WEEK - 2 (Module - 2)

Yash Raj

SuperSet ID - (6364442)

**Exercise 1: Ranking and Window Functions :**

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

INSERT INTO Products (ProductID, ProductName, Category, Price) VALUES

(1, 'Laptop A', 'Electronics', 1000),

(2, 'Laptop B', 'Electronics', 1000),

(3, 'Phone A', 'Electronics', 800),

(4, 'Phone B', 'Electronics', 800),

(5, 'Speaker', 'Electronics', 500),

(6, 'Chair A', 'Furniture', 300),

(7, 'Chair B', 'Furniture', 300),

(8, 'Table', 'Furniture', 200),

(9, 'Couch', 'Furniture', 200);

SELECT

ProductID,

ProductName,

Category,

Price,

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

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

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

FROM Products;

WITH Ranked AS (

SELECT

\*,

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

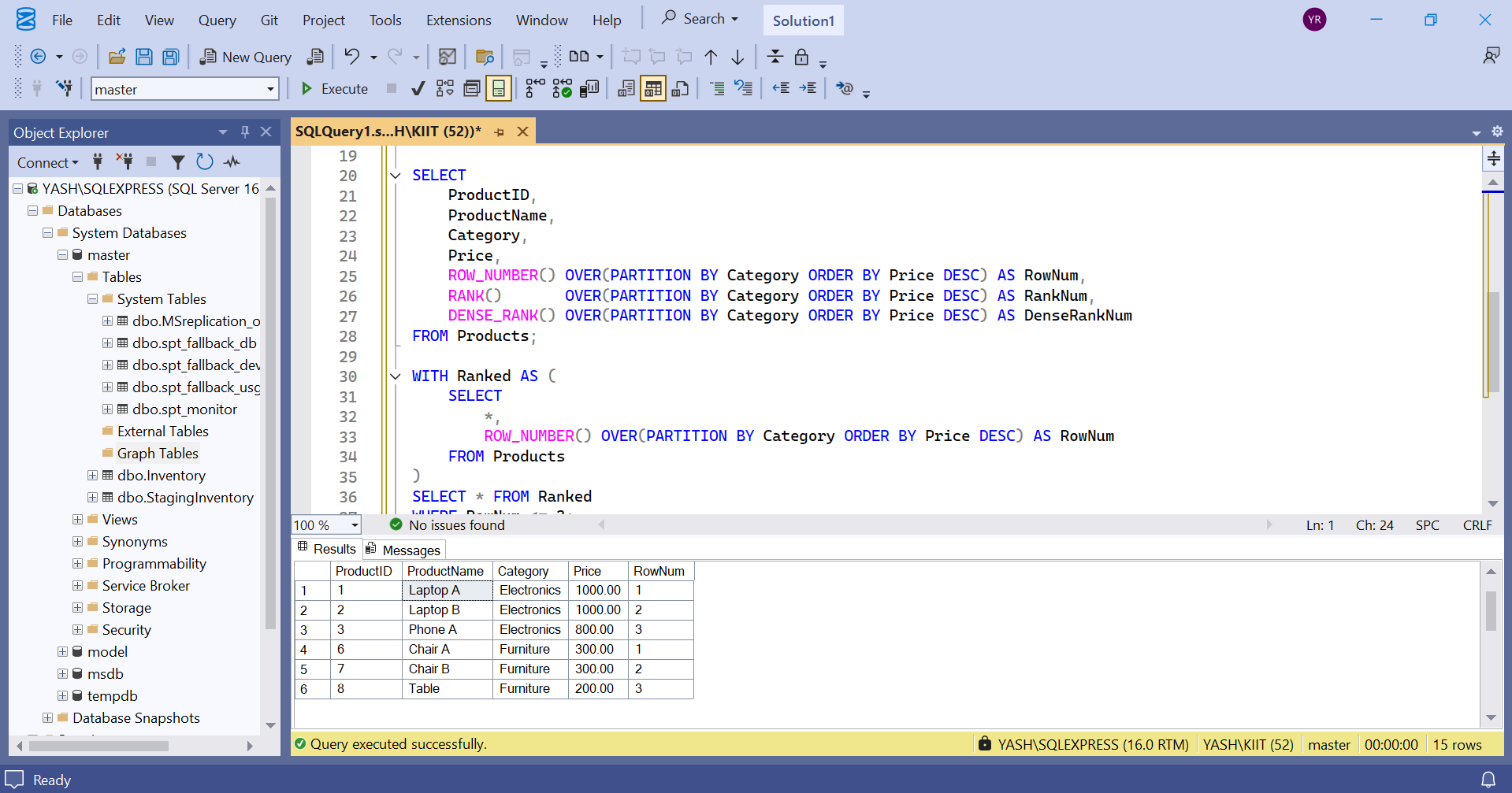
FROM Products

)

SELECT \* FROM Ranked

WHERE RowNum <= 3;

**Output:**



**Exercise 2: Aggregation with GROUPING SETS, CUBE, and ROLLUP:**

1. Join Orders, OrderDetails, Customers, and Products.

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

EmployeeName VARCHAR(100),

Department VARCHAR(50),

Region VARCHAR(50)

);

CREATE TABLE Sales (

SaleID INT PRIMARY KEY,

EmployeeID INT,

SaleAmount DECIMAL(10, 2),

SaleDate DATE,

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)

);

INSERT INTO Employees VALUES

(1, 'Aditya', 'HR', 'North'),

(2, 'Uditya', 'HR', 'South'),

(3, 'Raj', 'Sales', 'North'),

(4, 'Rahul', 'Sales', 'West'),

(5, 'Rohit', 'IT', 'South');

INSERT INTO Sales VALUES

(101, 1, 1000.00, '2024-01-10'),

(102, 2, 1500.00, '2024-01-11'),

(103, 3, 2000.00, '2024-01-12'),

(104, 4, 2500.00, '2024-01-13'),

(105, 5, 1800.00, '2024-01-14'),

(106, 1, 1200.00, '2024-02-10'),

(107, 3, 800.00, '2024-02-12');

SELECT

E.Region,

E.Department,

SUM(S.SaleAmount) AS TotalSales

FROM Sales S

JOIN Employees E ON S.EmployeeID = E.EmployeeID

GROUP BY GROUPING SETS (

(E.Region),

(E.Department),

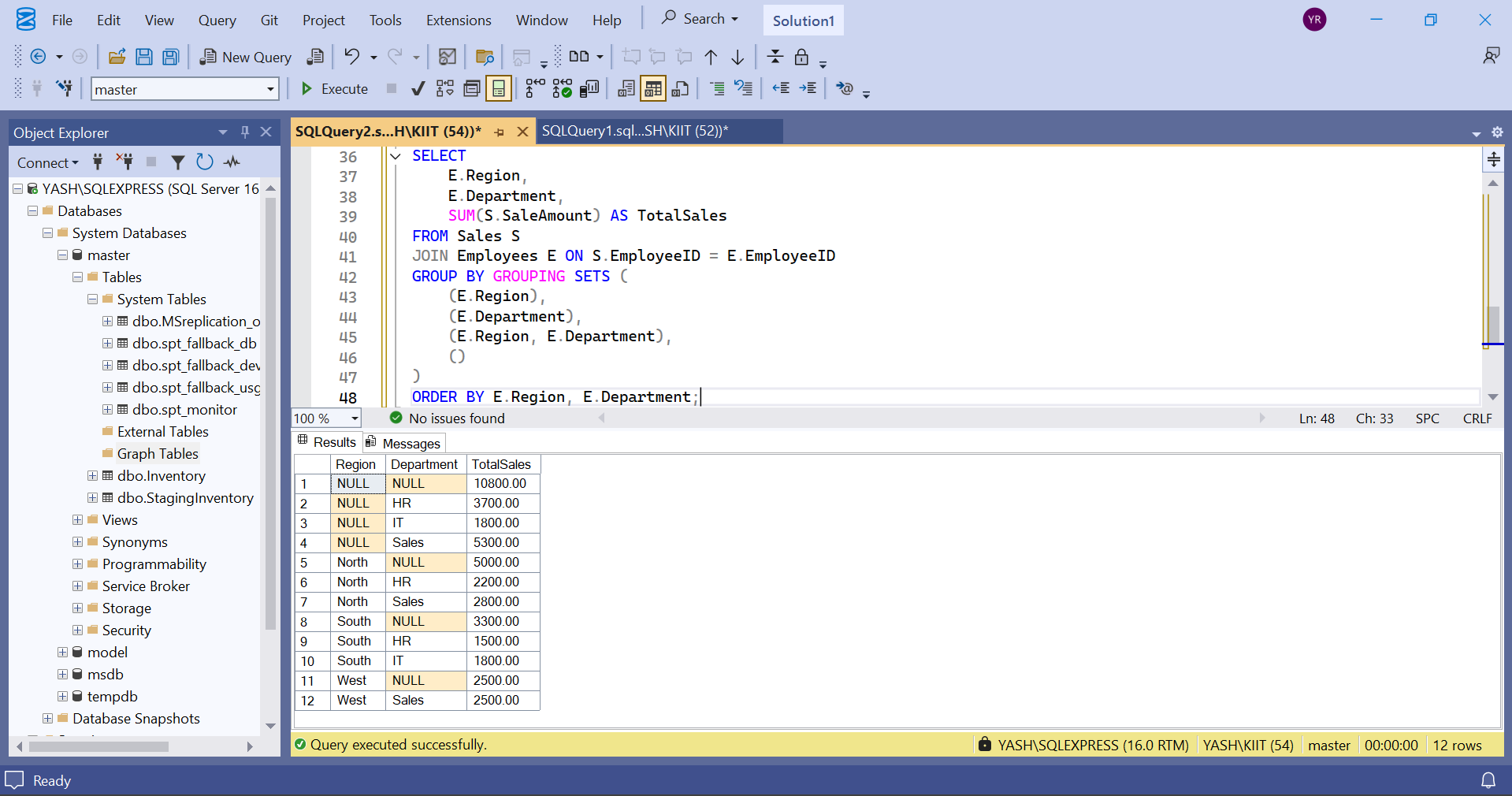
(E.Region, E.Department),

()

)

ORDER BY E.Region, E.Department;

Output :



1. **Use GROUPING SETS to get totals by Region, Category, and both.**

SELECT

E.Region,

E.Department,

SUM(S.SaleAmount) AS TotalSales

FROM Sales S

JOIN Employees E ON S.EmployeeID = E.EmployeeID

GROUP BY GROUPING SETS (

(E.Region),

(E.Department),

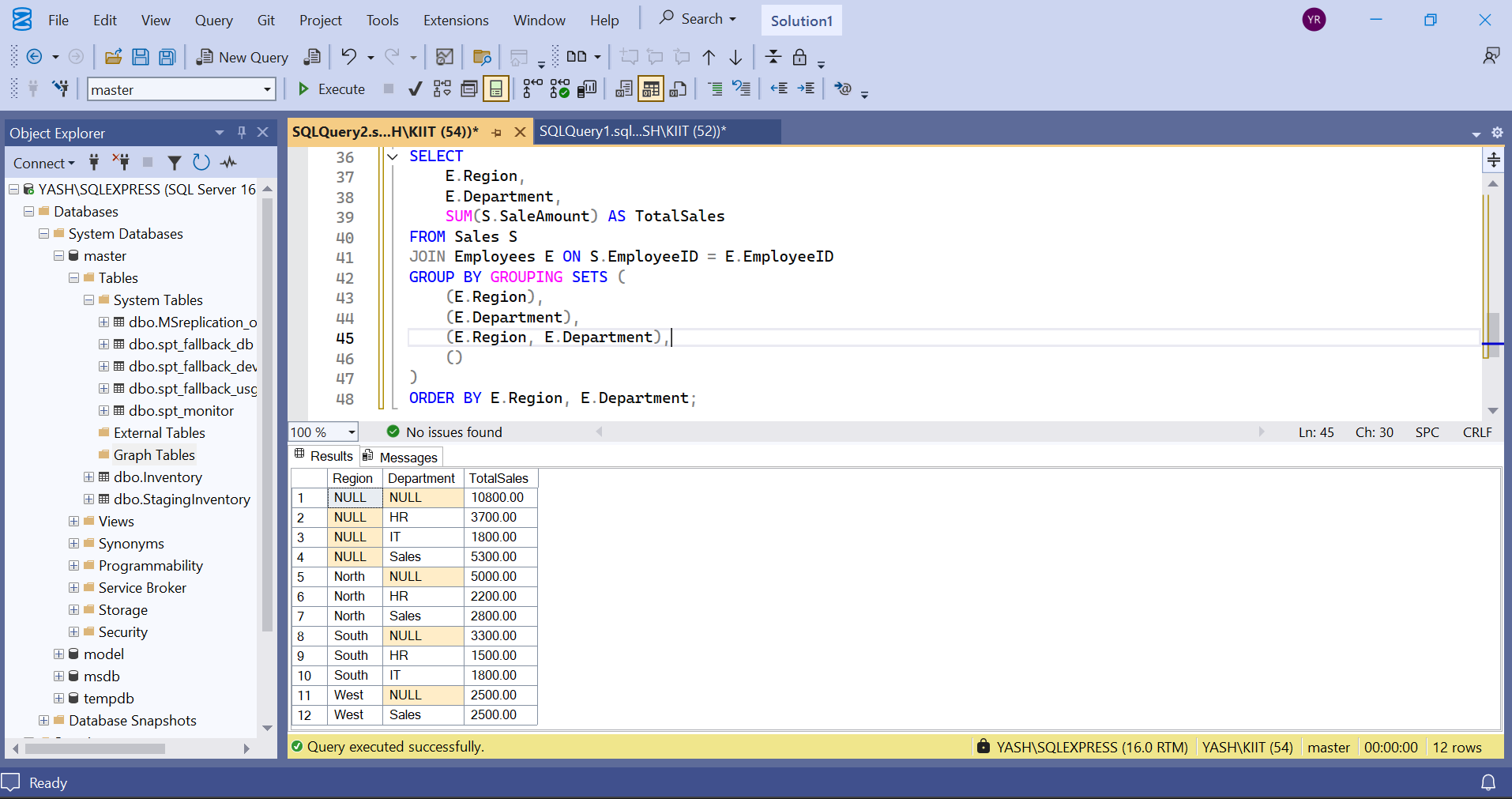
(E.Region, E.Department),

()

)

ORDER BY E.Region, E.Department;

**Output**



1. **Use ROLLUP to get subtotals and grand totals.**

SELECT

E.Region,

E.Department,

SUM(S.SaleAmount) AS TotalSales

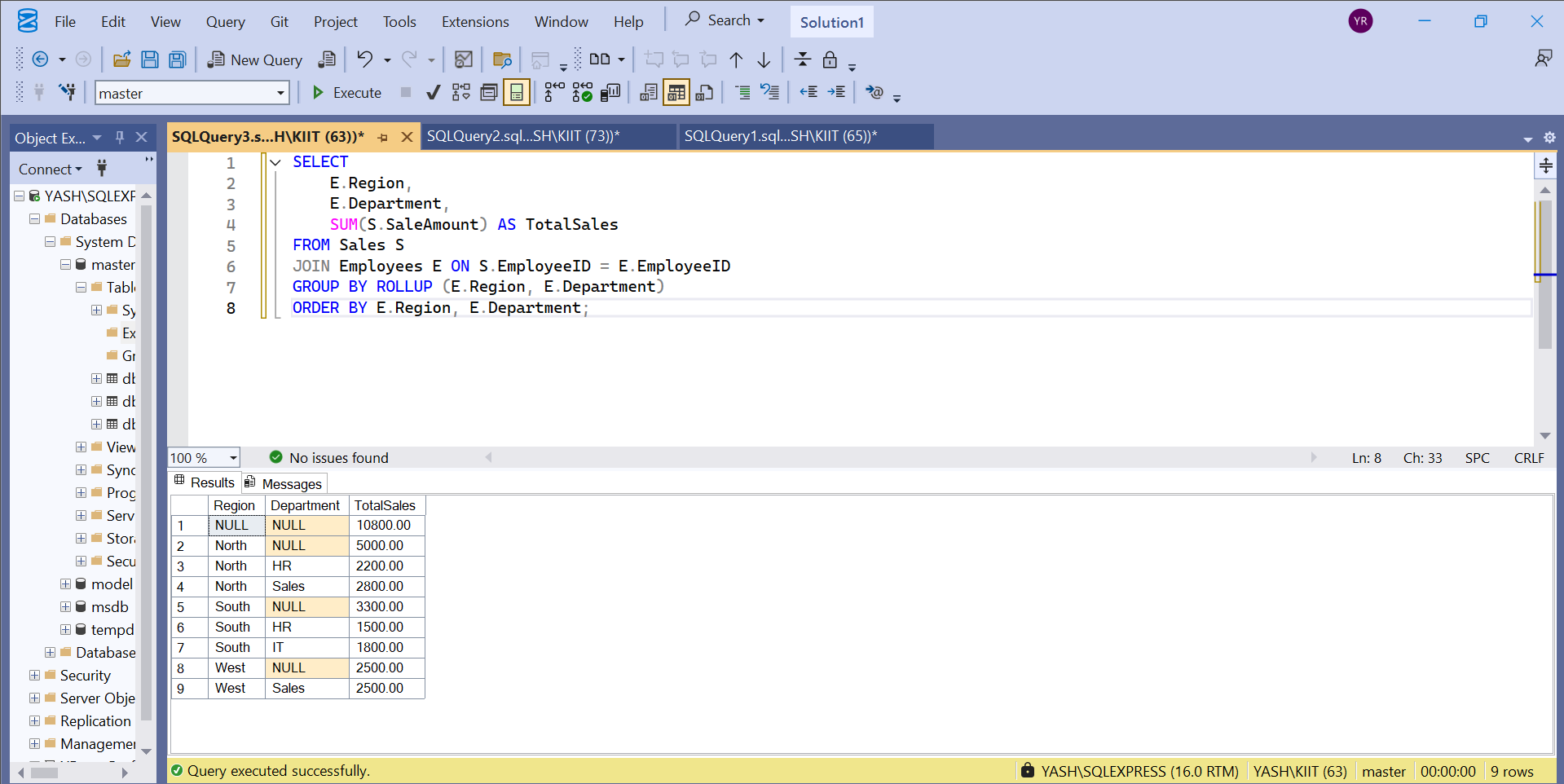
FROM Sales S

JOIN Employees E ON S.EmployeeID = E.EmployeeID

GROUP BY ROLLUP (E.Region, E.Department)

ORDER BY E.Region, E.Department;

**Output:**



1. Use CUBE to get all combinations of Region and Category.

SELECT

E.Region,

E.Department,

GROUPING(E.Region) AS RegionGrouped,

GROUPING(E.Department) AS DepartmentGrouped,

SUM(S.SaleAmount) AS TotalSales

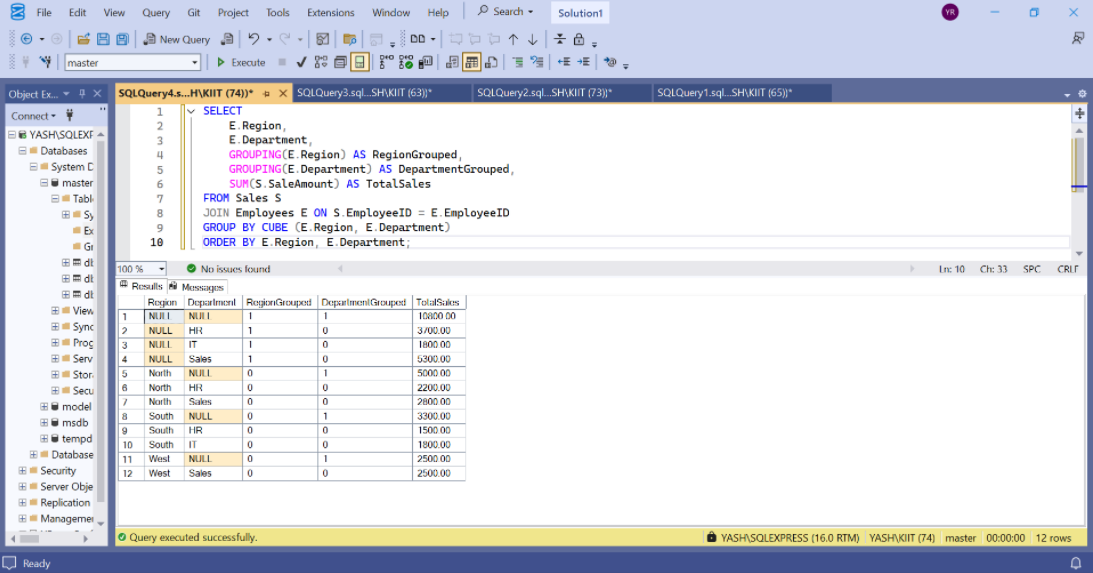
FROM Sales S

JOIN Employees E ON S.EmployeeID = E.EmployeeID

GROUP BY CUBE (E.Region, E.Department)

ORDER BY E.Region, E.Department;

**Output :**



**Exercise 3: CTEs and MERGE**

1. **Create a recursive CTE to generate dates from '2025-01-01' to '2025-01-31'**

WITH CalendarCTE AS (

SELECT CAST('2025-01-01' AS DATE) AS CalendarDate

UNION ALL

SELECT DATEADD(DAY, 1, CalendarDate)

FROM CalendarCTE

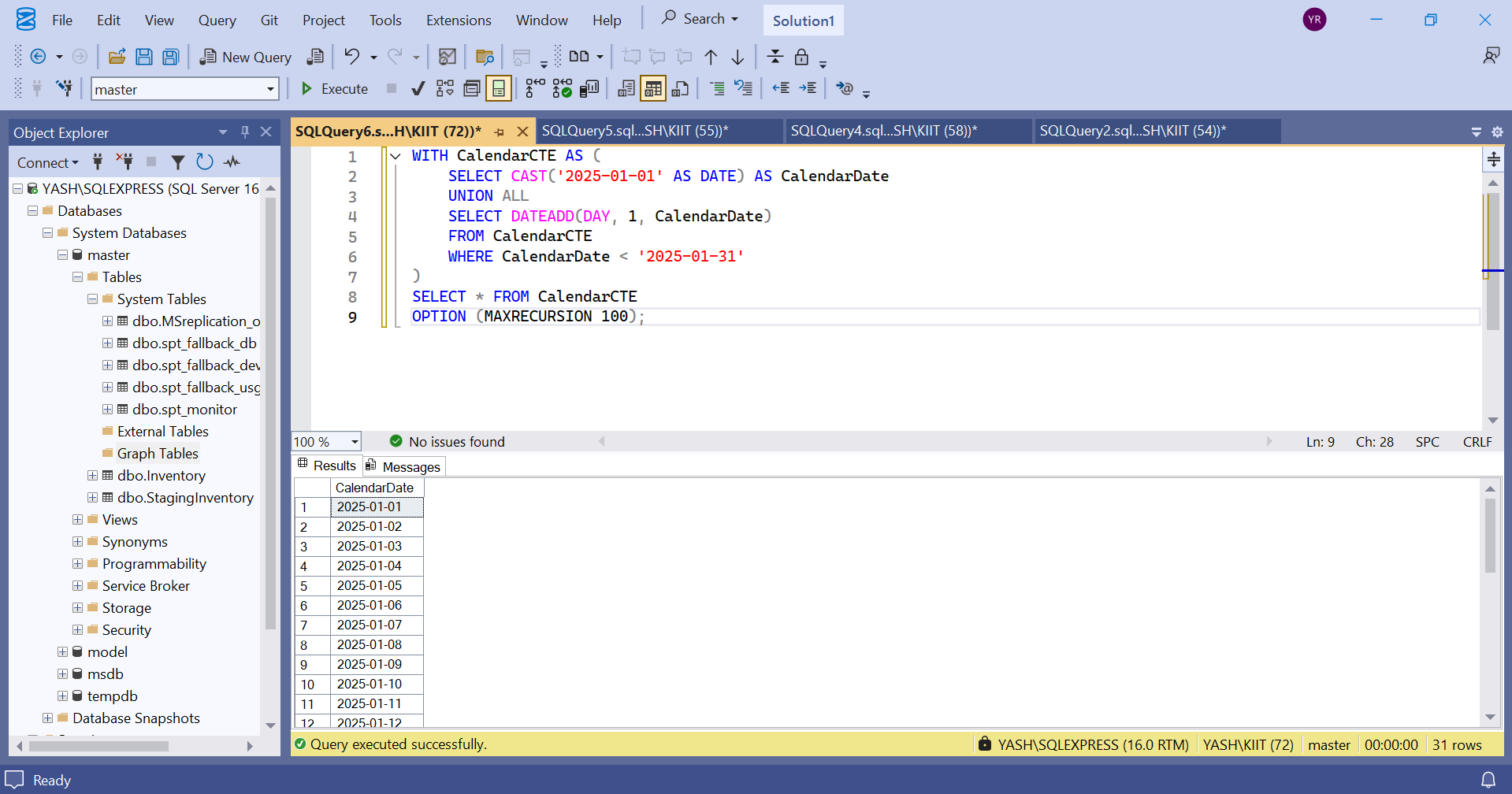
WHERE CalendarDate < '2025-01-31'

)

SELECT \* FROM CalendarCTE

OPTION (MAXRECURSION 100);

Output :



1. **Create a StagingProducts table with updated prices.**

CREATE TABLE Inventory (

ItemID INT PRIMARY KEY,

ItemName VARCHAR(100),

Price DECIMAL(10, 2)

);

INSERT INTO Inventory VALUES

(1, 'Mouse', 250.00),

(2, 'Keyboard', 450.00),

(3, 'Monitor', 5000.00);

CREATE TABLE StagingInventory (

ItemID INT,

ItemName VARCHAR(100),

Price DECIMAL(10, 2)

);

INSERT INTO StagingInventory VALUES

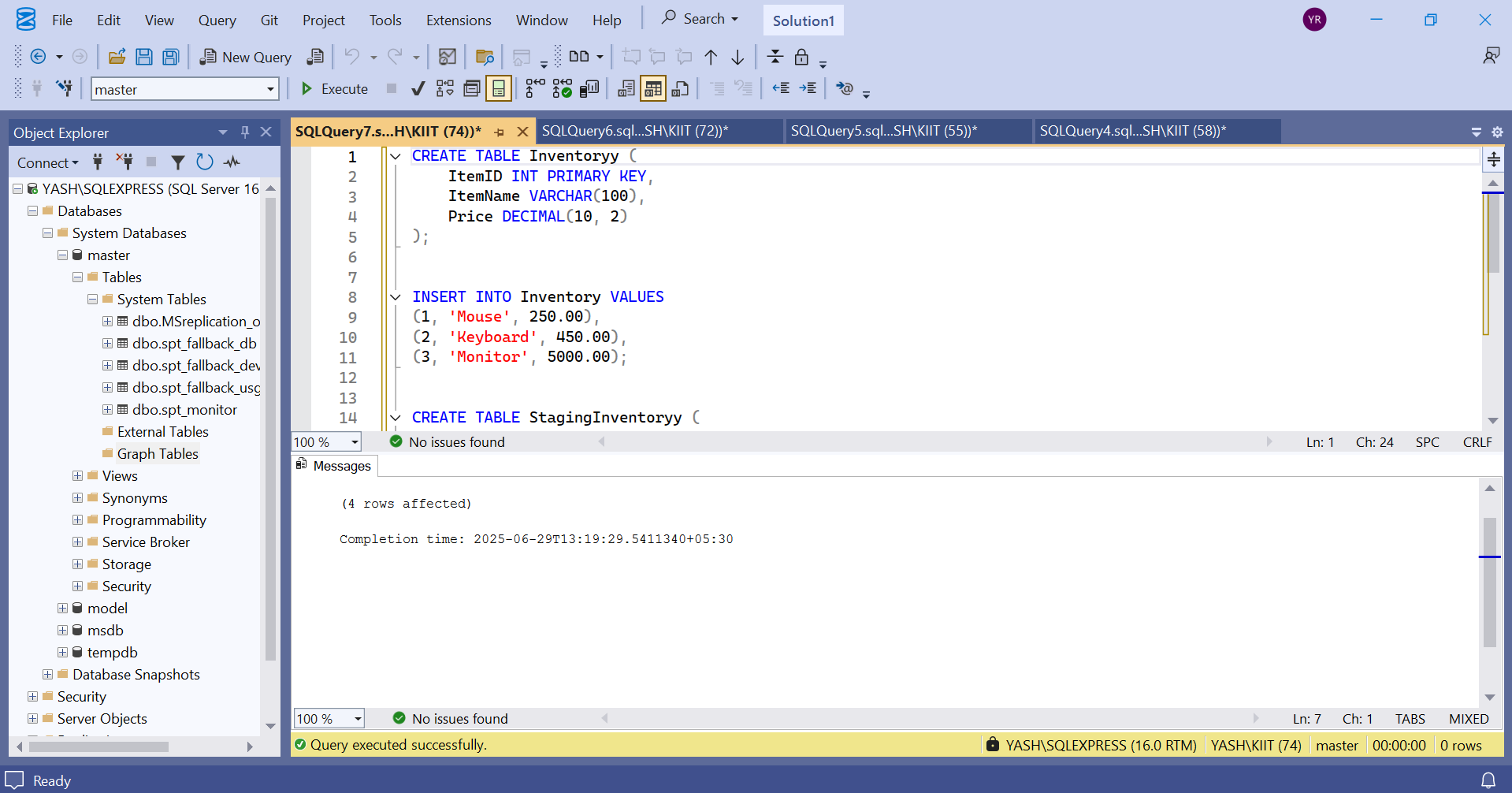
(1, 'Mouse', 275.00),

(3, 'Monitor', 4800.00),

(4, 'Webcam', 1200.00),

(5, 'Speaker', 1800.00);

**Output :**



1. **Use MERGE to update existing products or insert new ones.**

MERGE Inventory AS Target

USING StagingInventory AS Source

ON Target.ItemID = Source.ItemID

WHEN MATCHED THEN

UPDATE SET

Target.ItemName = Source.ItemName,

Target.Price = Source.Price

WHEN NOT MATCHED BY TARGET THEN

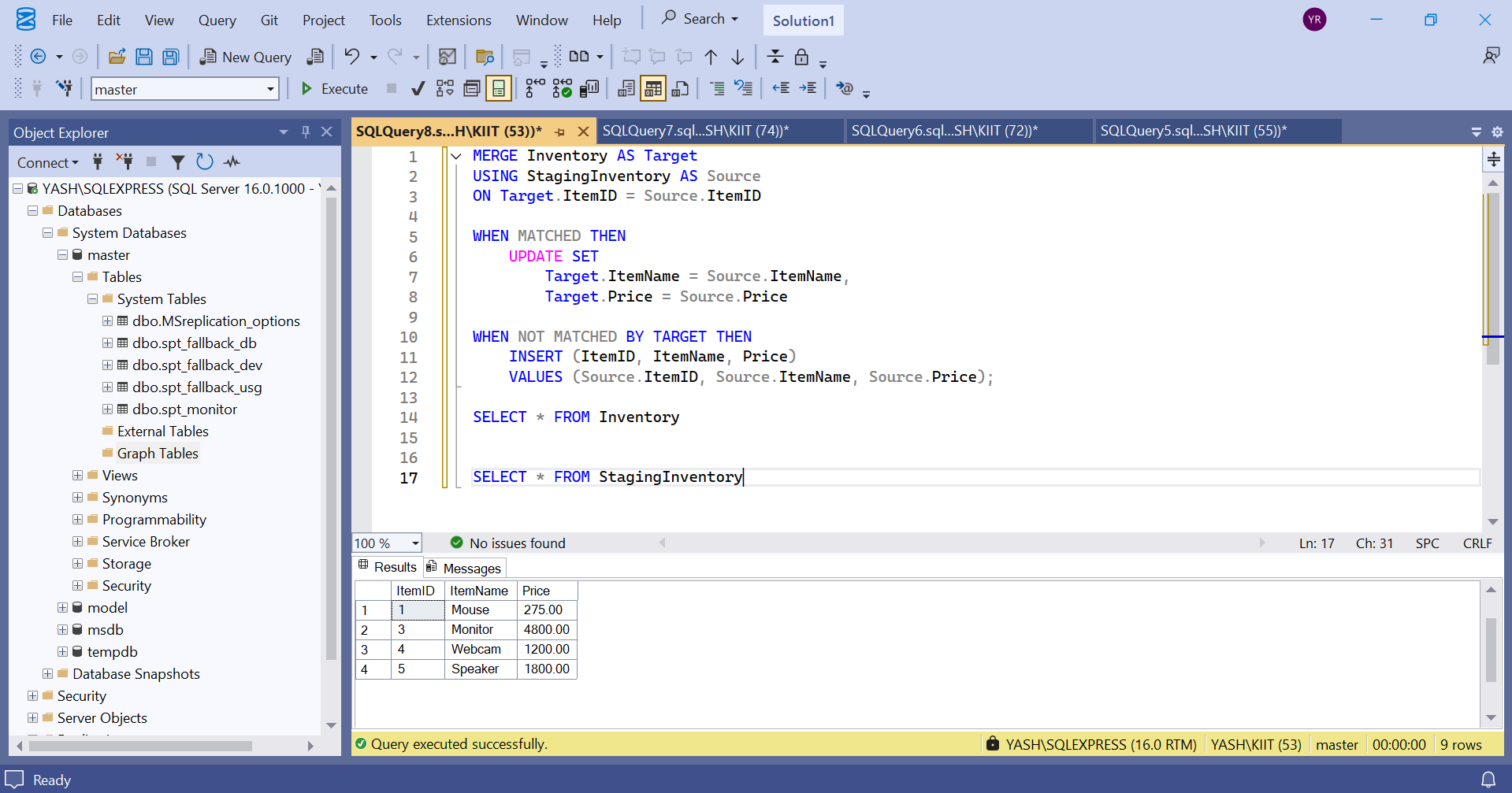
INSERT (ItemID, ItemName, Price)

VALUES (Source.ItemID, Source.ItemName, Source.Price);

SELECT \* FROM Inventory

SELECT \* FROM StagingInventory

Output :



**Exercise 4: PIVOT and UNPIVOT Goal: Transform data for reporting.**

1. Aggregate sales by Product and Month.

CREATE TABLE SalesData (

ProductName VARCHAR(100),

SaleDate DATE,

Quantity INT

);

INSERT INTO SalesData VALUES

('Laptop', '2025-01-05', 10),

('Laptop', '2025-02-12', 15),

('Laptop', '2025-03-20', 8),

('Phone', '2025-01-15', 20),

('Phone', '2025-02-17', 25),

('Phone', '2025-03-10', 12);

SELECT

ProductName,

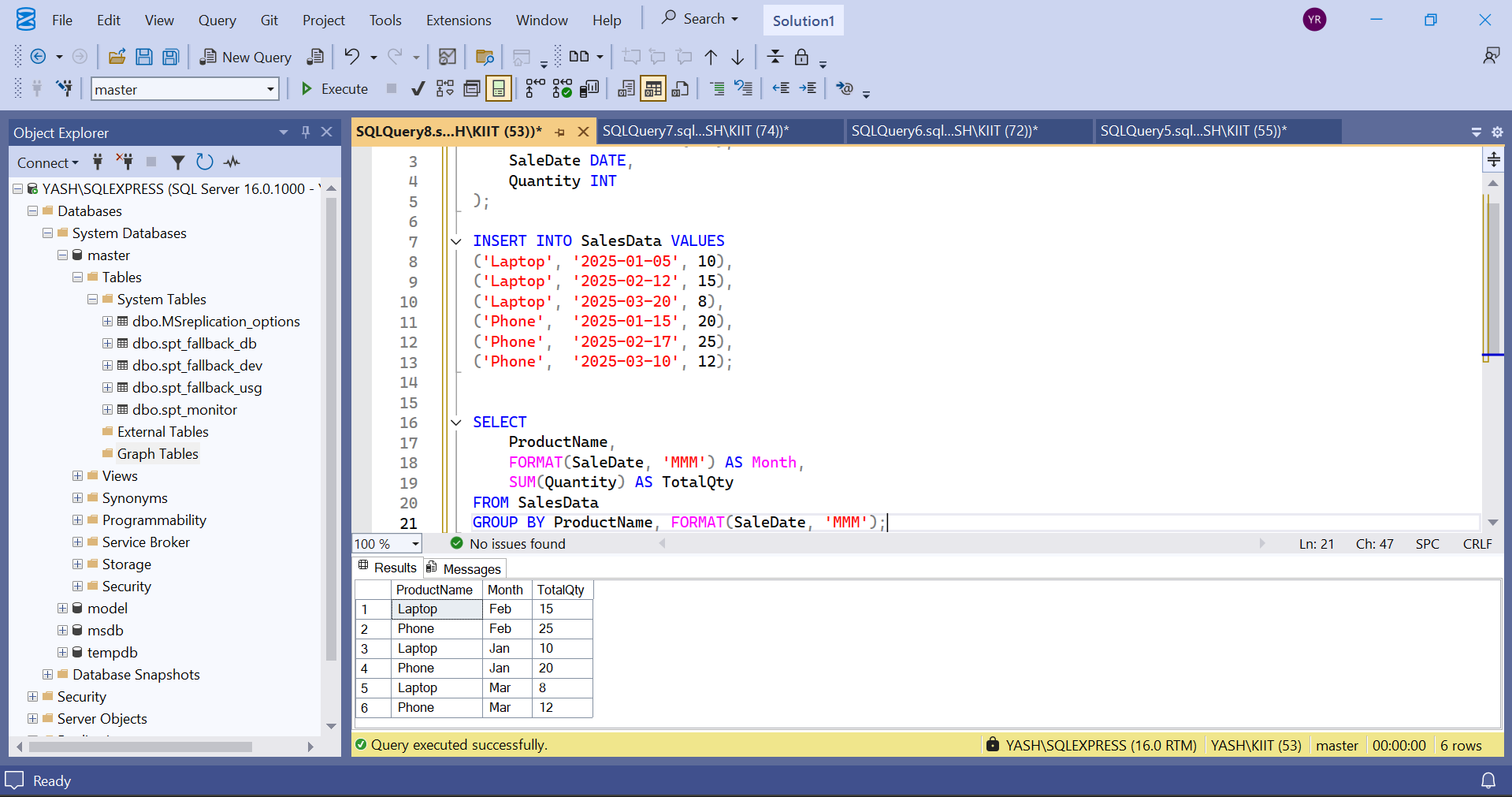
FORMAT(SaleDate, 'MMM') AS Month,

SUM(Quantity) AS TotalQty

FROM SalesData

GROUP BY ProductName, FORMAT(SaleDate, 'MMM');

Output :



1. **Use PIVOT to convert rows into columns (one column per month).**

SELECT \*

FROM (

SELECT

ProductName,

FORMAT(SaleDate, 'MMM') AS SaleMonth,

Quantity

FROM SalesData

) AS SourceTable

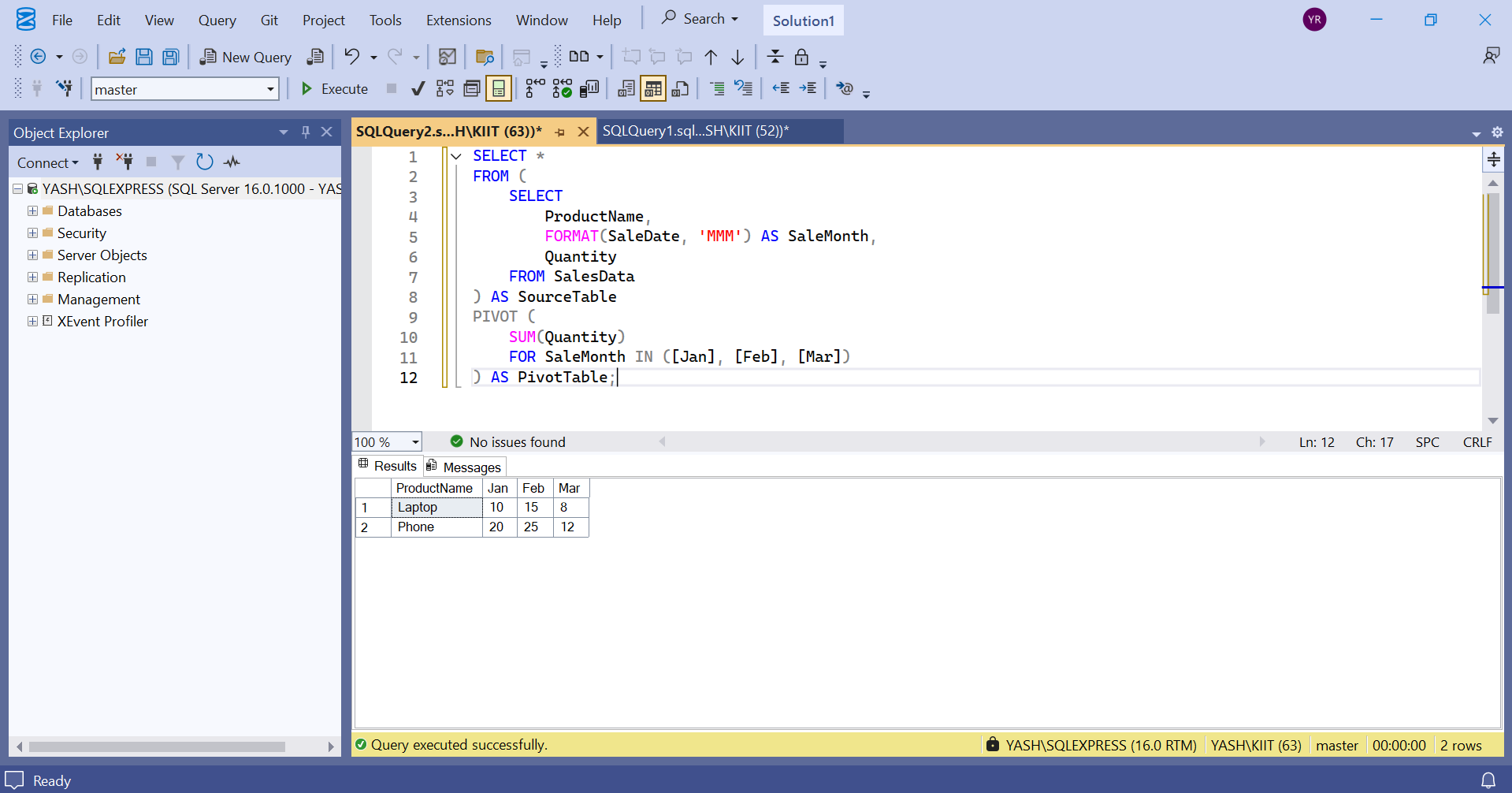
PIVOT (

SUM(Quantity)

FOR SaleMonth IN ([Jan], [Feb], [Mar])

) AS PivotTable;

Output :



1. **Use UNPIVOT to convert the pivoted data back into row format.**

SELECT

ProductName,

SaleMonth,

Quantity

FROM (

SELECT

ProductName, [Jan], [Feb], [Mar]

FROM (

SELECT

ProductName,

FORMAT(SaleDate, 'MMM') AS SaleMonth,

Quantity

FROM SalesData

) AS SourceTable

PIVOT (

SUM(Quantity)

FOR SaleMonth IN ([Jan], [Feb], [Mar])

) AS Pivoted

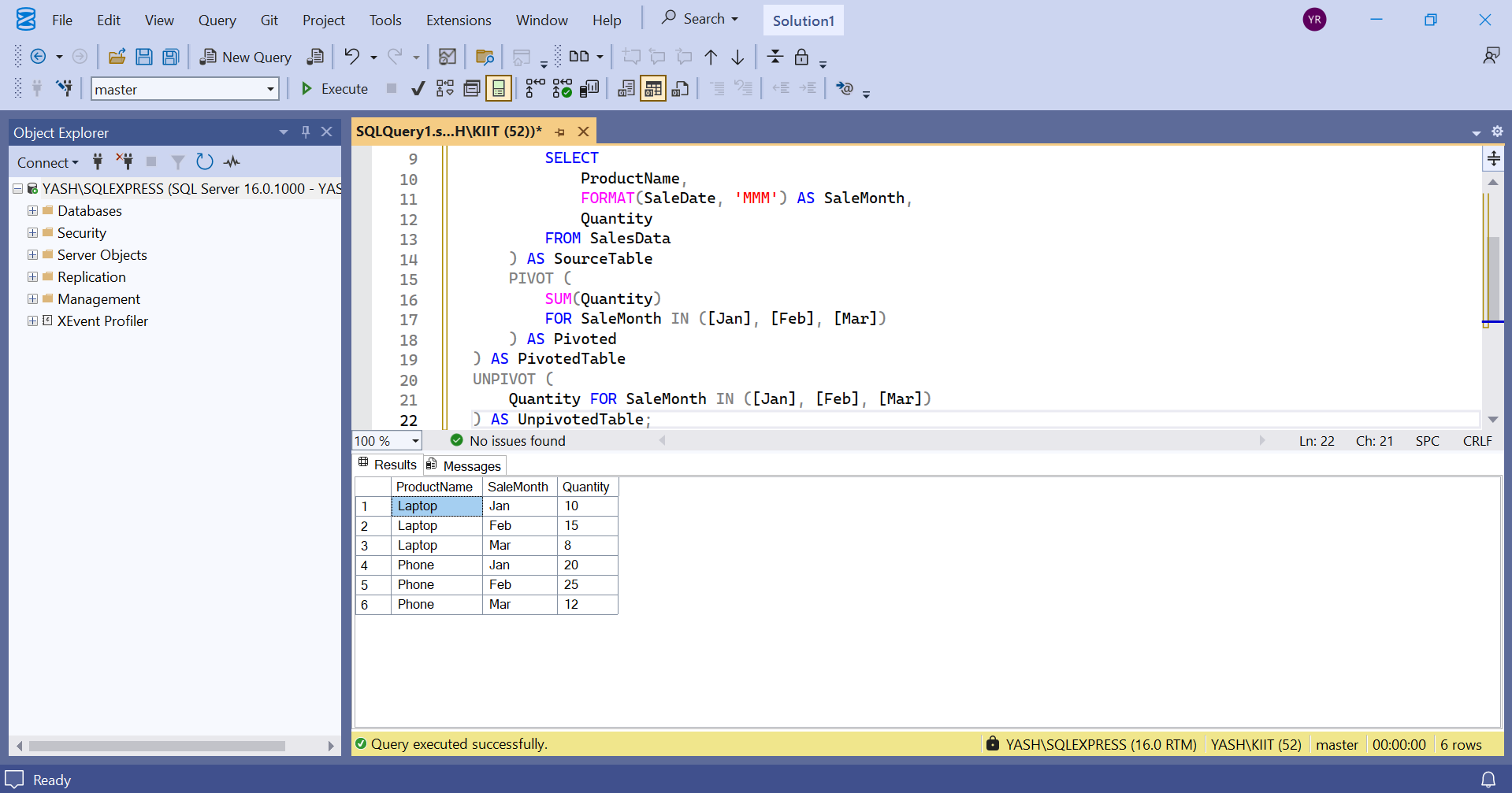
) AS PivotedTable

UNPIVOT (

Quantity FOR SaleMonth IN ([Jan], [Feb], [Mar])

) AS UnpivotedTable;

Output :



**Exercise 5: Using CTE to Simplify a Query**

CREATE TABLE DemoCustomers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100)

);

CREATE TABLE DemoOrders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES DemoCustomers(CustomerID)

);

INSERT INTO DemoCustomers VALUES

(1, 'Yash'),

(2, 'Uditya'),

(3, 'Raj'),

(4, 'Rahul'),

(5, 'Rohit');

INSERT INTO DemoOrders VALUES

(101, 1, '2024-01-01'),

(102, 1, '2024-01-15'),

(103, 1, '2024-02-01'),

(104, 1, '2024-03-01'),

(105, 2, '2024-01-20'),

(106, 2, '2024-02-14'),

(107, 3, '2024-03-10'),

(108, 3, '2024-04-11'),

(109, 3, '2024-05-12'),

(110, 3, '2024-06-01'),

(111, 4, '2024-03-03'),

(112, 5, '2024-03-05'),

(113, 5, '2024-04-05'),

(114, 5, '2024-05-05');

WITH CustomerOrderCounts AS (

SELECT

CustomerID,

COUNT(OrderID) AS OrderCount

FROM DemoOrders

GROUP BY CustomerID

)

SELECT

c.CustomerID,

c.Name,

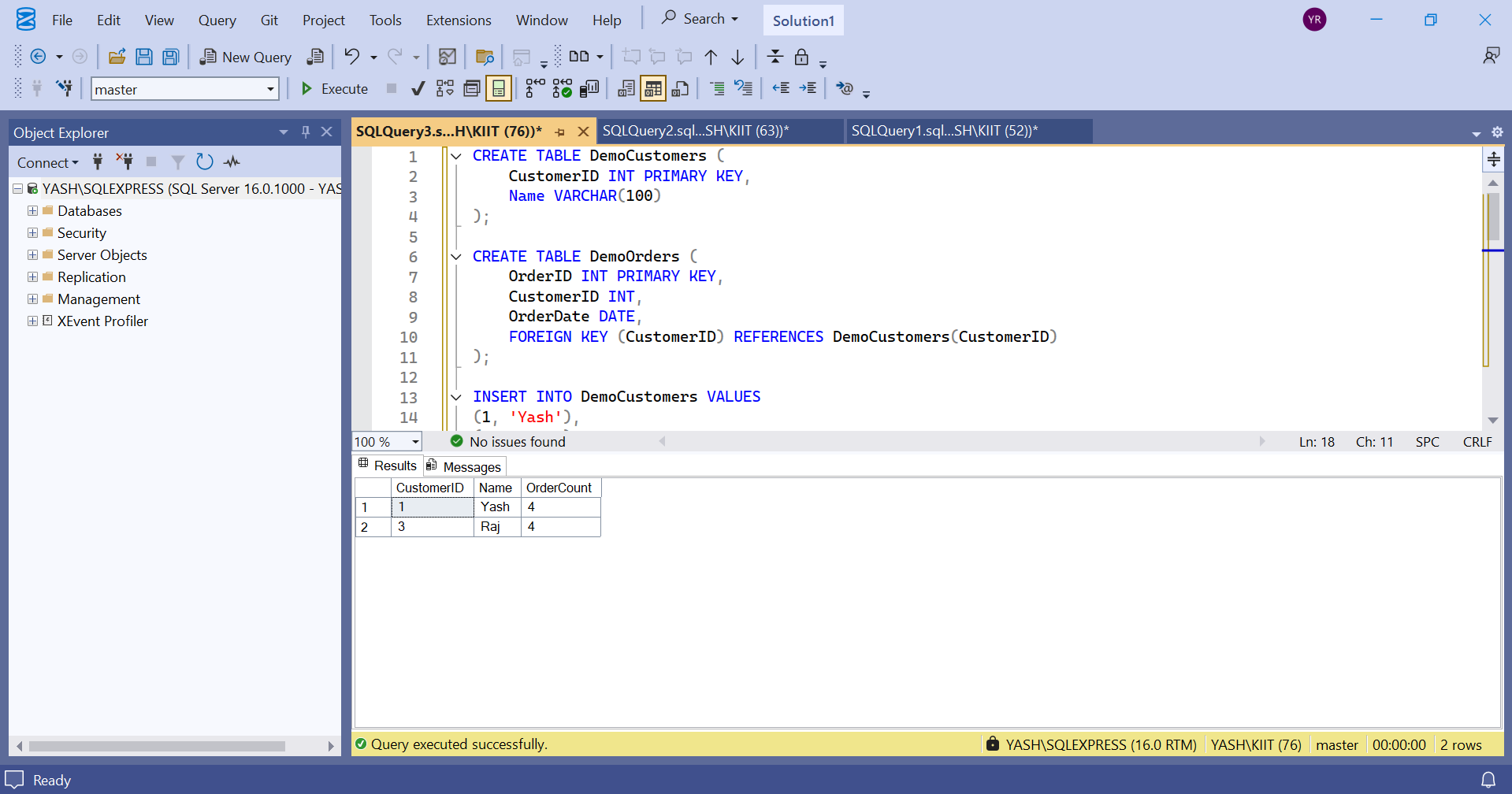
coc.OrderCount

FROM CustomerOrderCounts coc

JOIN DemoCustomers c ON c.CustomerID = coc.CustomerID

WHERE coc.OrderCount > 3;

Output :



**Exercise 2: SQL EXERCISE - Index.SQL :**

**- Question 1: Creating a Non-Clustered Index**

DROP TABLE IF EXISTS OrderDetails;

DROP TABLE IF EXISTS Orders;

DROP TABLE IF EXISTS Products;

DROP TABLE IF EXISTS Customers;

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

Region VARCHAR(50)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10, 2)

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

INSERT INTO Customers (CustomerID, Name, Region) VALUES

(1, 'Yash', 'North'),

(2, 'Uditya', 'South'),

(3, 'Raj', 'East'),

(4, 'Rahul', 'West'),

(5, 'Rohit', 'Central');

INSERT INTO Products (ProductID, ProductName, Category, Price) VALUES

(1, 'Laptop', 'Electronics', 1200.00),

(2, 'Smartphone', 'Electronics', 800.00),

(3, 'Tablet', 'Electronics', 600.00),

(4, 'Headphones', 'Accessories', 150.00);

INSERT INTO Orders (OrderID, CustomerID, OrderDate) VALUES

(1, 1, '2023-01-15'),

(2, 2, '2023-02-20'),

(3, 3, '2023-03-25'),

(4, 4, '2023-04-30'),

(5, 5, '2023-05-05');

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity) VALUES

(1, 1, 1, 1),

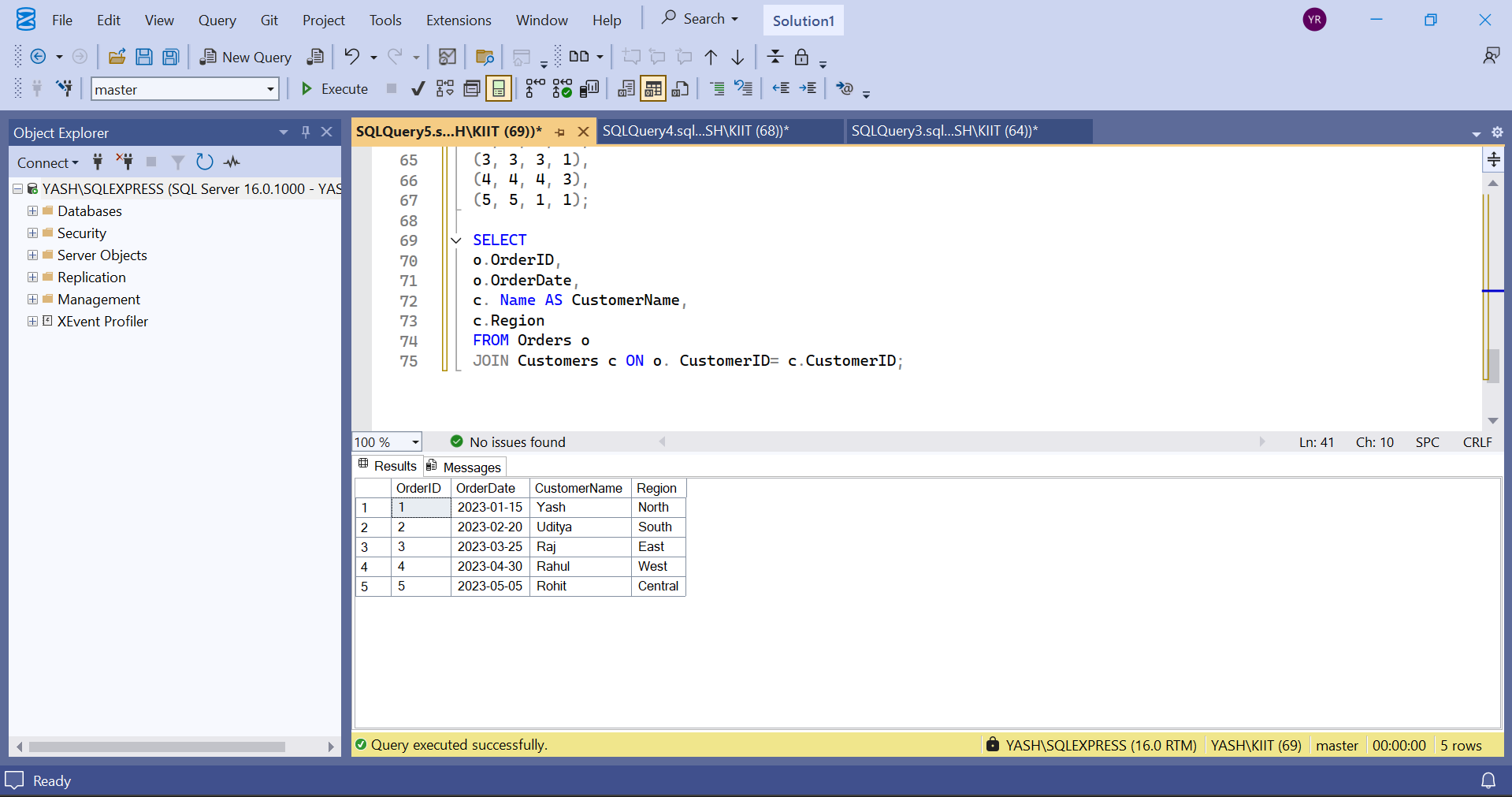
(2, 2, 2, 2),

(3, 3, 3, 1),

(4, 4, 4, 3),

(5, 5, 1, 1);

**Output:**



**Question 2: Creating a Clustered Index:**

SELECT

c.CustomerID,

c.Name,

SUM(od.Quantity) AS TotalQuantity

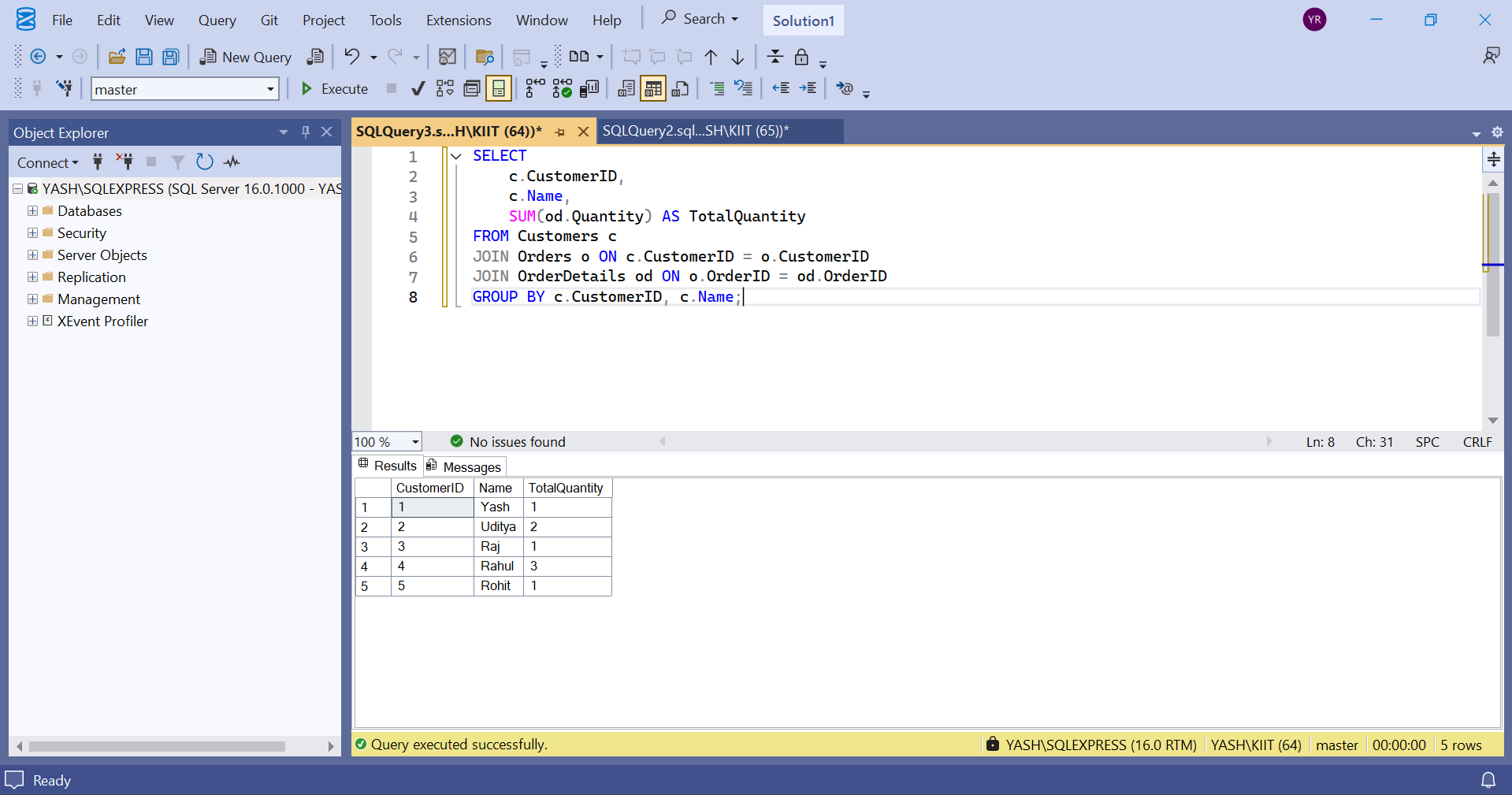
FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

GROUP BY c.CustomerID, c.Name;

**Output :**



**Exercise 3: Creating a Composite Index**

SELECT

c.CustomerID,

c.Name,

SUM(od.Quantity \* p.Price) AS TotalSales

FROM Customers c

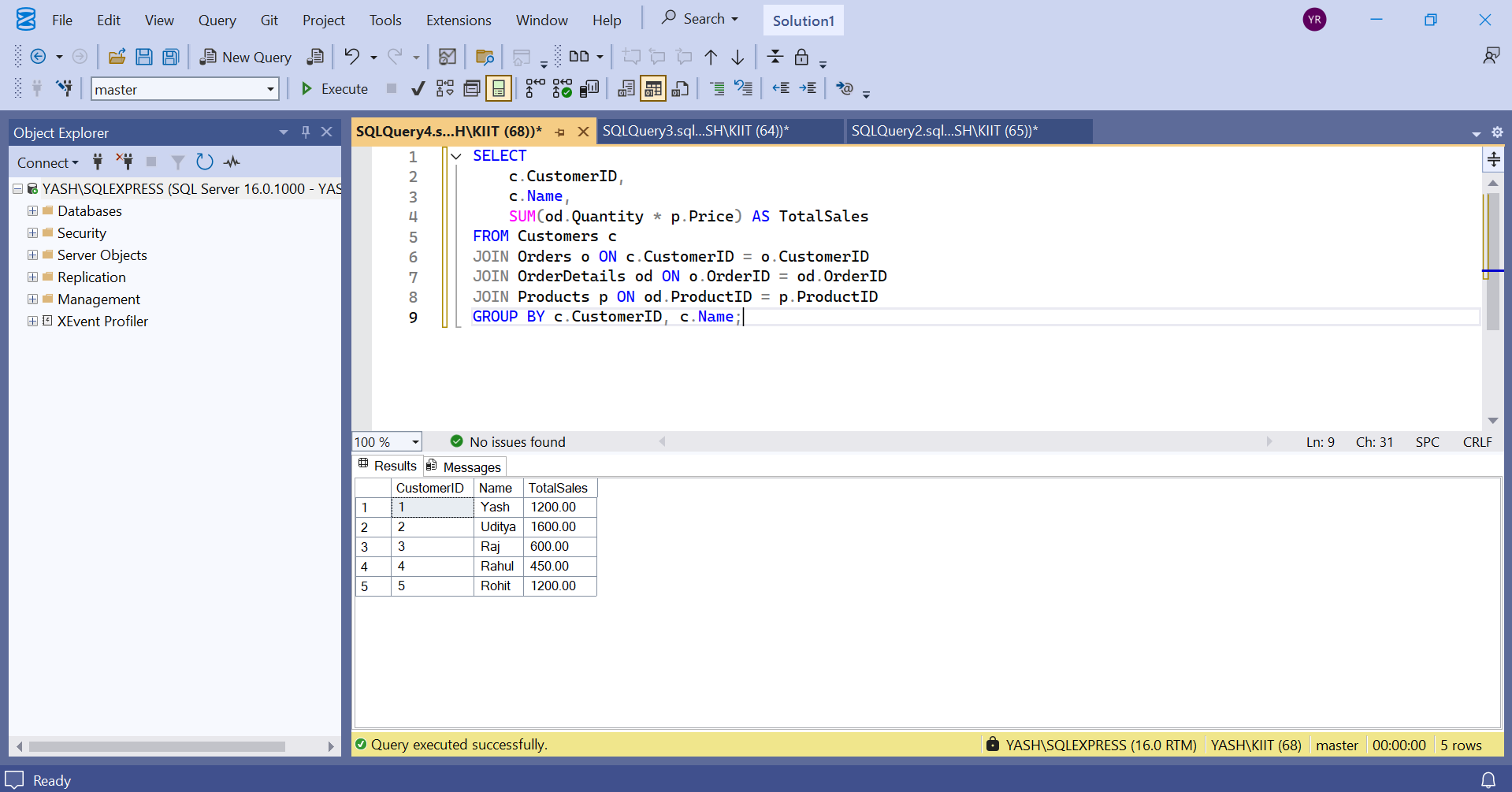
JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY c.CustomerID, c.Name;

Output :



**Exercise 3: SQL EXERCISE - Views.SQL :**

Question 1: Create a Simple View:

DROP VIEW IF EXISTS vw\_EmployeeBasicInfo;

GO

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT,

Salary DECIMAL(10, 2),

JoinDate DATE,

FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)

);

GO

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'Engineering'),

(2, 'Marketing'),

(3, 'HR');

GO

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

(101, 'Aditya', 'Raj', 1, 60000.00, '2022-01-10'),

(102, 'Uditya', 'Kumar', 2, 55000.00, '2022-05-18'),

(103, 'Rahul', 'Mehta', 3, 50000.00, '2023-03-12');

GO

CREATE VIEW vw\_EmployeeBasicInfo AS

SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

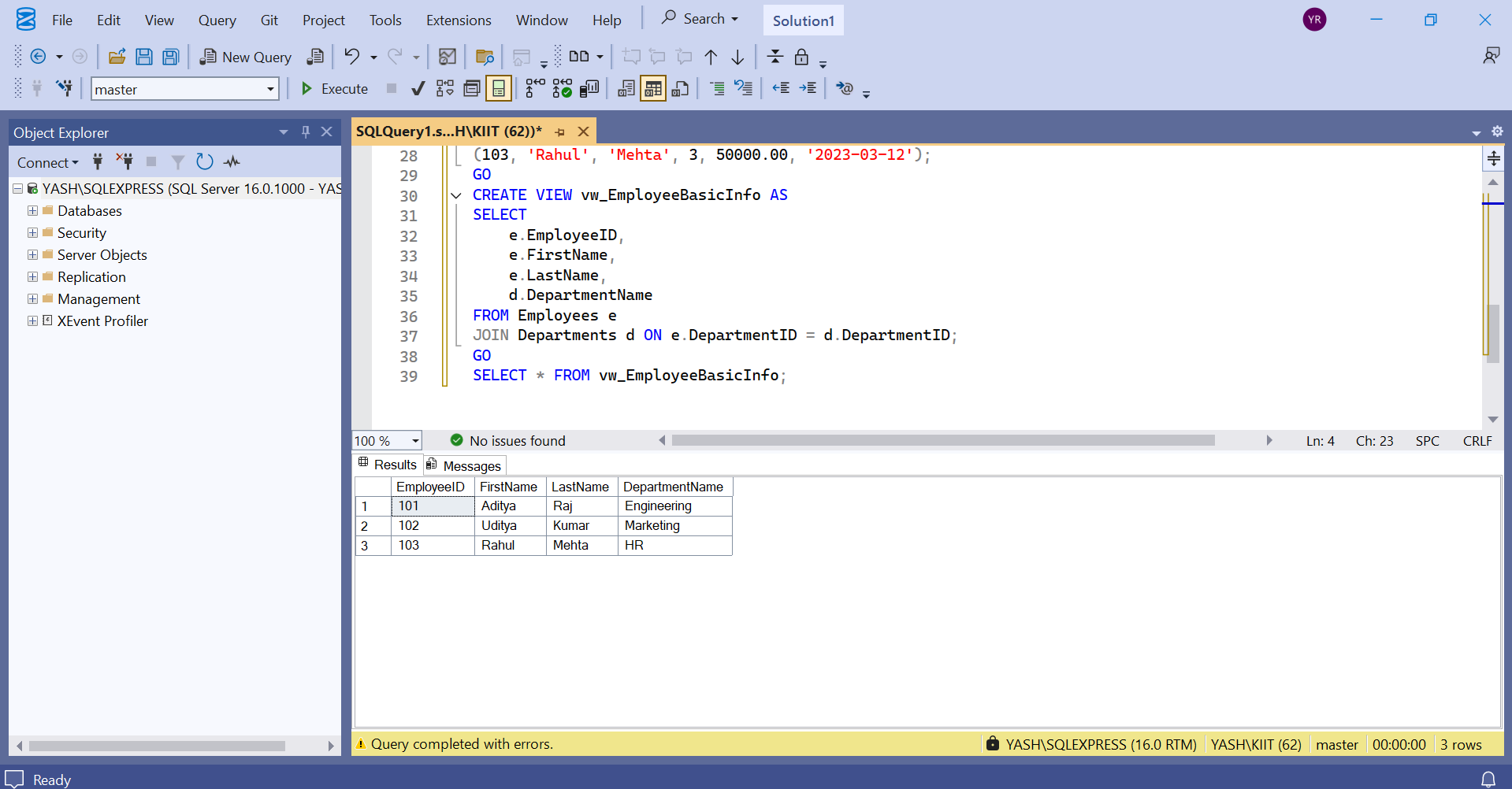
d.DepartmentName

FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID;

GO

SELECT \* FROM vw\_EmployeeBasicInfo;

Output :

**Question2: Add Computed Column - Full Name**

CREATE VIEW vw\_EmployeeFullName AS

SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

e.FirstName + ' ' + e.LastName AS FullName,

d.DepartmentName

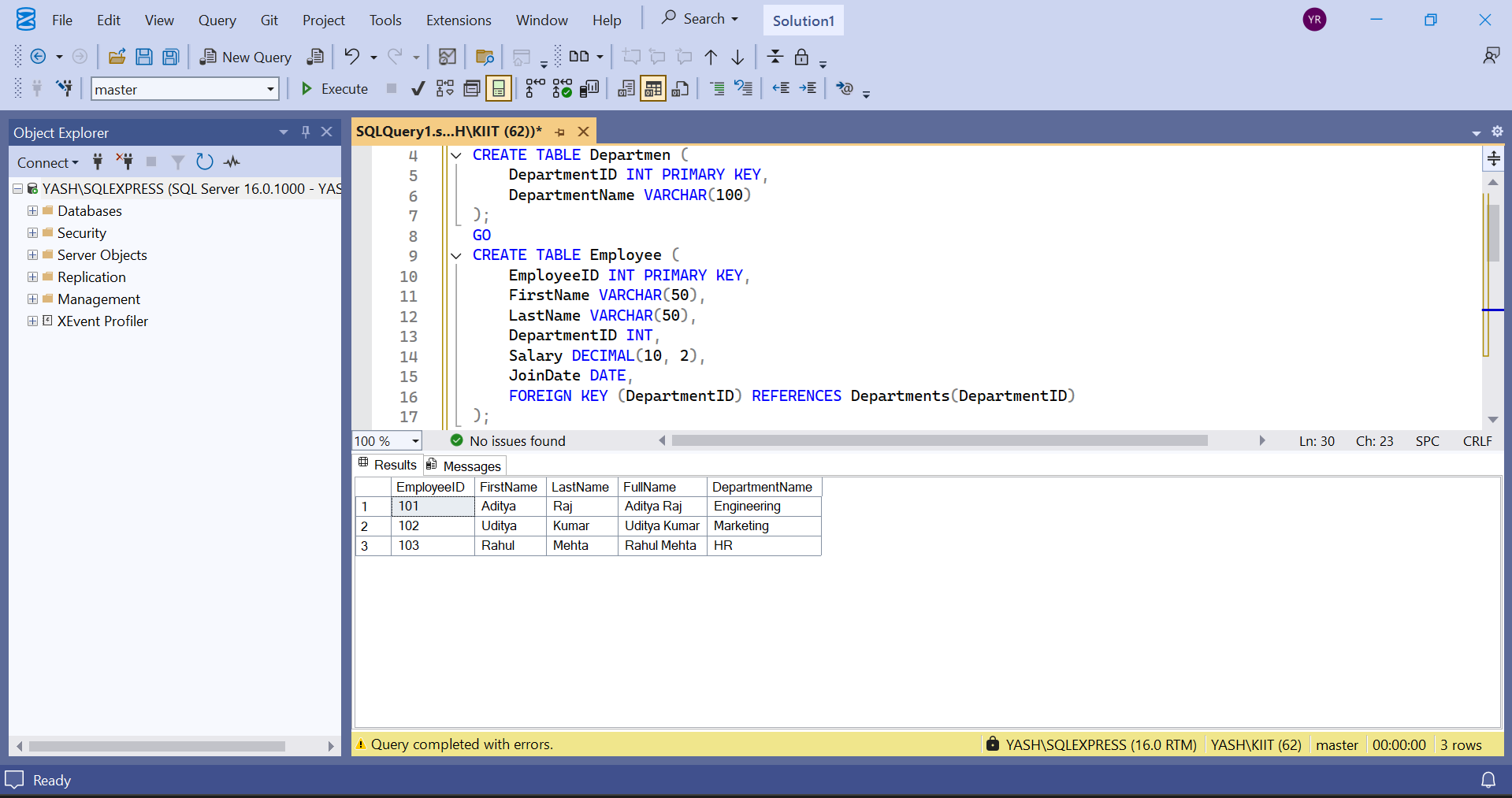
FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID;

GO

SELECT \* FROM vw\_EmployeeFullName;

Output :



**Question 3: Add Computed Column - Annual Salary:**

CREATE VIEW vw\_EmployeeAnnualSalary AS

SELECT

e.EmployeeID,

e.FirstName,

e.LastName,

d.DepartmentName,

e.Salary,

e.Salary \* 12 AS AnnualSalary

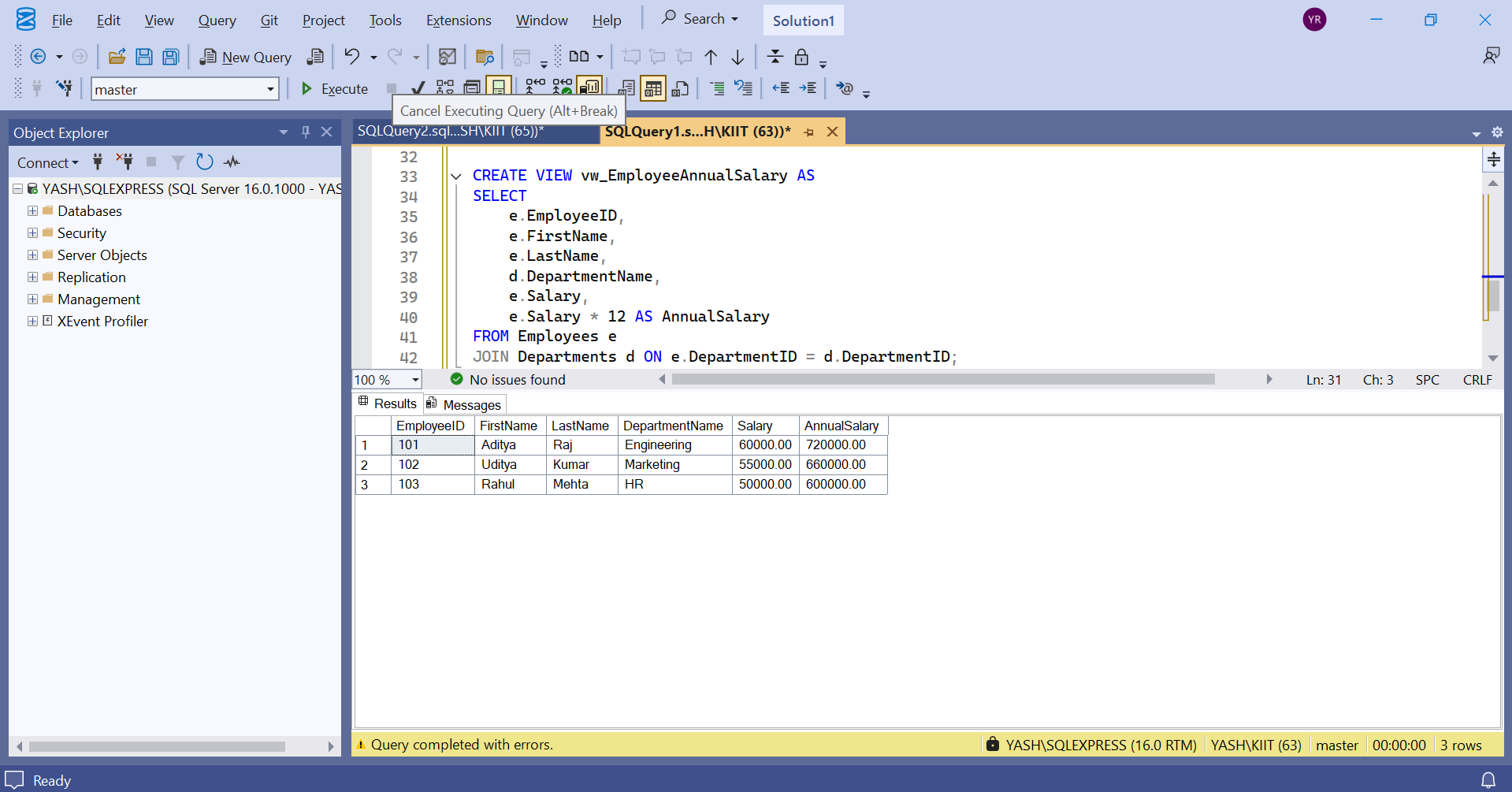
FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID;

GO

SELECT \* FROM vw\_EmployeeAnnualSalary;

Output:



Exercise 4: Add Multiple Computed Columns

CREATE VIEW vw\_EmployeeReport AS

SELECT

e.EmployeeID,

e.FirstName + ' ' + e.LastName AS FullName,

d.DepartmentName,

e.Salary \* 12 AS AnnualSalary,

(e.Salary \* 12) \* 0.10 AS Bonus

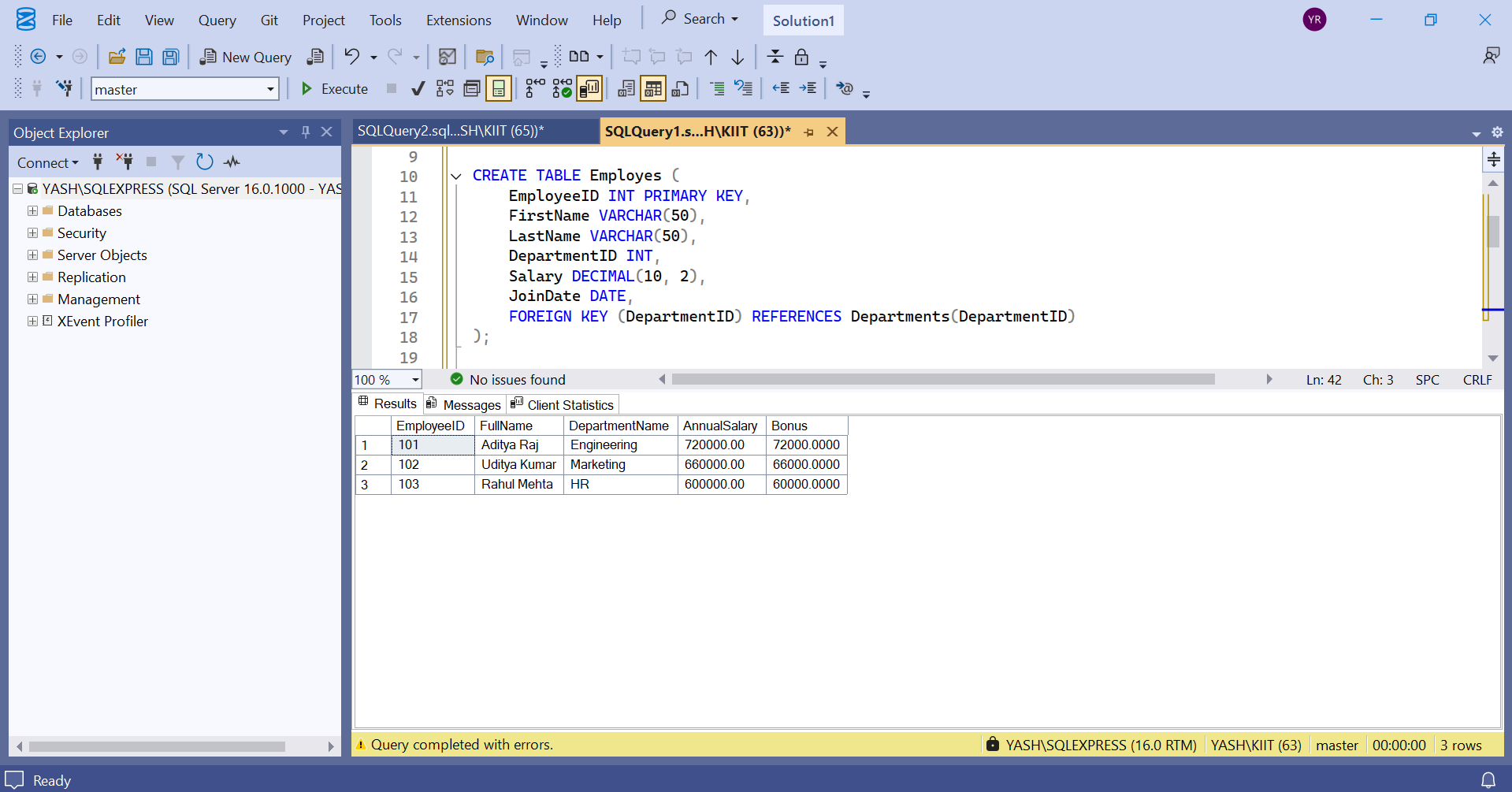
FROM Employees e

JOIN Departments d ON e.DepartmentID = d.DepartmentID;

GO

SELECT \* FROM vw\_EmployeeReport;

Output



**Exercise 4: SQL EXERCISE - Stored Procedure.SQL :**

Question 1: Create a Stored Procedure

1. **Define the stored procedure with a parameter for DepartmentID:**

CREATE DATABASE CompanyDB;

GO

USE CompanyDB;

GO

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(50)

);

GO

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

GO

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'Finance'),

(3, 'IT'),

(4, 'Marketing');

GO

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

(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),

(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),

(3, 'Michael', 'Johnson', 3, 7000.00, '2018-07-30'),

(4, 'Emily', 'Davis', 4, 5500.00, '2021-11-05');

GO

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary,

JoinDate

FROM Employees

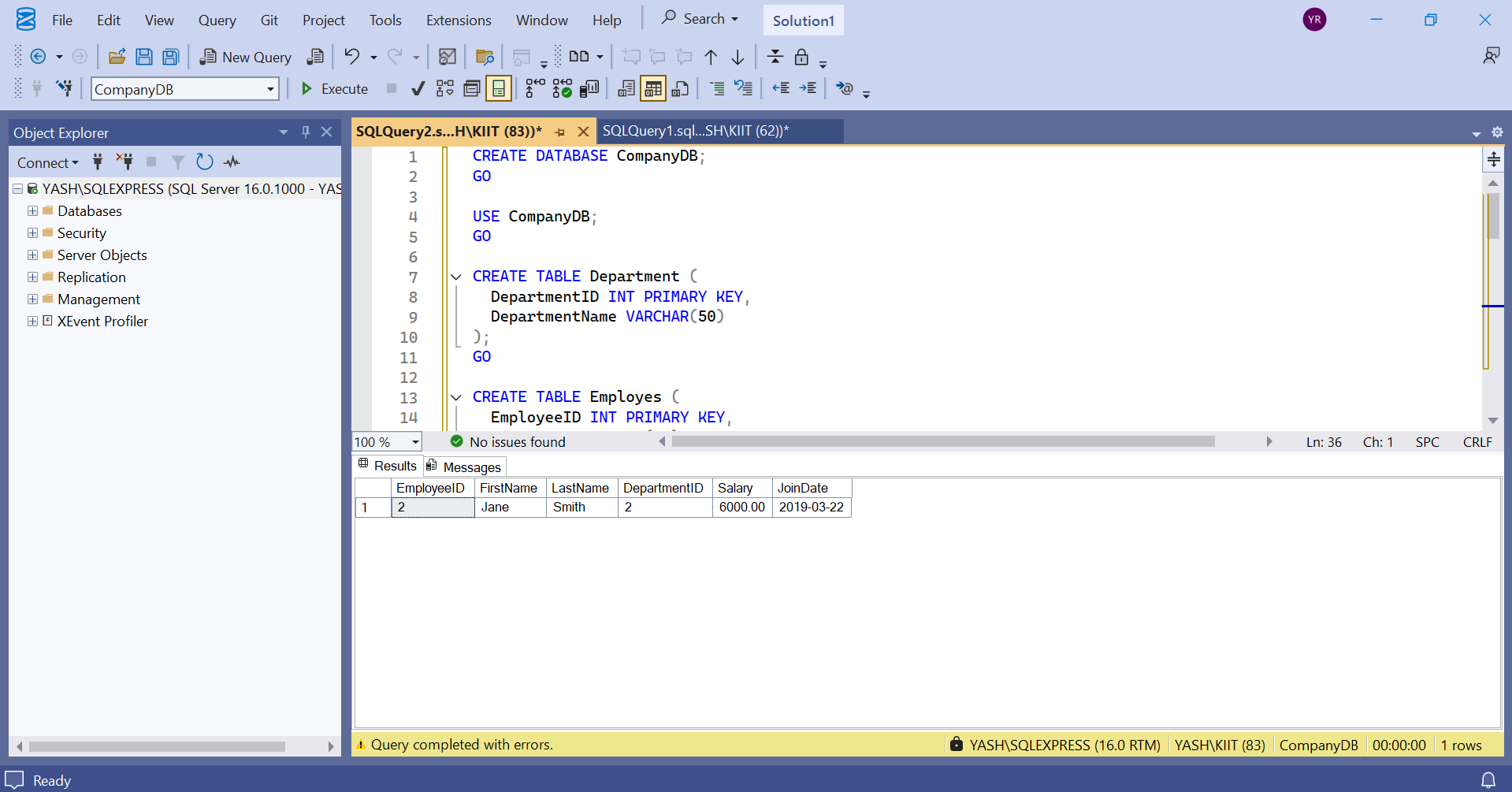
WHERE DepartmentID = @DepartmentID;

END;

GO

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 2;

Output



1. **Write the SQL query to select employee details based on the DepartmentID.**

CREATE TABLE Employees\_temp (

EmployeeID INT IDENTITY(1,1) PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

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

SELECT FirstName, LastName, DepartmentID, Salary, JoinDate FROM Employees;

DROP TABLE Employees;

EXEC sp\_rename 'Employees\_temp', 'Employees';

EXEC sp\_InsertEmployee

@FirstName = 'Alice',

@LastName = 'Brown',

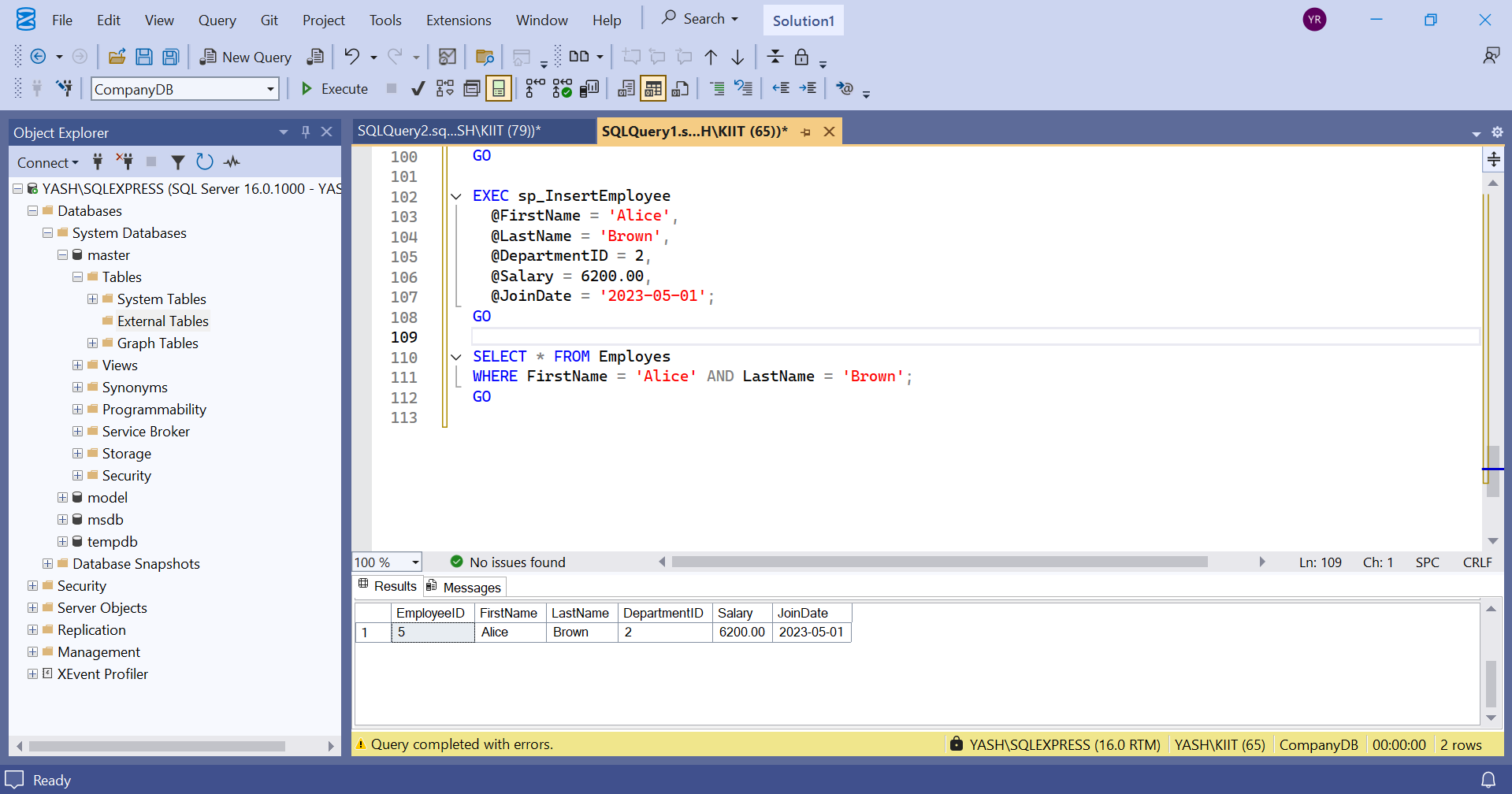
@DepartmentID = 2,

@Salary = 6200.00,

@JoinDate = '2023-05-01';

SELECT \* FROM Employees WHERE FirstName = 'Alice' AND LastName = 'Brown';

Output



1. **Create a stored procedure named `sp\_InsertEmployee` with the following code:**

**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;**

CREATE PROCEDURE sp\_InsertEmployee1

@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

EXEC sp\_InsertEmployee1

@FirstName = 'David',

@LastName = 'Green',

@DepartmentID = 3,

@Salary = 6500.00,

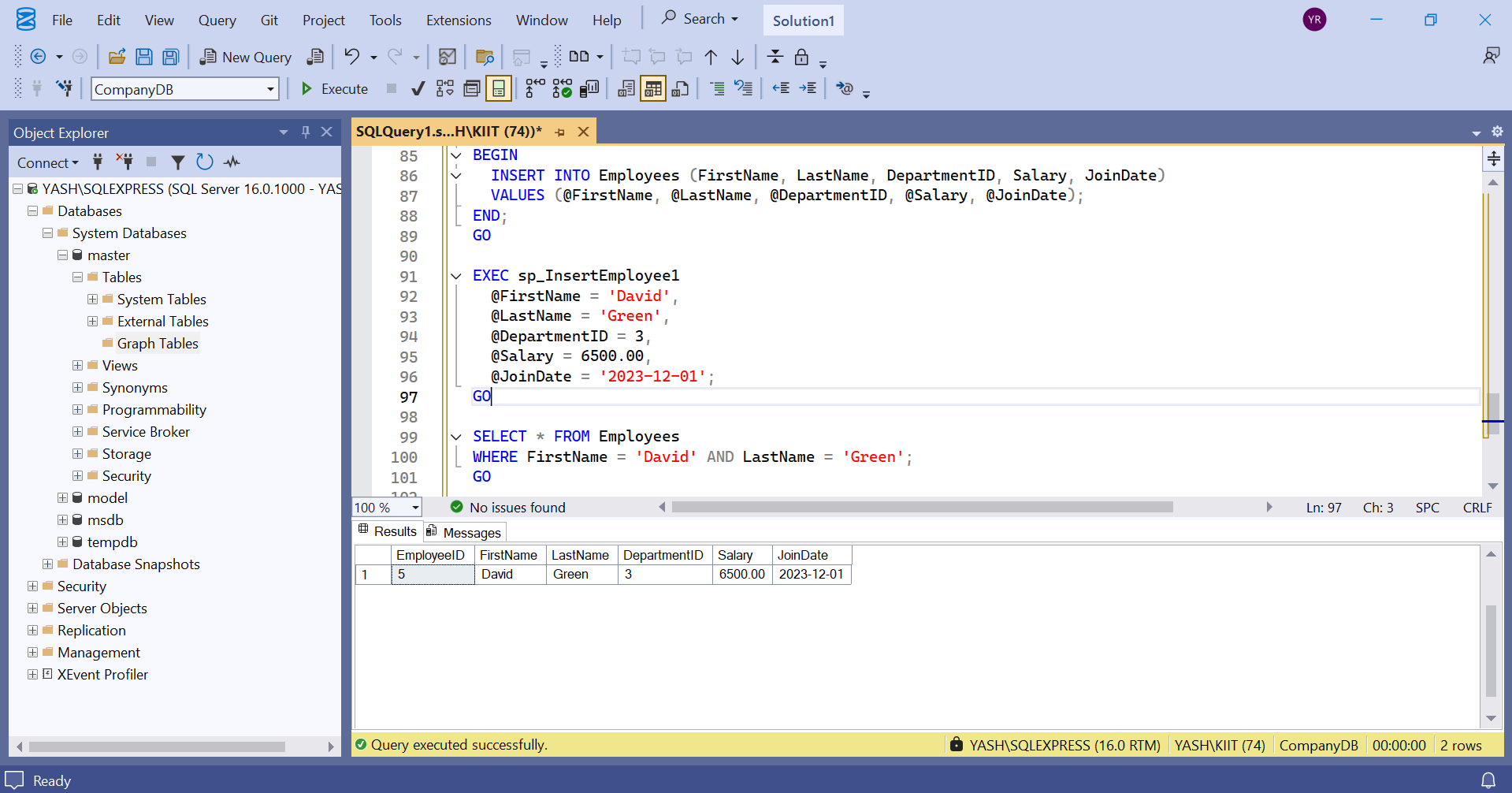
@JoinDate = '2023-12-01';

GO

SELECT \* FROM Employees WHERE FirstName = 'David' AND LastName = 'Green';

GO

Output



**Question 5: Return Data from a Stored Procedure Goal:**

**Create a stored procedure that returns the total number of employees in a department.**

**Write the SQL query to count the number of employees in the specified department.**

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DepartmentID INT

AS

BEGIN

SELECT COUNT(\*) AS TotalEmployees

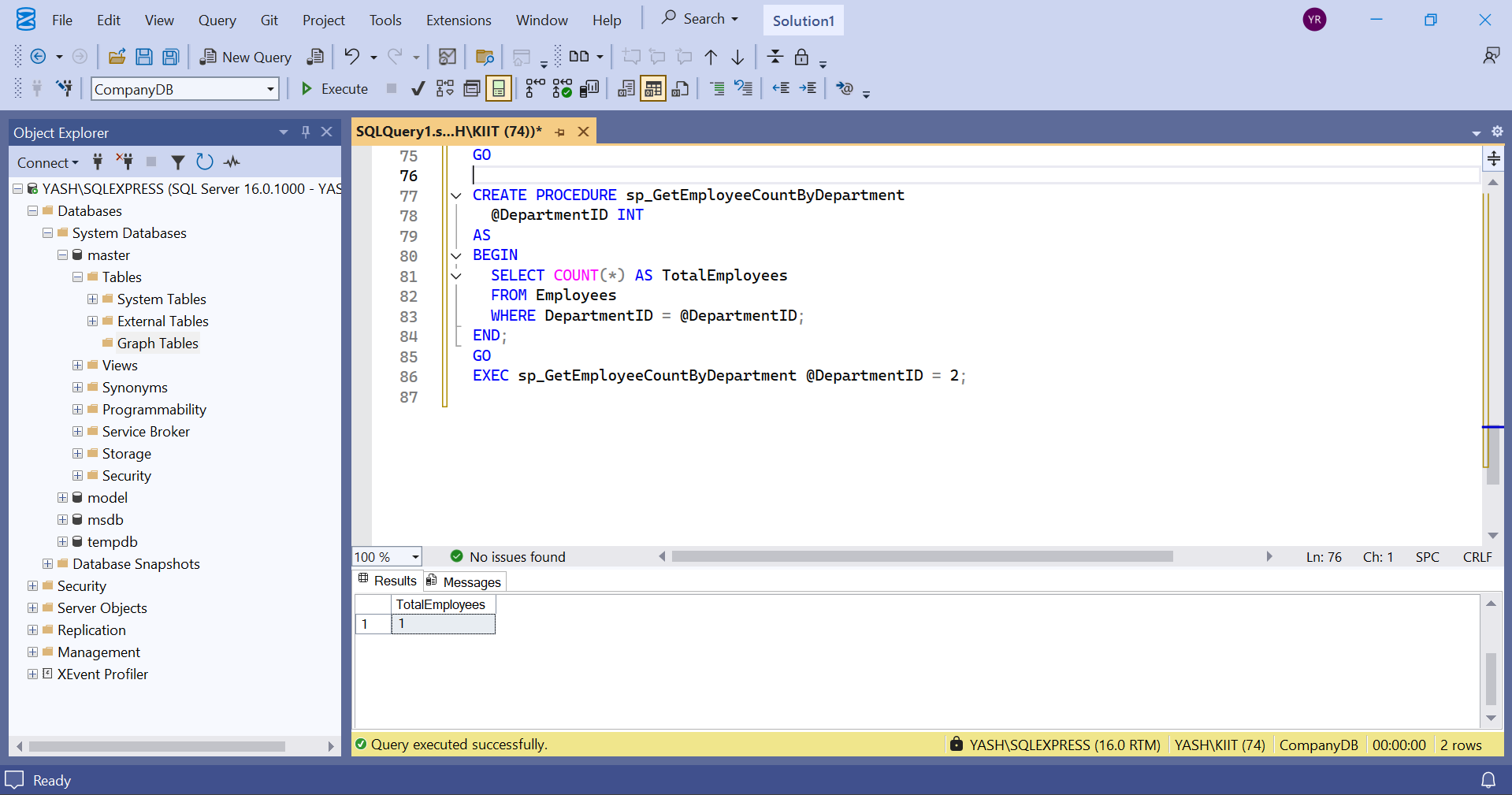
FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 2;

**Output :**



**Exercise 4: SQL EXERCISE - Functions :-**

Question 1: Create a Scalar Function

IF OBJECT\_ID('dbo.fn\_CalculateAnnualSalary', 'FN') IS NOT NULL DROP FUNCTION dbo.fn\_CalculateAnnualSalary;

IF OBJECT\_ID('dbo.fn\_GetEmployeesByDepartment', 'IF') IS NOT NULL DROP FUNCTION dbo.fn\_GetEmployeesByDepartment;

IF OBJECT\_ID('dbo.fn\_CalculateBonus', 'FN') IS NOT NULL DROP FUNCTION dbo.fn\_CalculateBonus;

IF OBJECT\_ID('Employees', 'U') IS NOT NULL DROP TABLE Employees;

IF OBJECT\_ID('Departments', 'U') IS NOT NULL DROP TABLE Departments;

GO

CREATE TABLE Departments (

DepartmentID INT PRIMARY KEY,

DepartmentName VARCHAR(100)

);

GO

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DepartmentID INT FOREIGN KEY REFERENCES Departments(DepartmentID),

Salary DECIMAL(10,2),

JoinDate DATE

);

GO

INSERT INTO Departments (DepartmentID, DepartmentName) VALUES

(1, 'HR'),

(2, 'IT'),

(3, 'Finance');

GO

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

(1, 'John', 'Doe', 1, 5000.00, '2020-01-15'),

(2, 'Jane', 'Smith', 2, 6000.00, '2019-03-22'),

(3, 'Bob', 'Johnson', 3, 5500.00, '2021-07-01');

GO

CREATE FUNCTION fn\_CalculateAnnualSalary (@Salary DECIMAL(10,2))

RETURNS DECIMAL(10,2)

AS

BEGIN

RETURN @Salary \* 12;

END;

GO

SELECT

EmployeeID,

FirstName,

LastName,

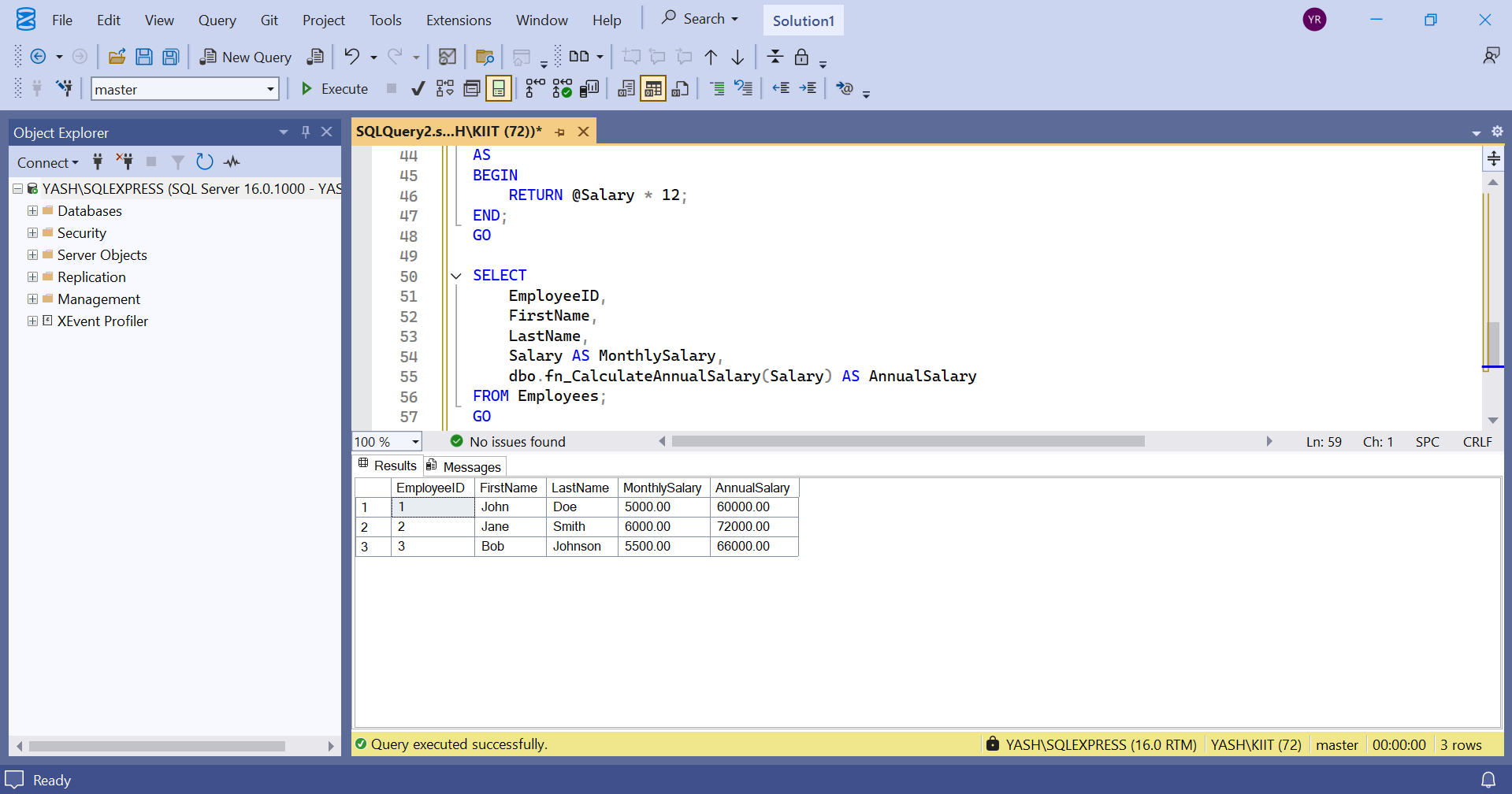
Salary AS MonthlySalary,

dbo.fn\_CalculateAnnualSalary(Salary) AS AnnualSalary

FROM Employees;

GO

OUTPUT:-



1. Create a Table-Valued Function:-

CREATE FUNCTION fn\_GetEmployeesByDepartment (@DeptID INT)

RETURNS TABLE

AS

RETURN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary,

JoinDate

FROM Employees

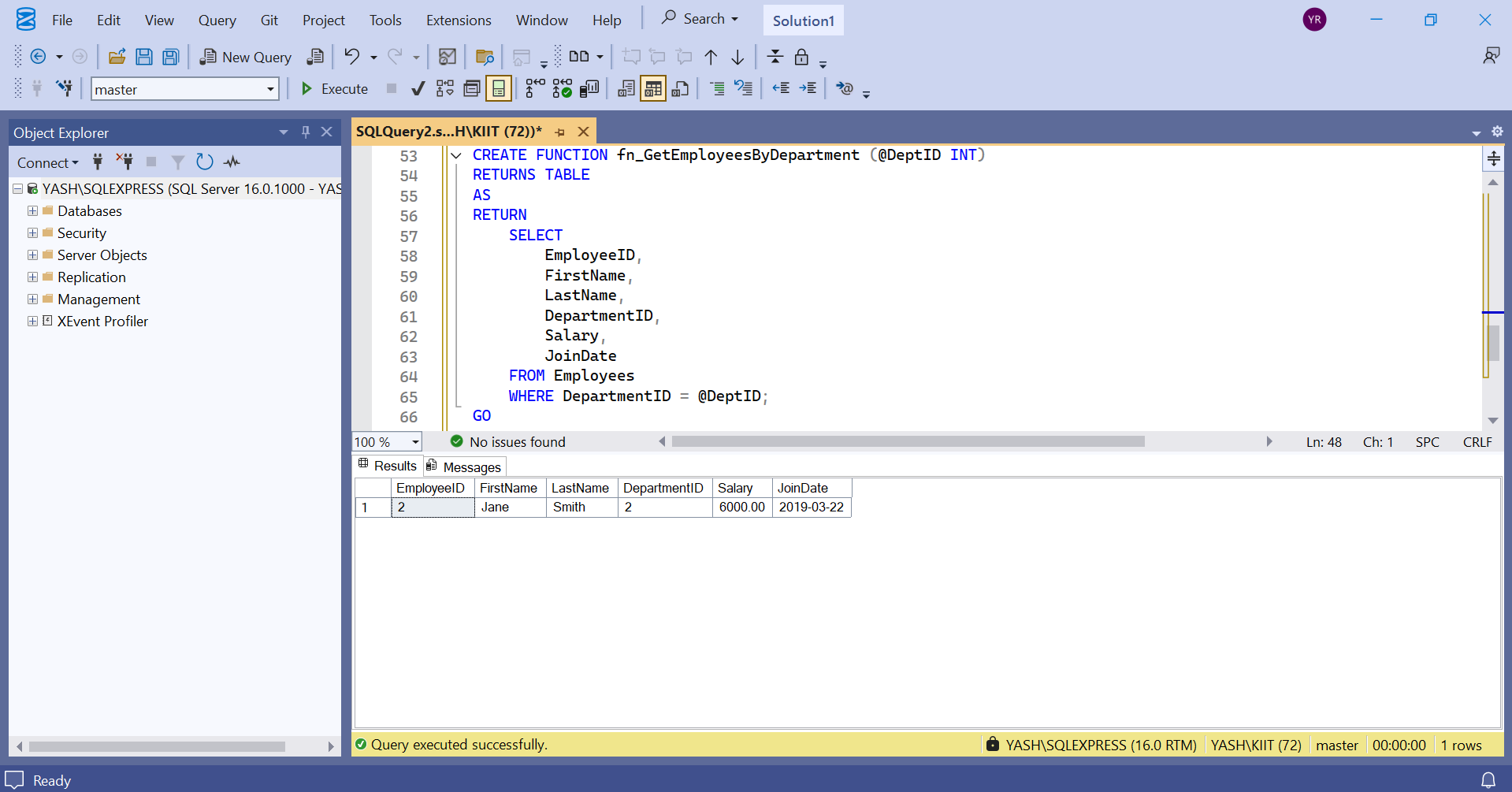
WHERE DepartmentID = @DeptID;

GO

SELECT \* FROM fn\_GetEmployeesByDepartment(2);

GO

OUTPUT:-



1. Create a User-Defined Function :-

CREATE FUNCTION fn\_CalculateBonus (@Salary DECIMAL(10,2))

RETURNS DECIMAL(10,2)

AS

BEGIN

RETURN @Salary \* 0.10;

END;

GO

SELECT

EmployeeID,

FirstName,

LastName,

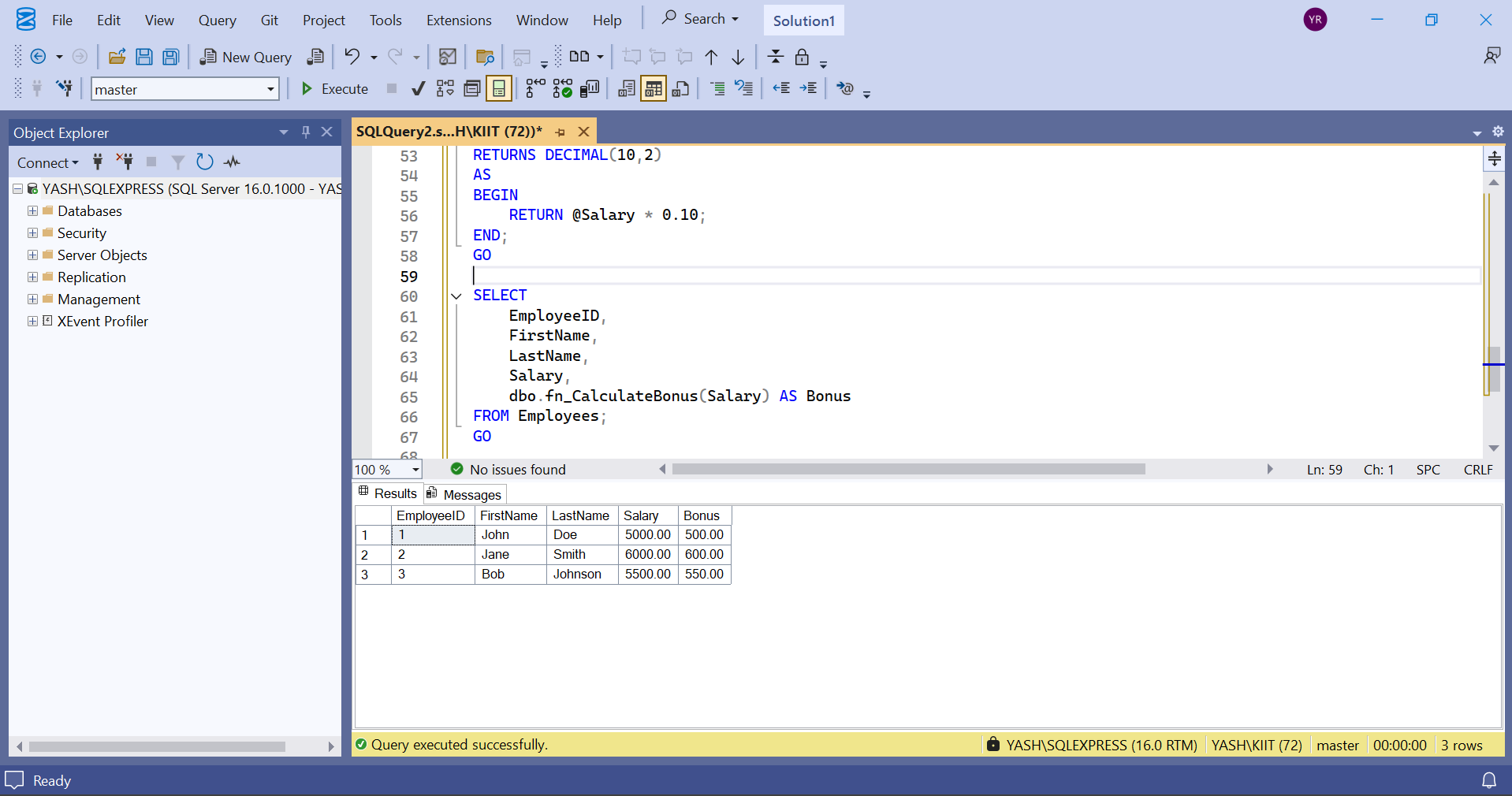
Salary,

dbo.fn\_CalculateBonus(Salary) AS Bonus

FROM Employees;

GO

OUTPUT:-



4: Modify a User-Defined Function:-

CREATE FUNCTION fn\_CalculateBonus (@Salary DECIMAL(10,2))

RETURNS DECIMAL(10,2)

AS

BEGIN

RETURN @Salary \* 0.15;

END;

GO

SELECT

EmployeeID,

FirstName,

LastName,

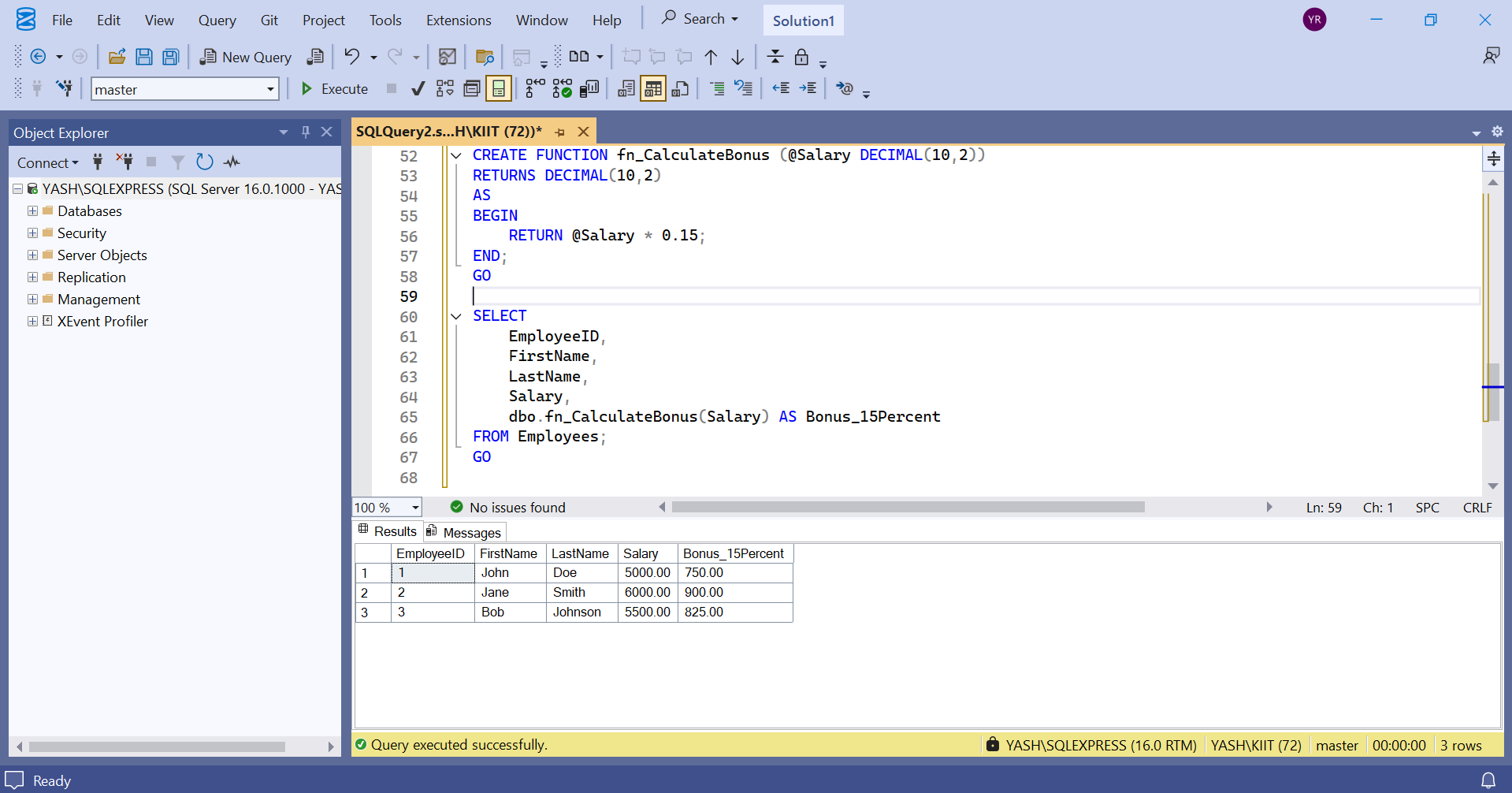
Salary,

dbo.fn\_CalculateBonus(Salary) AS Bonus\_15Percent

FROM Employees;

GO

OUTPUT:-



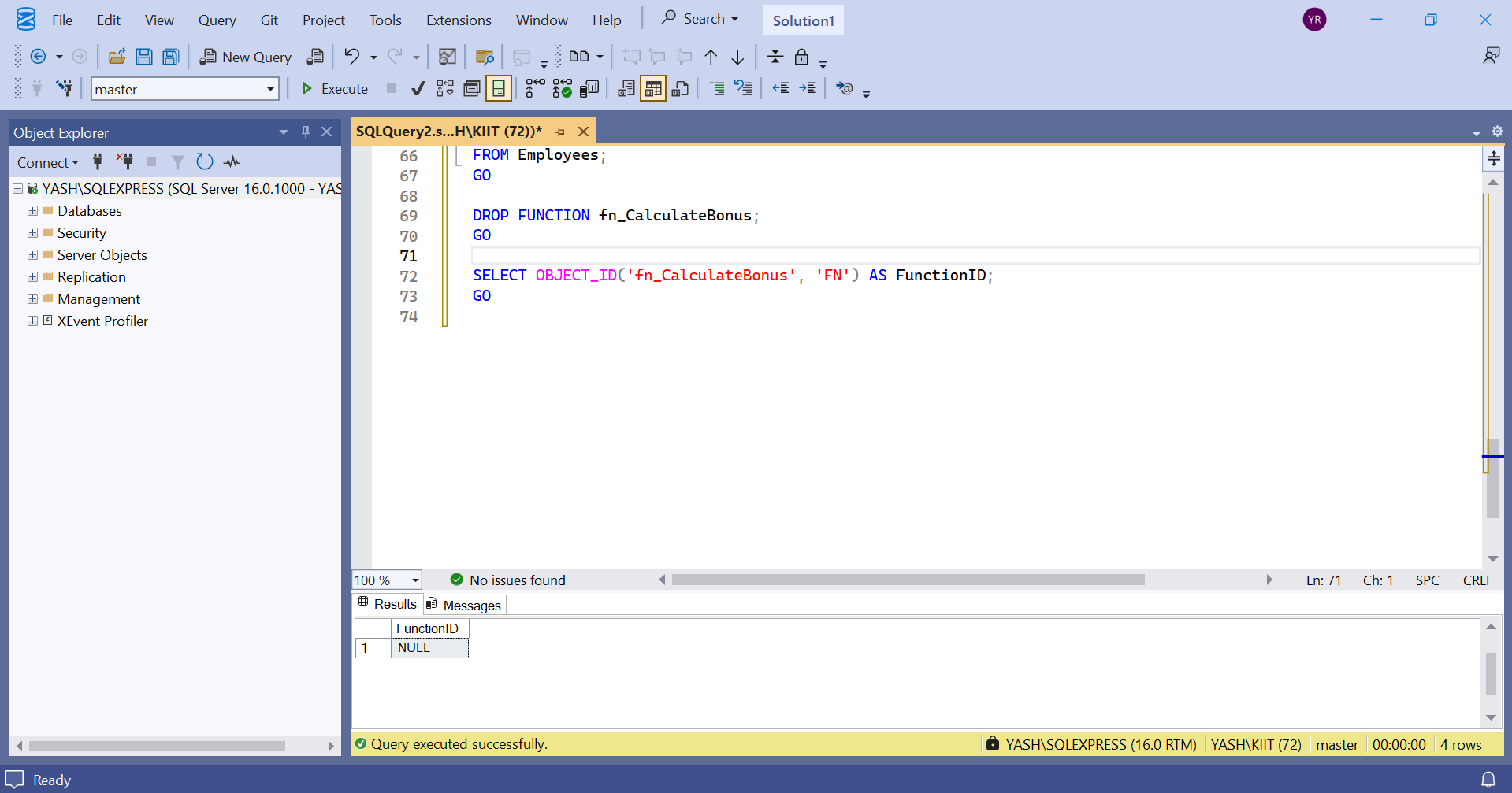
5: Delete a User-Defined Function :-

DROP FUNCTION fn\_CalculateBonus;

GO

SELECT OBJECT\_ID('fn\_CalculateBonus', 'FN') AS FunctionID;

GO



6: Execute a User-Defined Function:-

SELECT

EmployeeID,

FirstName,

LastName,

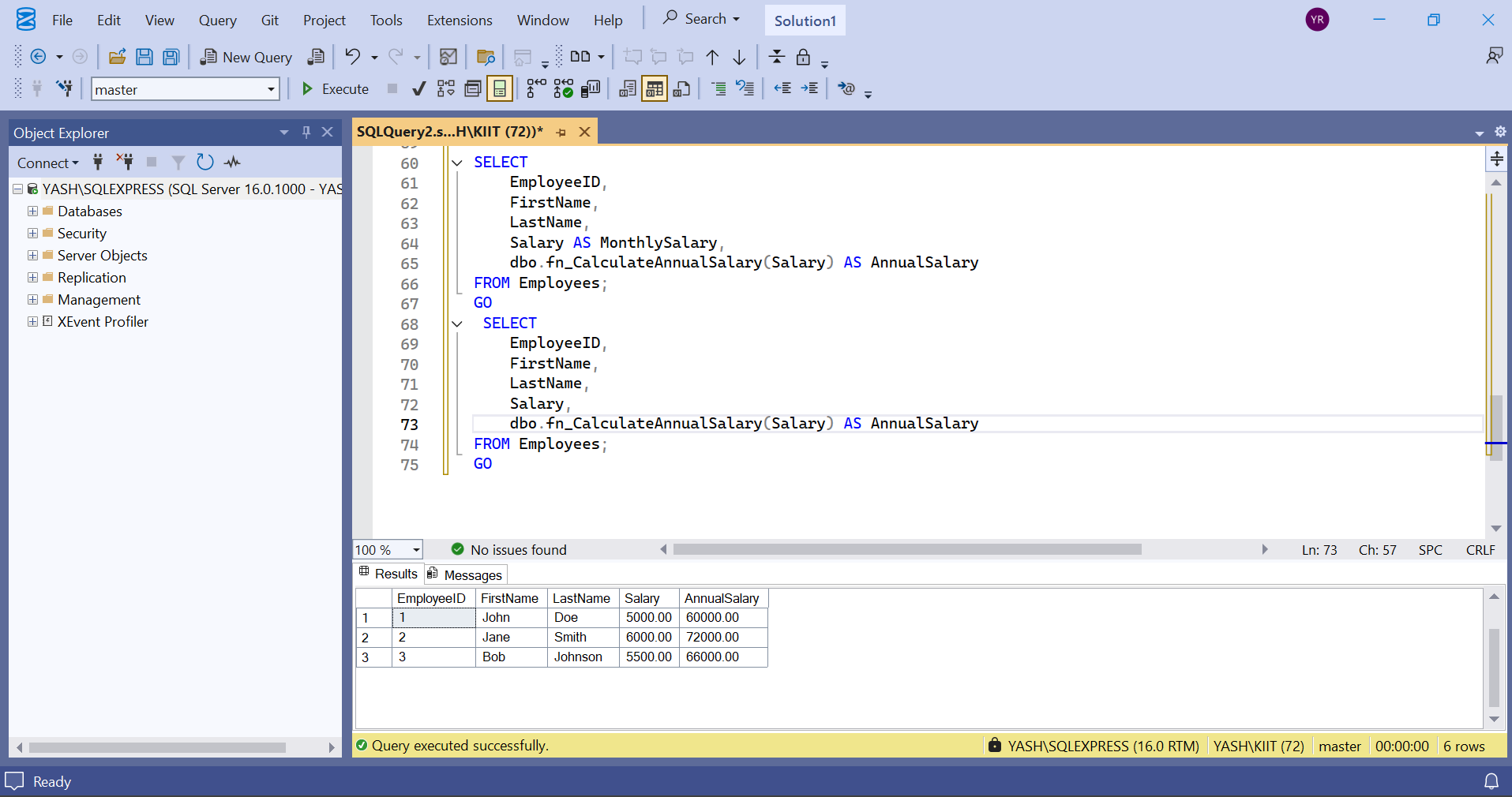
Salary,

dbo.fn\_CalculateAnnualSalary(Salary) AS AnnualSalary

FROM Employees;

GO

OUTPUT:-



7: Return Data from a Scalar Function:-

DECLARE @MonthlySalary DECIMAL(10,2);

SELECT @MonthlySalary = Salary FROM Employees WHERE EmployeeID = 1;

SELECT

dbo.fn\_CalculateAnnualSalary(@MonthlySalary) AS AnnualSalary\_Employee1;

GO

OUTPUT:-

