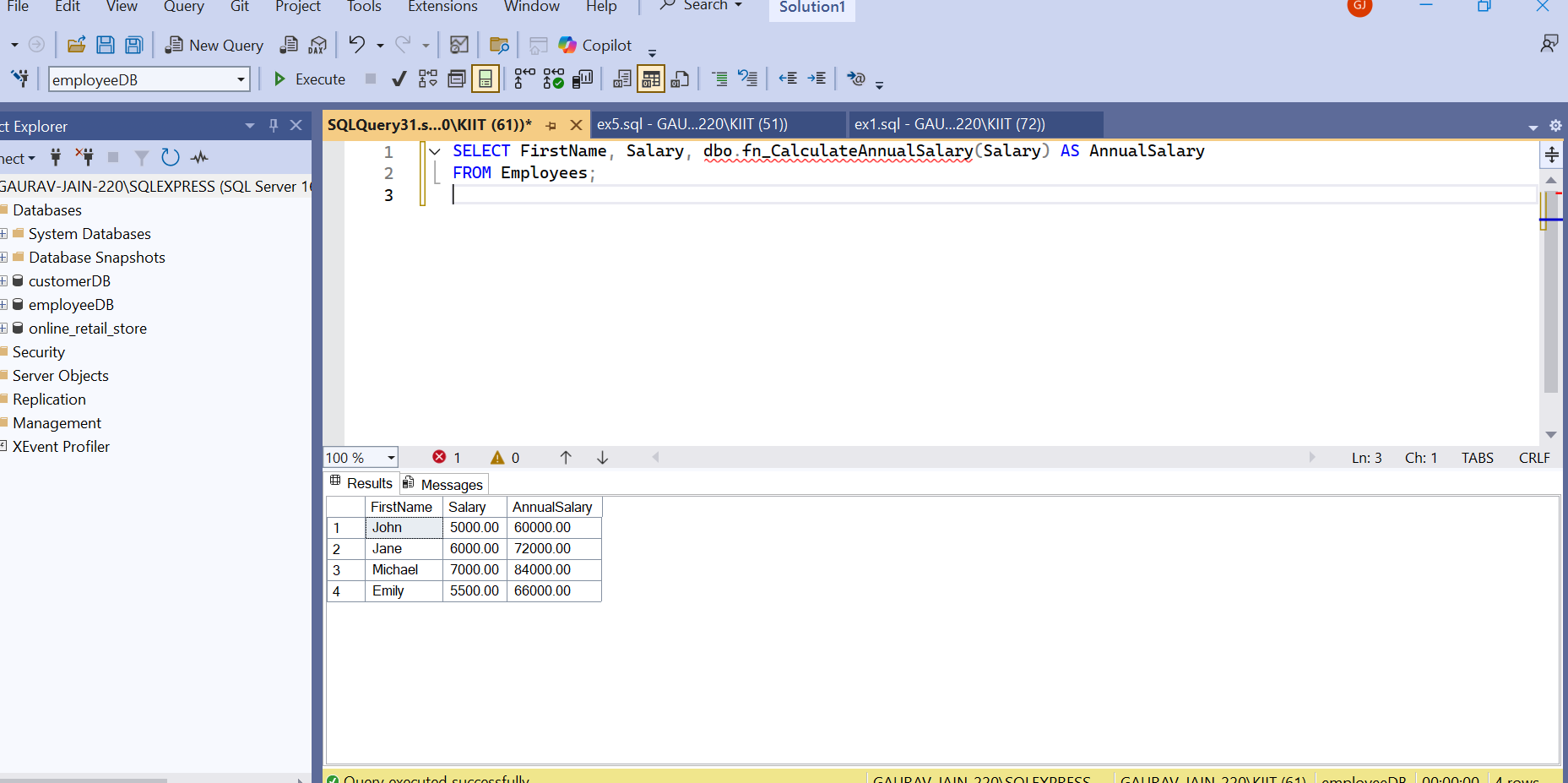
**Ex5.6**

**Code:**

**SELECT FirstName, Salary, dbo.fn\_CalculateAnnualSalary(Salary) AS AnnualSalary**

**FROM Employees;**

**Output:**



**Ex5.7**

**Code:**

**SELECT dbo.fn\_CalculateAnnualSalary(Salary) AS AnnualSalary**

**FROM Employees**

**WHERE EmployeeID = 1;**

**Output:**

