**Employee Management System SQL Exercises**

**Exercise 1: Create a Stored Procedure**

**Goal: Create a stored procedure to retrieve employee details by department.**

**SOLUTION :  
Execute the Stored Procedure to Insert a Record**

EXEC sp\_InsertEmployee

@FirstName = 'Asha',

@LastName = 'Sharma',

@DepartmentID = 1,

@Salary = 50000.00,

@JoinDate = '2023-06-01';

**Insert More Employees**

EXEC sp\_InsertEmployee

@FirstName = 'Bala',

@LastName = 'Verma',

@DepartmentID = 2,

@Salary = 65000.00,

@JoinDate = '2022-11-15';

EXEC sp\_InsertEmployee

@FirstName = 'Chetan',

@LastName = 'Joshi',

@DepartmentID = 3,

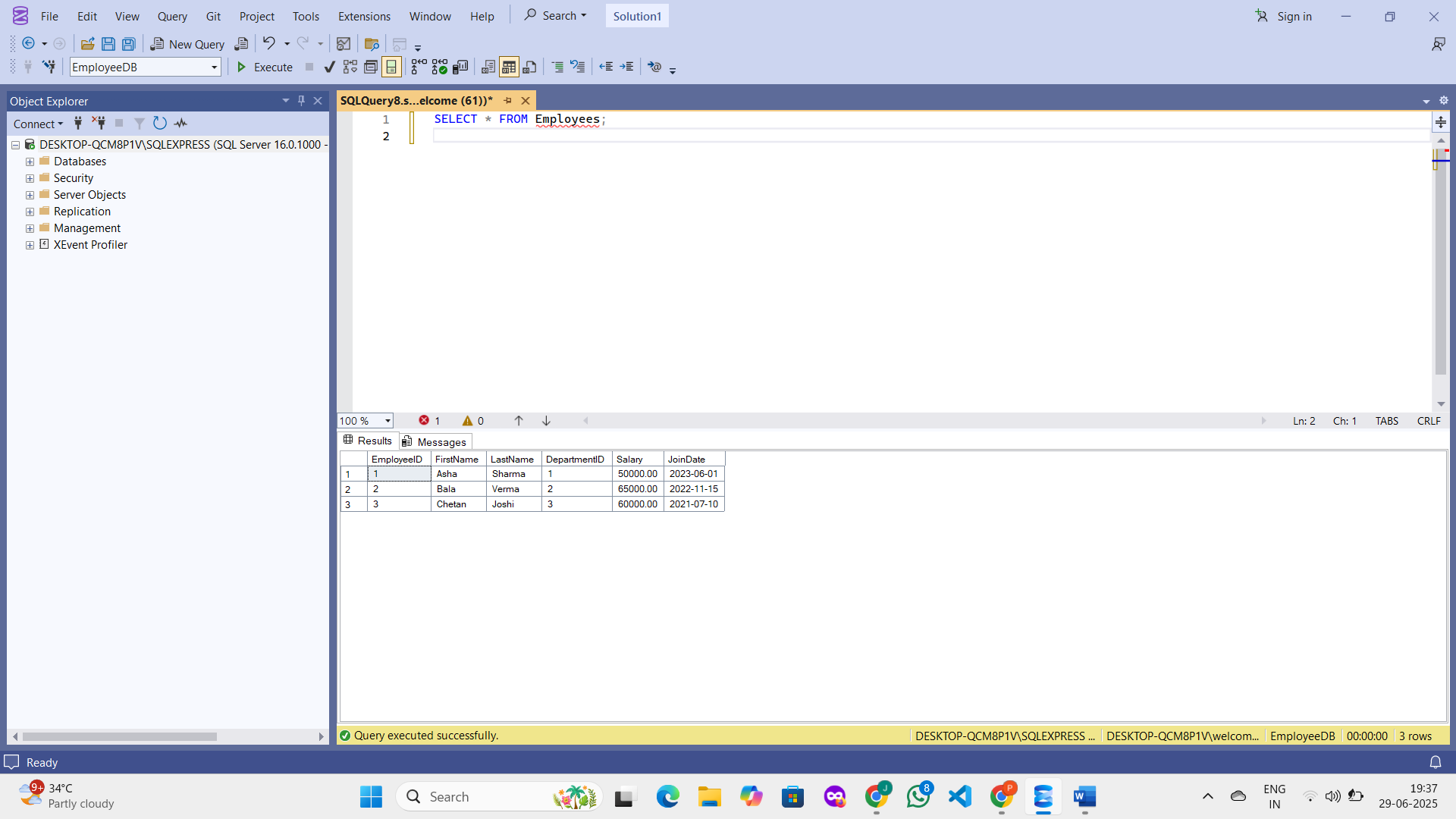
@Salary = 60000.00,

@JoinDate = '2021-07-10';

**Verify the Inserted Data**

SELECT \* FROM Employees;

**OUTPUT :**



**Exercise 2: Modify a Stored Procedure.**

**Goal: Modify the stored procedure to include employee salary in the result.**

**Ensure using the correct database**

USE EmployeeDB;

GO

**CREATE PROCEDURE**

sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID

FROM Employees

WHERE DepartmentID = @DepartmentID;

END; **Modify it to include Salary**

ALTER PROCEDURE sp\_GetEmployeesByDepartment

@DepartmentID INT

AS

BEGIN

SELECT

EmployeeID,

FirstName,

LastName,

DepartmentID,

Salary

FROM Employees

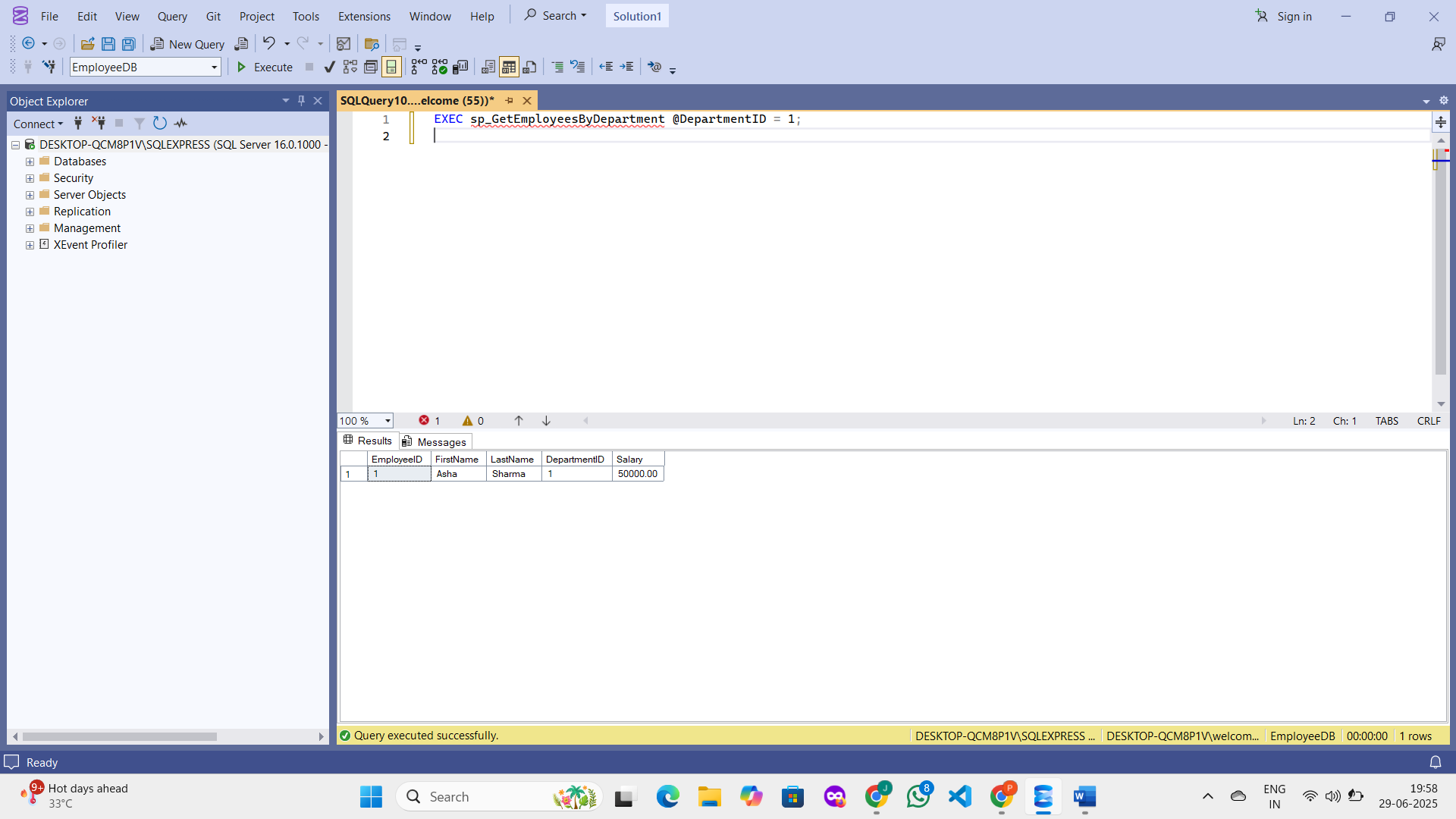
WHERE DepartmentID = @DepartmentID;

END;

**Execute the stored procedure**

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 1;

**OUTPUT :**



**Exercise 3: Delete a Stored Procedure**

**Goal: Delete the stored procedure created in Exercise 1.**

**Use the Correct Database**

USE EmployeeDB;

GO

**Delete the Stored Procedure**

DROP PROCEDURE sp\_InsertEmployee;

GO

**Verify Deletion**

EXEC sp\_InsertEmployee

@FirstName = 'Test',

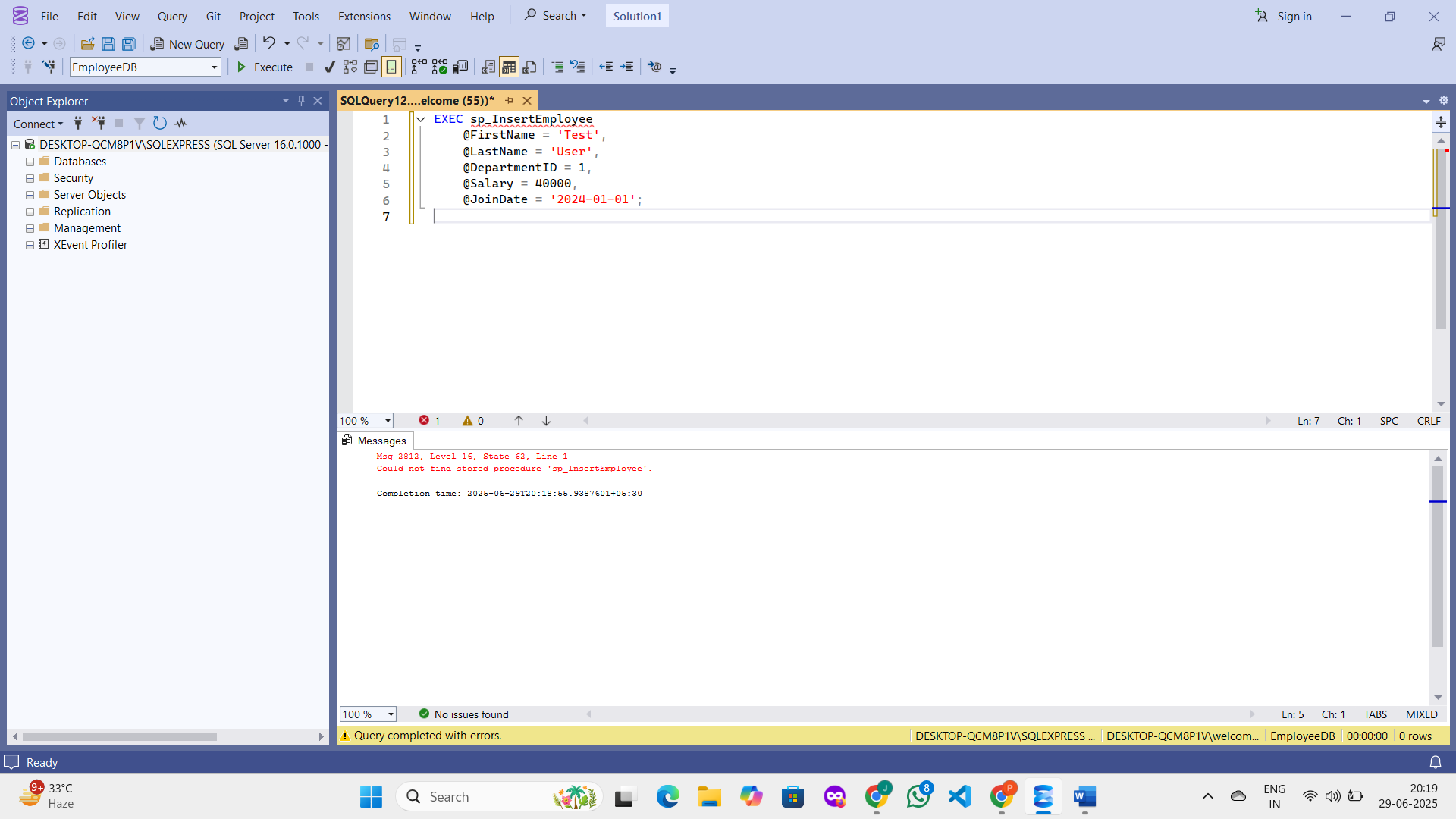
@LastName = 'User',

@DepartmentID = 1,

@Salary = 40000,

@JoinDate = '2024-01-01';

**OUTPUT :**



**Exercise 4: Execute a Stored Procedure Goal: Execute the stored procedure to retrieve employee details for a specific department.**

**SOLUTION :  
Use the Correct Database**

USE EmployeeDB;

GO

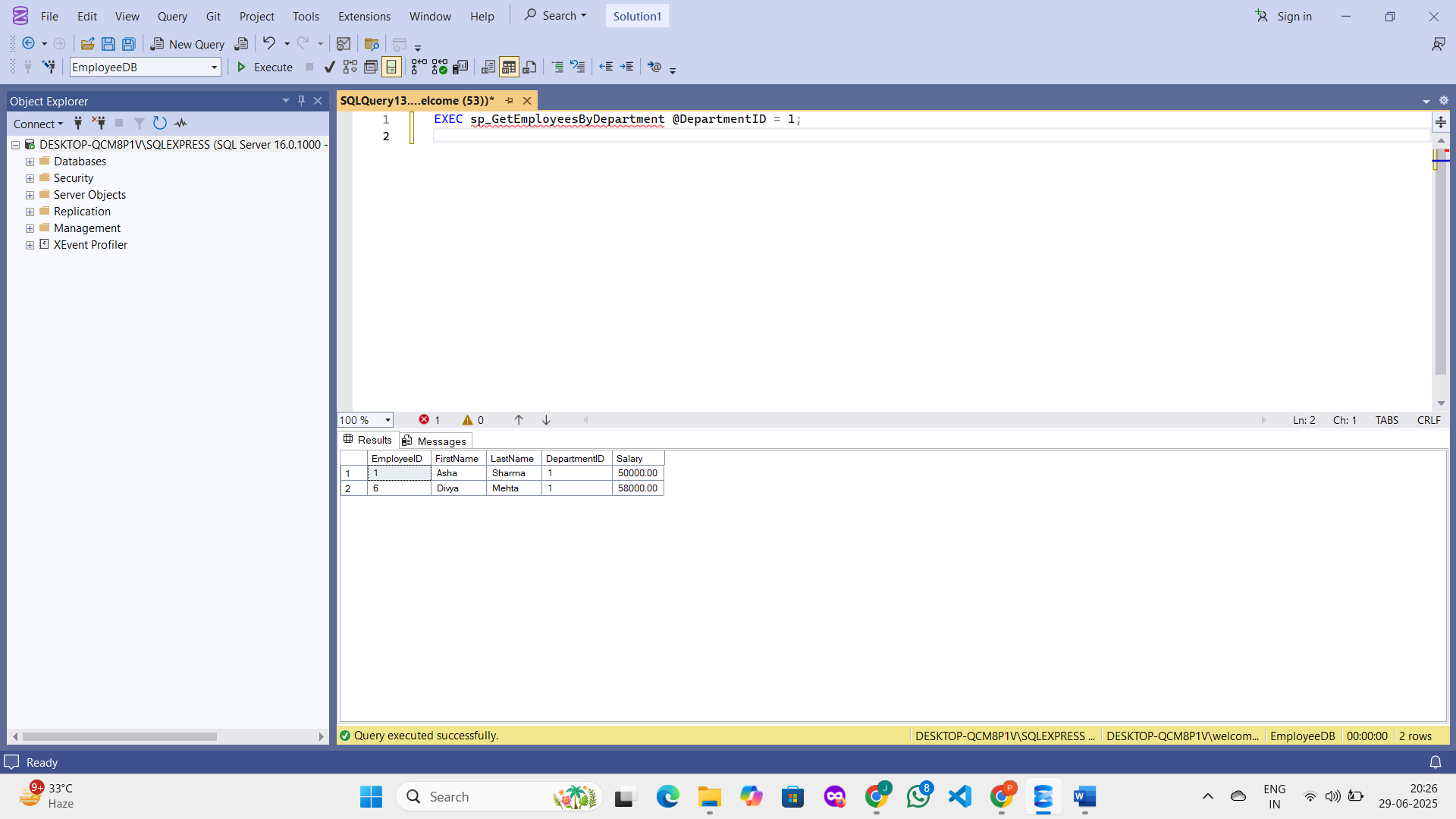
**Execute the Stored Procedure with a DepartmentID Parameter**

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 1;

**Expected Output**

EXEC sp\_GetEmployeesByDepartment @DepartmentID = 1;

**OUTPUT :**



**Exercise 5: Return Data from a Stored Procedure**

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

**SOLUTION :  
Use the Correct Database**

USE EmployeeDB;

GO

**Create the Stored Procedure**

**CREATE PROCEDURE sp\_GetEmployeeCountByDepartment**

@ DepartmentID INT

AS

BEGIN

SELECT

COUNT(\*) AS TotalEmployees

FROM

Employees

WHERE

DepartmentID = @DepartmentID;

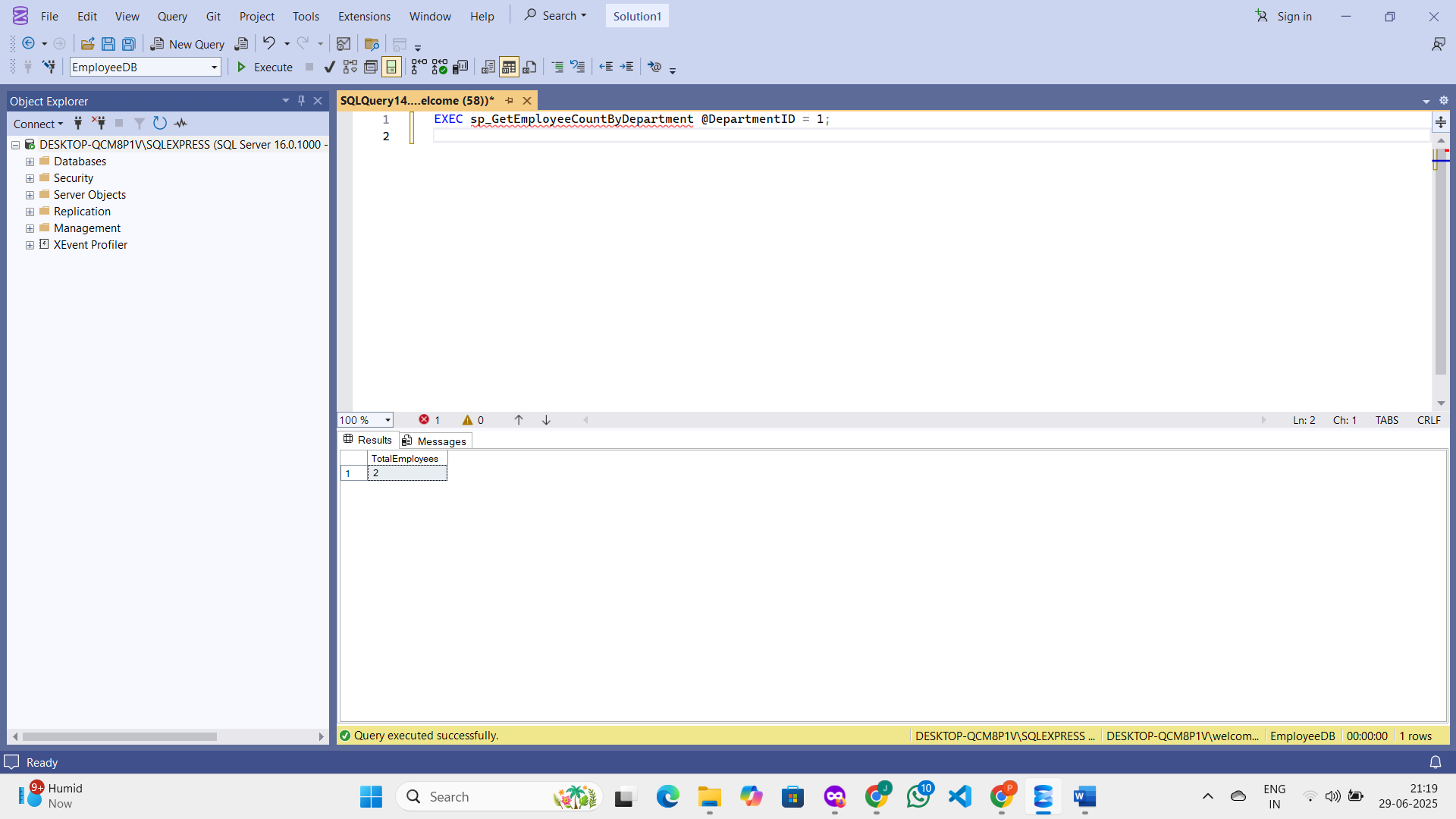
END;

GO

**Execute the Procedure to Get the Count**

EXEC sp\_GetEmployeeCountByDepartment @DepartmentID = 1;

**OUTPUT :**



**Exercise 6: Use Output Parameters in a Stored Procedure.**

**Goal: Create a stored procedure that returns the total salary of employees in a department using an output parameter.**

**SOLUTION :**

**Use the Correct Database**

USE EmployeeDB;

GO

**Create the Stored Procedure with Output Parameter**

CREATE PROCEDURE sp\_GetTotalSalaryByDepartment

@DepartmentID INT,

@TotalSalary DECIMAL(18,2) OUTPUT

AS

BEGIN

SELECT @TotalSalary = SUM(Salary)

FROM Employees

WHERE DepartmentID = @DepartmentID;

END;

GO

**Declare a Variable and Execute the Procedure**

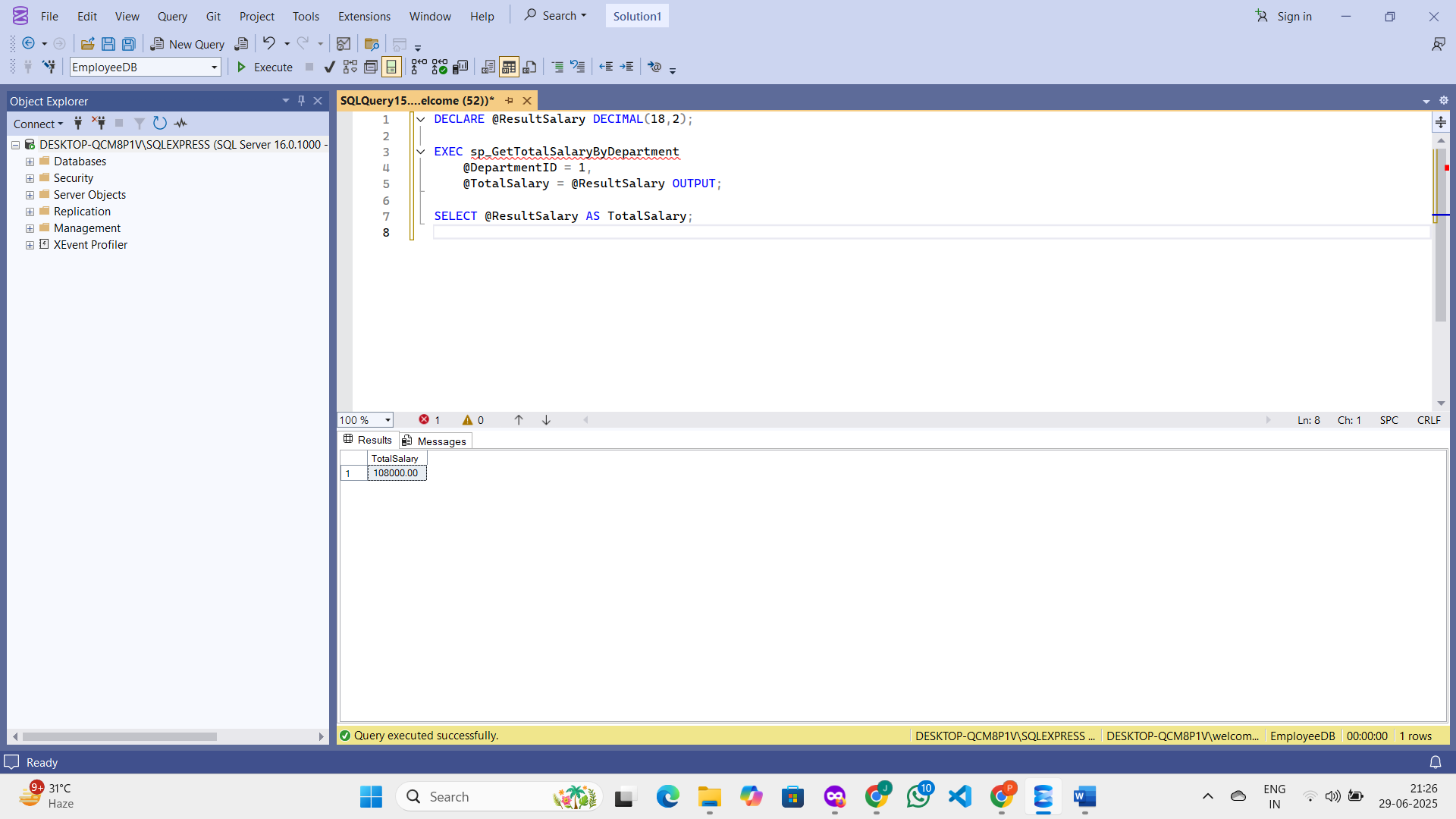
DECLARE @ResultSalary DECIMAL(18,2);

EXEC sp\_GetTotalSalaryByDepartment

@DepartmentID = 1,

@TotalSalary = @ResultSalary OUTPUT;

SELECT @ResultSalary AS TotalSalary;

**OUTPUT :**

**Exercise 7: Create a Stored Procedure with Multiple Parameters.**

**Goal: Create a stored procedure to update employee salary.**

**SOLUTION :  
Use Your Database**

USE EmployeeDB;

GO

**Create the Stored Procedure**

CREATE PROCEDURE sp\_UpdateEmployeeSalary

@EmployeeID INT,

@NewSalary DECIMAL(10, 2)

AS

BEGIN

UPDATE Employees

SET Salary = @NewSalary

WHERE EmployeeID = @EmployeeID;

END;

GO

**Execute the Stored Procedure**

EXEC sp\_UpdateEmployeeSalary 1, 5500.00;

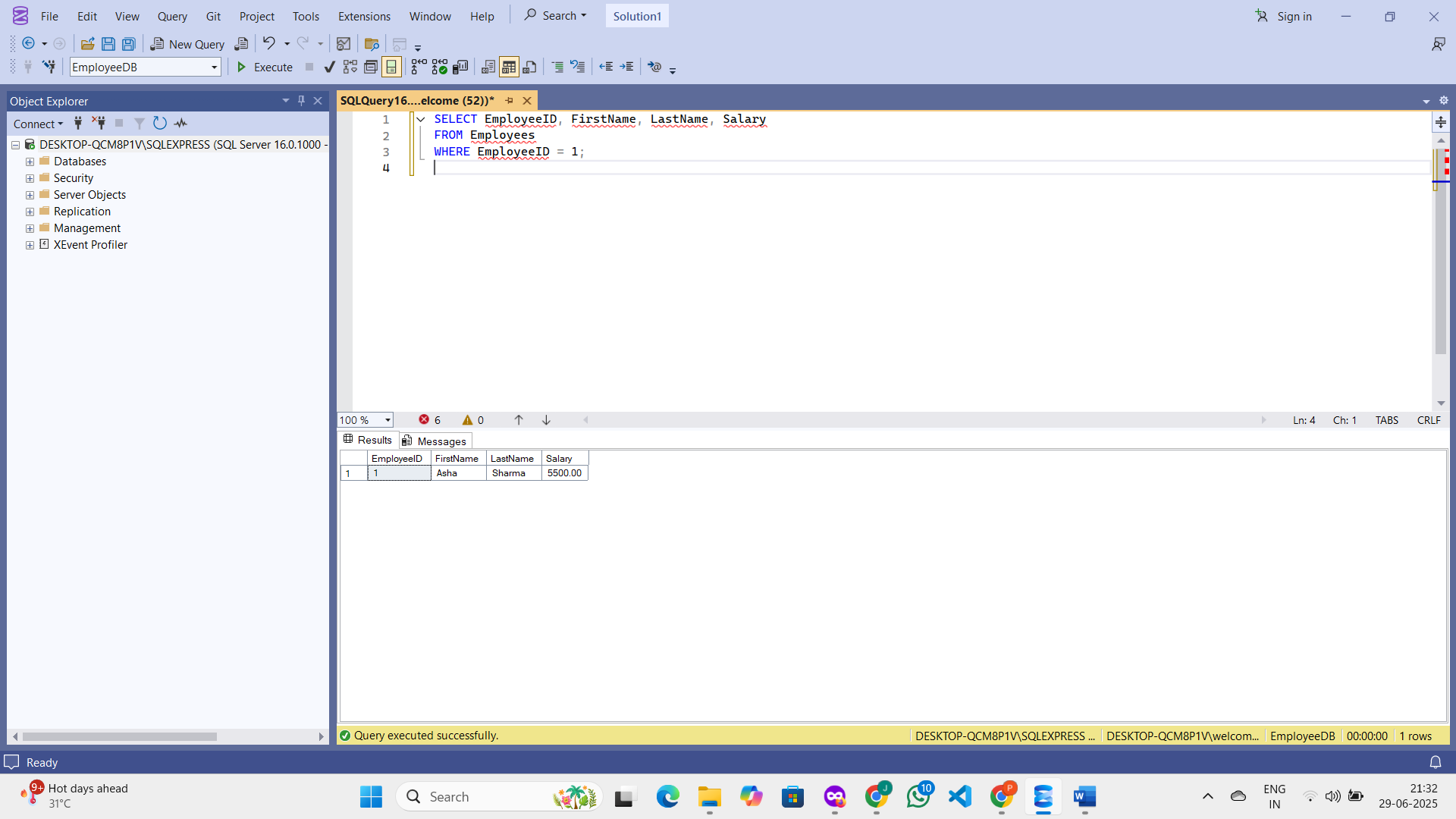
**Verify the Update**

SELECT EmployeeID, FirstName, LastName, Salary

FROM Employees

WHERE EmployeeID = 1;

**Output :**



**Exercise 8: Create a Stored Procedure with Conditional Logic.**

**Goal: Create a stored procedure to give a bonus to employees based on their department.**

**SOLUTION :**

**Use the Correct Database**

USE EmployeeDB;

GO

**Create the Stored Procedure**

CREATE PROCEDURE sp\_GiveBonus

@DepartmentID INT,

@BonusAmount DECIMAL(10, 2)

AS

BEGIN

UPDATE Employees

SET Salary = Salary + @BonusAmount

WHERE DepartmentID = @DepartmentID;

END;

GO

**Execute the Stored Procedure**

EXEC sp\_GiveBonus 1, 500.00;

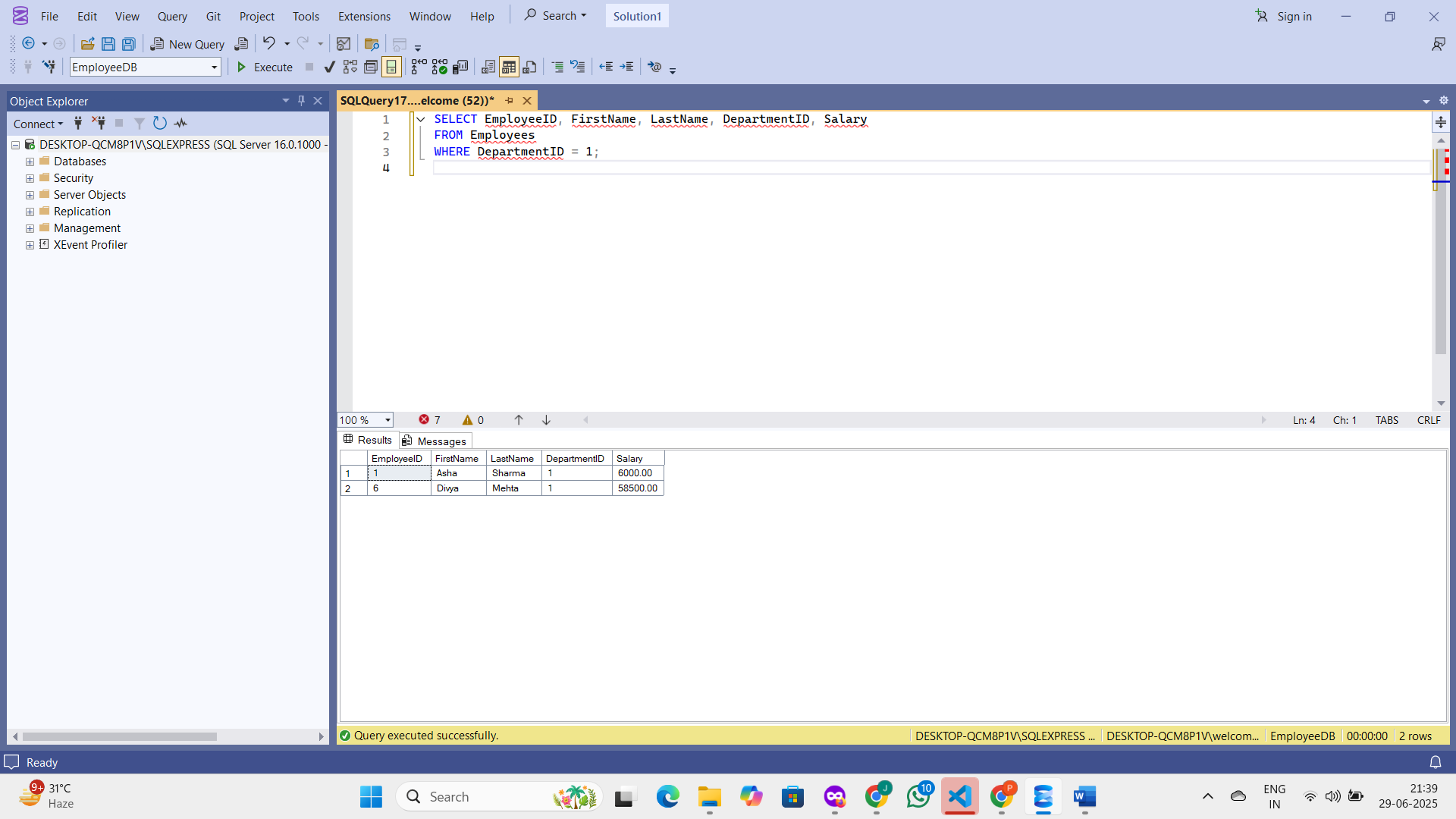
**Verify the Update**

SELECT EmployeeID, FirstName, LastName, DepartmentID, Salary

FROM Employees

WHERE DepartmentID = 1;

**OUTPUT :**



**Exercise 9: Use Transactions in a Stored Procedure.**

**Goal: Create a stored procedure that updates employee salaries and uses a transaction to ensure data integrity.**

**SOLUTION :  
  
Use the Correct Database**

USE EmployeeDB;

GO

**Create the Stored Procedure with Transaction Logic**

**CREATE PROCEDURE sp\_UpdateSalaryWithTransaction**

@EmployeeID INT,

@NewSalary DECIMAL(10, 2)

AS

BEGIN

BEGIN TRY

BEGIN TRANSACTION;

-- Update employee salary

UPDATE Employees

SET Salary = @NewSalary

WHERE EmployeeID = @EmployeeID;

-- Check if the employee exists

IF @@ROWCOUNT = 0

BEGIN

-- No employee was updated → rollback

RAISERROR('Employee not found.', 16, 1);

ROLLBACK TRANSACTION;

RETURN;

END

-- If successful, commit the transaction

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

-- Rollback in case of any error

ROLLBACK TRANSACTION;

-- Show the error

DECLARE @ErrorMessage NVARCHAR(4000);

SET @ErrorMessage = ERROR\_MESSAGE();

RAISERROR(@ErrorMessage, 16, 1);

END CATCH

END;

GO

**Execute the Stored Procedure**

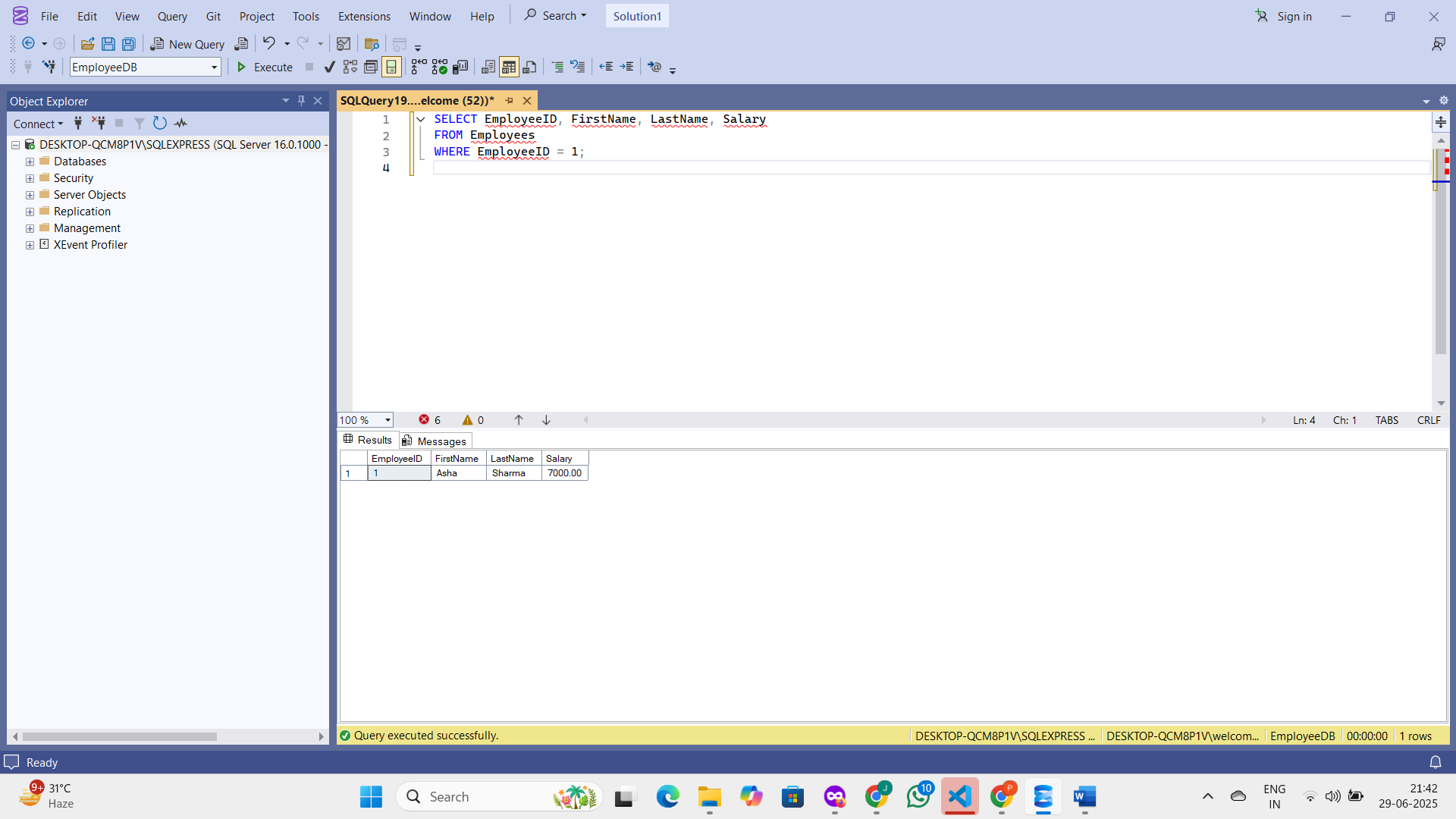
EXEC sp\_UpdateSalaryWithTransaction @EmployeeID = 1, @NewSalary = 7000.00;

**Verify the Update**

SELECT EmployeeID, FirstName, LastName, Salary

FROM Employees

WHERE EmployeeID = 1;

**OUTPUT :**

**Exercise 10: Use Dynamic SQL in a Stored Procedure**

**Goal: Create a stored procedure that uses dynamic SQL to retrieve employee details based**

**on a flexible filter.**

**SOLUTION :**

**Use the Correct Database**

USE EmployeeDB;

GO

**Create the Stored Procedure with Dynamic SQL**

CREATE PROCEDURE sp\_GetEmployeesByDynamicFilter

@FilterColumn NVARCHAR(50),

@FilterValue NVARCHAR(100)

AS

BEGIN

DECLARE @SQL NVARCHAR(MAX);

SET @SQL = 'SELECT EmployeeID, FirstName, LastName, DepartmentID, Salary, JoinDate

FROM Employees

WHERE ' + QUOTENAME(@FilterColumn) + ' = @Value';

EXEC sp\_executesql

@SQL,

N'@Value NVARCHAR(100)',

@Value = @FilterValue;

END;

GO

**Execute the Stored Procedure**

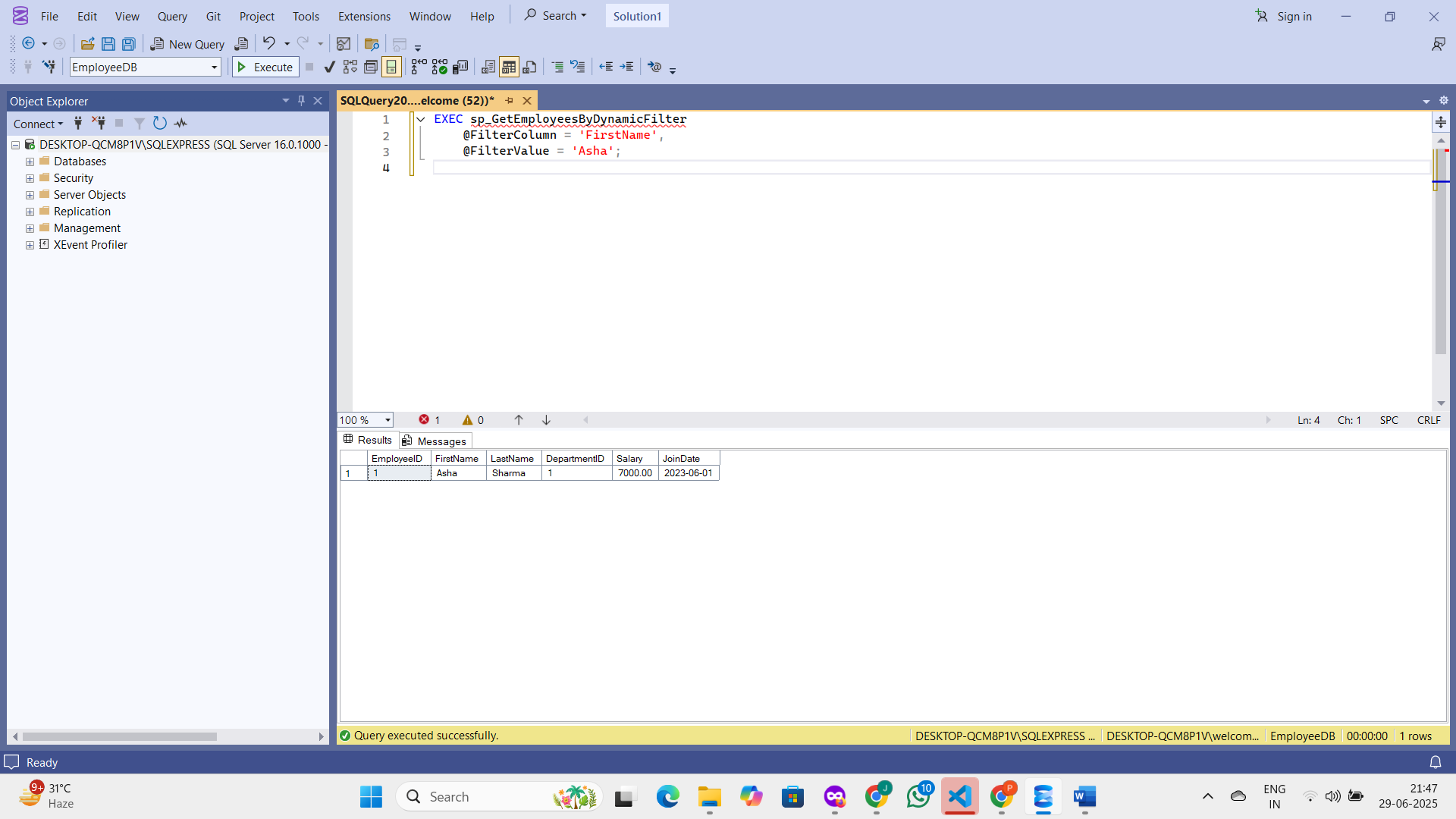
Example 1: Filter by FirstName

EXEC sp\_GetEmployeesByDynamicFilter

@FilterColumn = 'FirstName',

@FilterValue = 'Asha';

**OUTPUT :**

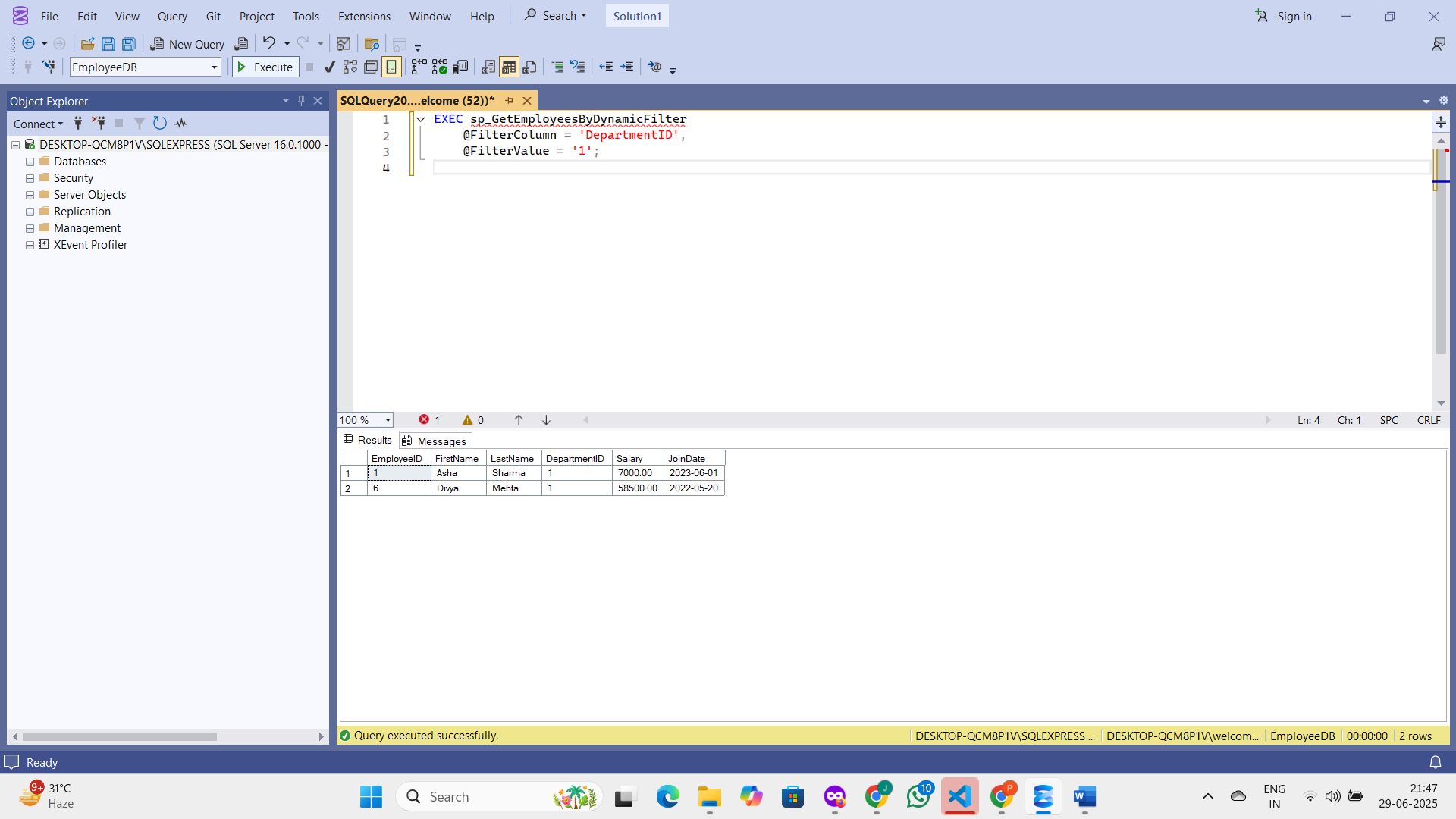


Example 2: Filter by DepartmentID (must convert to string):

EXEC sp\_GetEmployeesByDynamicFilter

@FilterColumn = 'DepartmentID',

@FilterValue = '1';

**OUTPUT :** 

**Exercise 11: Handle Errors in a Stored Procedure.**

**Goal: Create a stored procedure that handles errors and returns a custom error message.**

**SOLUTION :  
Use the Correct Database**

USE EmployeeDB;

GO

**Create the Stored Procedure with Error Handling**

CREATE PROCEDURE sp\_UpdateSalaryWithErrorHandling

@EmployeeID INT,

@NewSalary DECIMAL(10, 2)

AS

BEGIN

BEGIN TRY

-- Attempt to update salary

UPDATE Employees

SET Salary = @NewSalary

WHERE EmployeeID = @EmployeeID;

-- If no rows were updated, raise custom error

IF @@ROWCOUNT = 0

BEGIN

RAISERROR('❌ Error: No employee found with the given EmployeeID.', 16, 1);

END

ELSE

BEGIN

PRINT '✅ Salary updated successfully.';

END

END TRY

BEGIN CATCH

-- Custom error message

DECLARE @ErrMsg NVARCHAR(4000);

SET @ErrMsg = '⚠️ Custom Error: ' + ERROR\_MESSAGE();

RAISERROR(@ErrMsg, 16, 1);

END CATCH

END;

GO

**Execute the Stored Procedure**

**EXEC sp\_UpdateSalaryWithErrorHandling @EmployeeID = 1, @NewSalary = 8000.00;**

**Verify the Update**

SELECT EmployeeID, FirstName, LastName, Salary

FROM Employees

WHERE EmployeeID = 1;

**OUTPUT :**