EXERCISE 1: Ranking and Window Functions

# Main Concepts Used:

• ROW\_NUMBER()

• RANK()

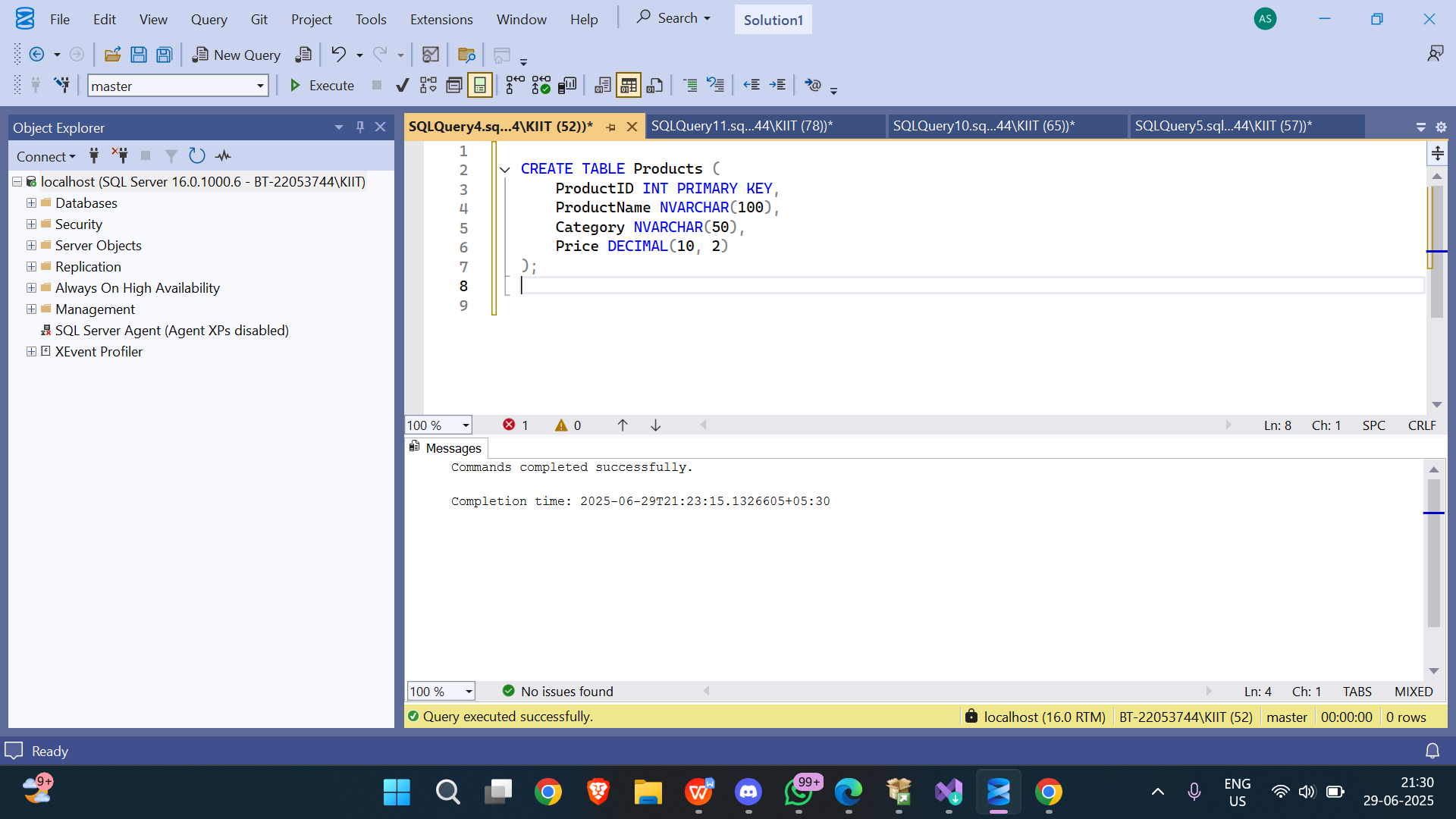
• DENSE\_RANK()

• PARTITION BY

• ORDER BY

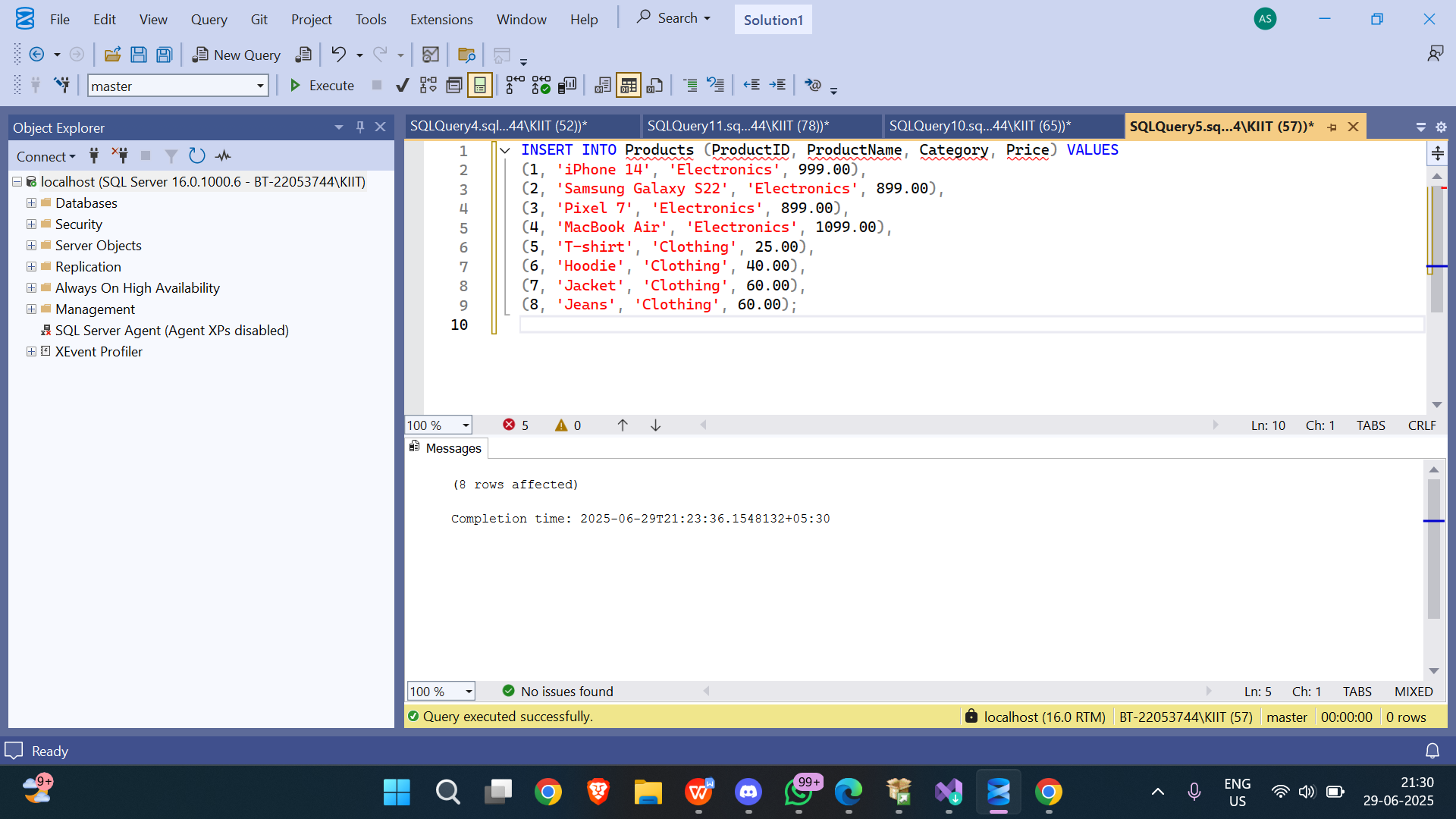
## 1️⃣ Create Table

CREATE TABLE Products (  
 ProductID INT PRIMARY KEY,  
 ProductName NVARCHAR(100),  
 Category NVARCHAR(50),  
 Price DECIMAL(10, 2)  
);



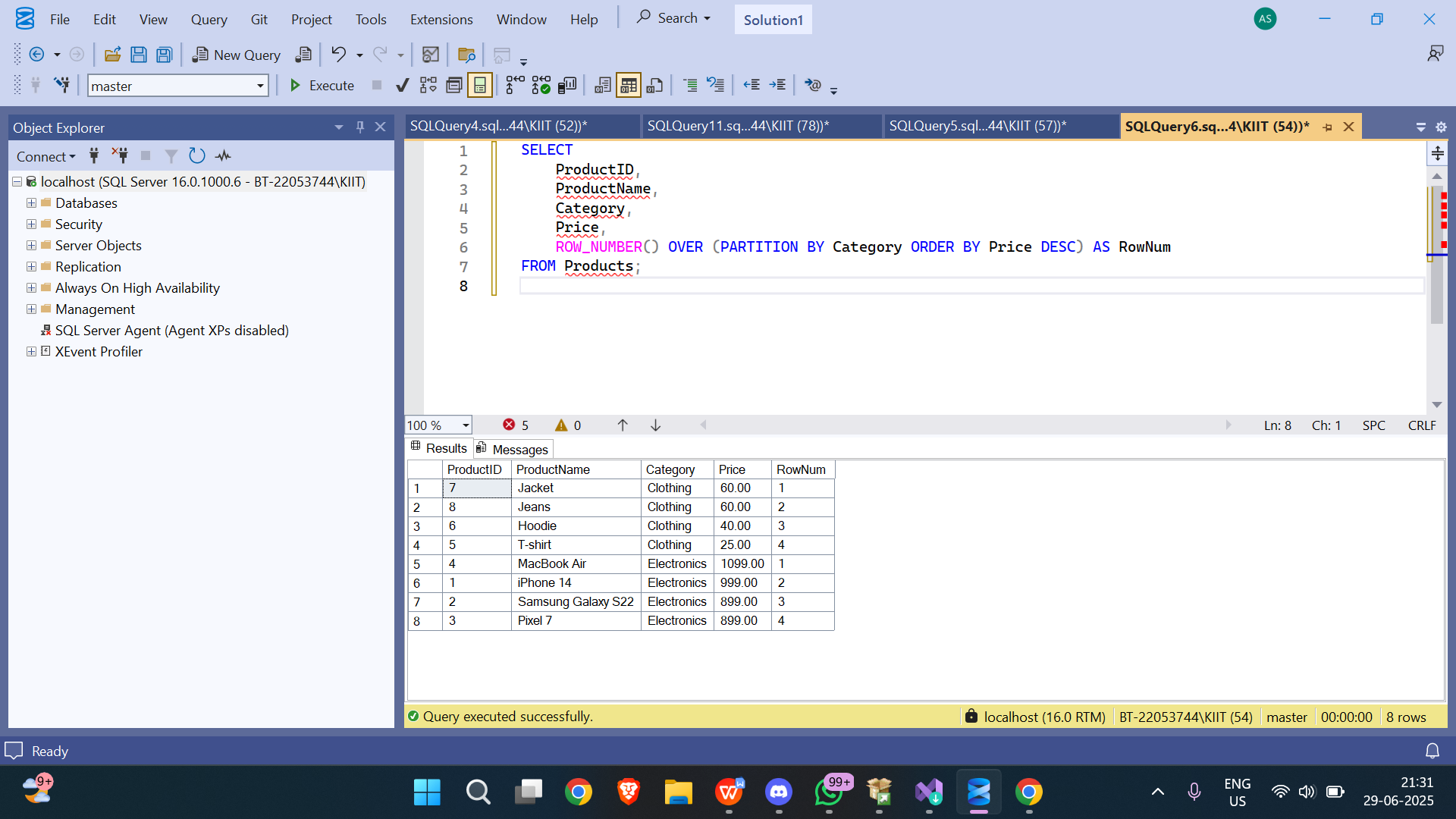
## 2️⃣ Insert Data

INSERT INTO Products (ProductID, ProductName, Category, Price) VALUES  
(1, 'iPhone 14', 'Electronics', 999.00),  
(2, 'Samsung Galaxy S22', 'Electronics', 899.00),  
(3, 'Pixel 7', 'Electronics', 899.00),  
(4, 'MacBook Air', 'Electronics', 1099.00),  
(5, 'T-shirt', 'Clothing', 25.00),  
(6, 'Hoodie', 'Clothing', 40.00),  
(7, 'Jacket', 'Clothing', 60.00),  
(8, 'Jeans', 'Clothing', 60.00);



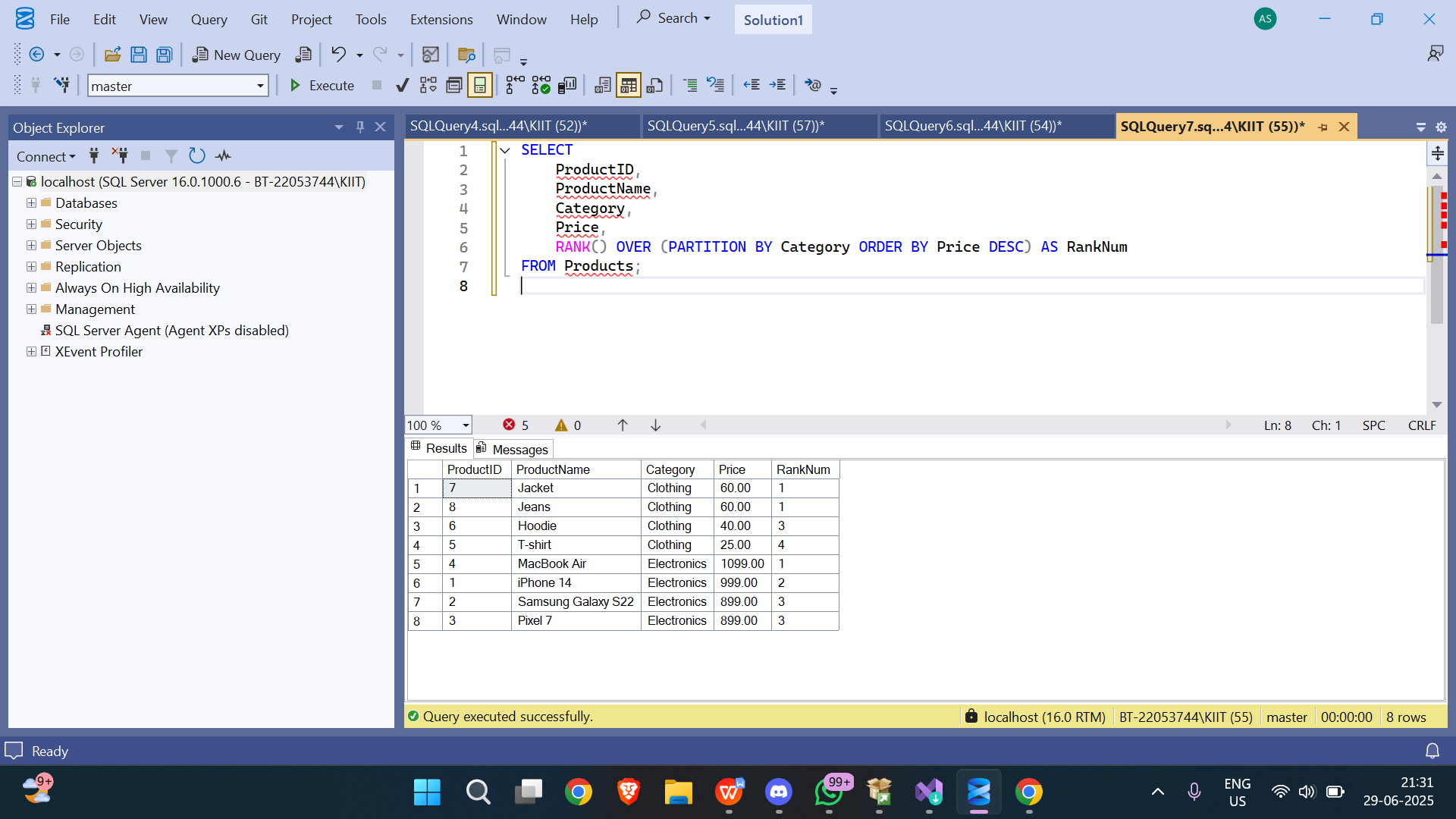
## 3️⃣ Apply ROW\_NUMBER()

SELECT   
 ProductID,  
 ProductName,  
 Category,  
 Price,  
 ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum  
FROM Products;



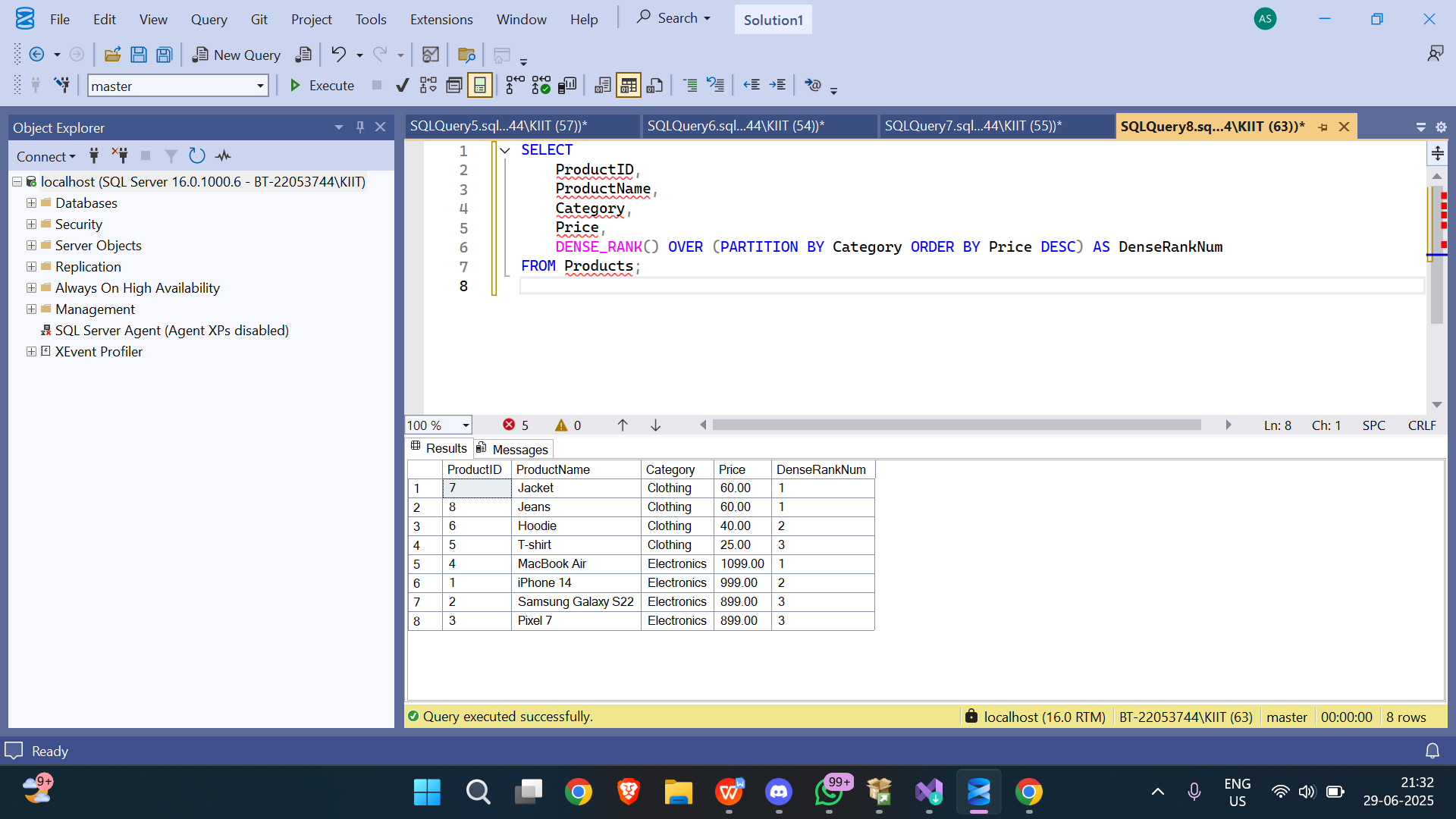
## 4️⃣ Apply RANK()

SELECT   
 ProductID,  
 ProductName,  
 Category,  
 Price,  
 RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankNum  
FROM Products;



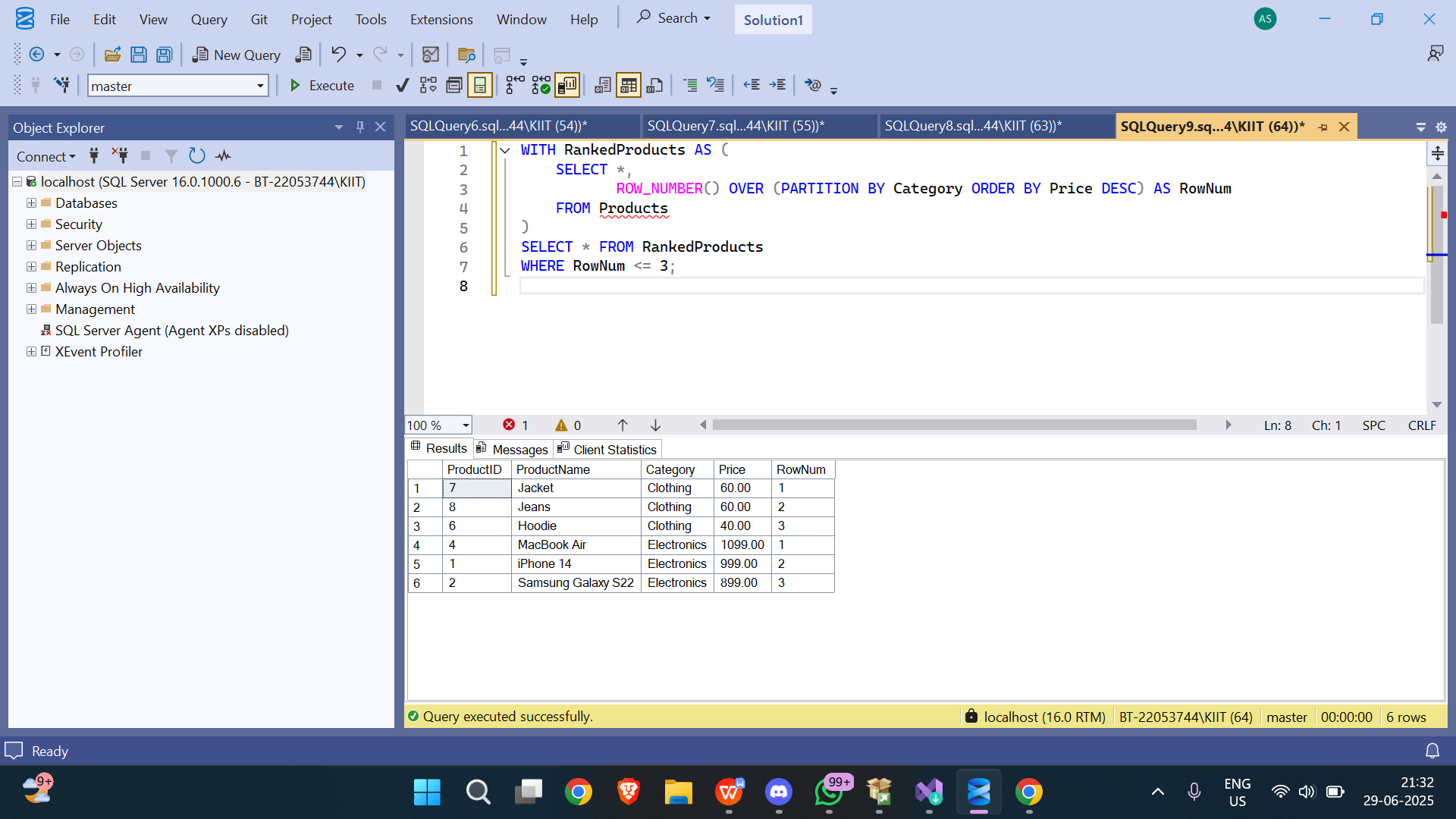
## 5️⃣ Apply DENSE\_RANK()

SELECT   
 ProductID,  
 ProductName,  
 Category,  
 Price,  
 DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum  
FROM Products;



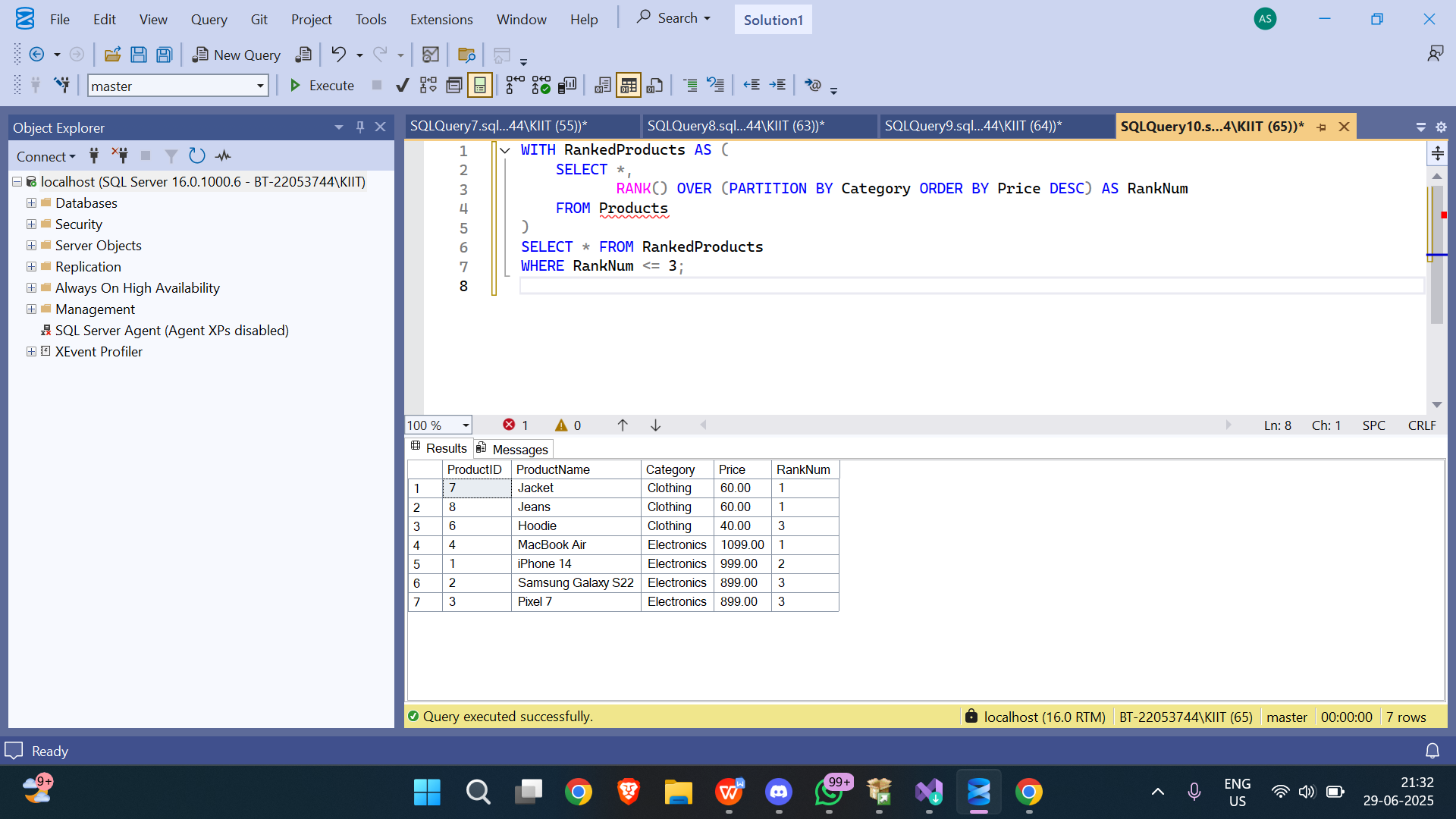
## 6️⃣ Top 3 per Category using ROW\_NUMBER()

WITH RankedProducts AS (  
 SELECT \*,  
 ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum  
 FROM Products  
)  
SELECT \* FROM RankedProducts  
WHERE RowNum <= 3;



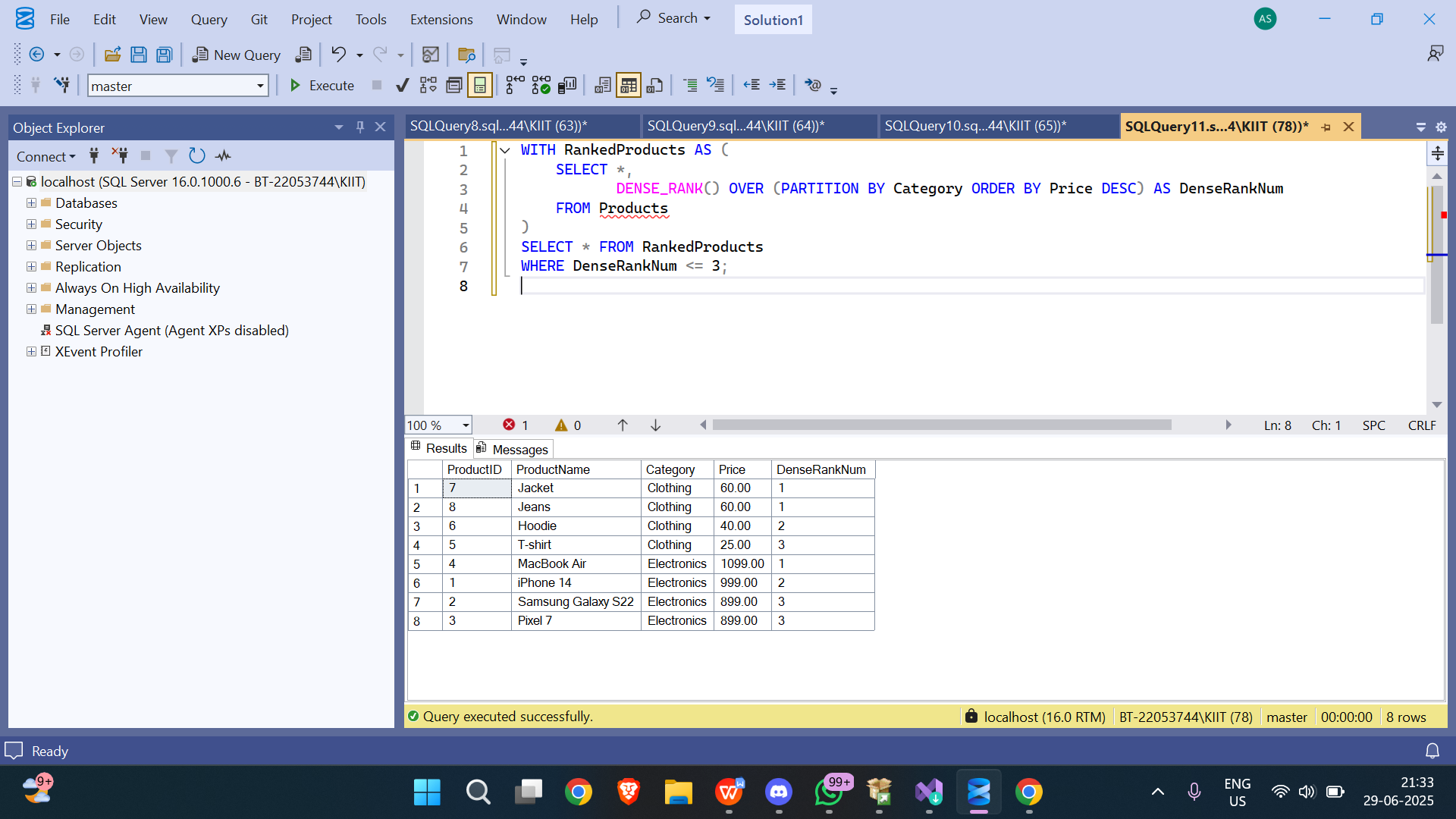
## 7️⃣ Top 3 per Category using RANK()

WITH RankedProducts AS (  
 SELECT \*,  
 RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS RankNum  
 FROM Products  
)  
SELECT \* FROM RankedProducts  
WHERE RankNum <= 3;



## 8️⃣ Top 3 per Category using DENSE\_RANK()

WITH RankedProducts AS (  
 SELECT \*,  
 DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRankNum  
 FROM Products  
)  
SELECT \* FROM RankedProducts  
WHERE DenseRankNum <= 3;



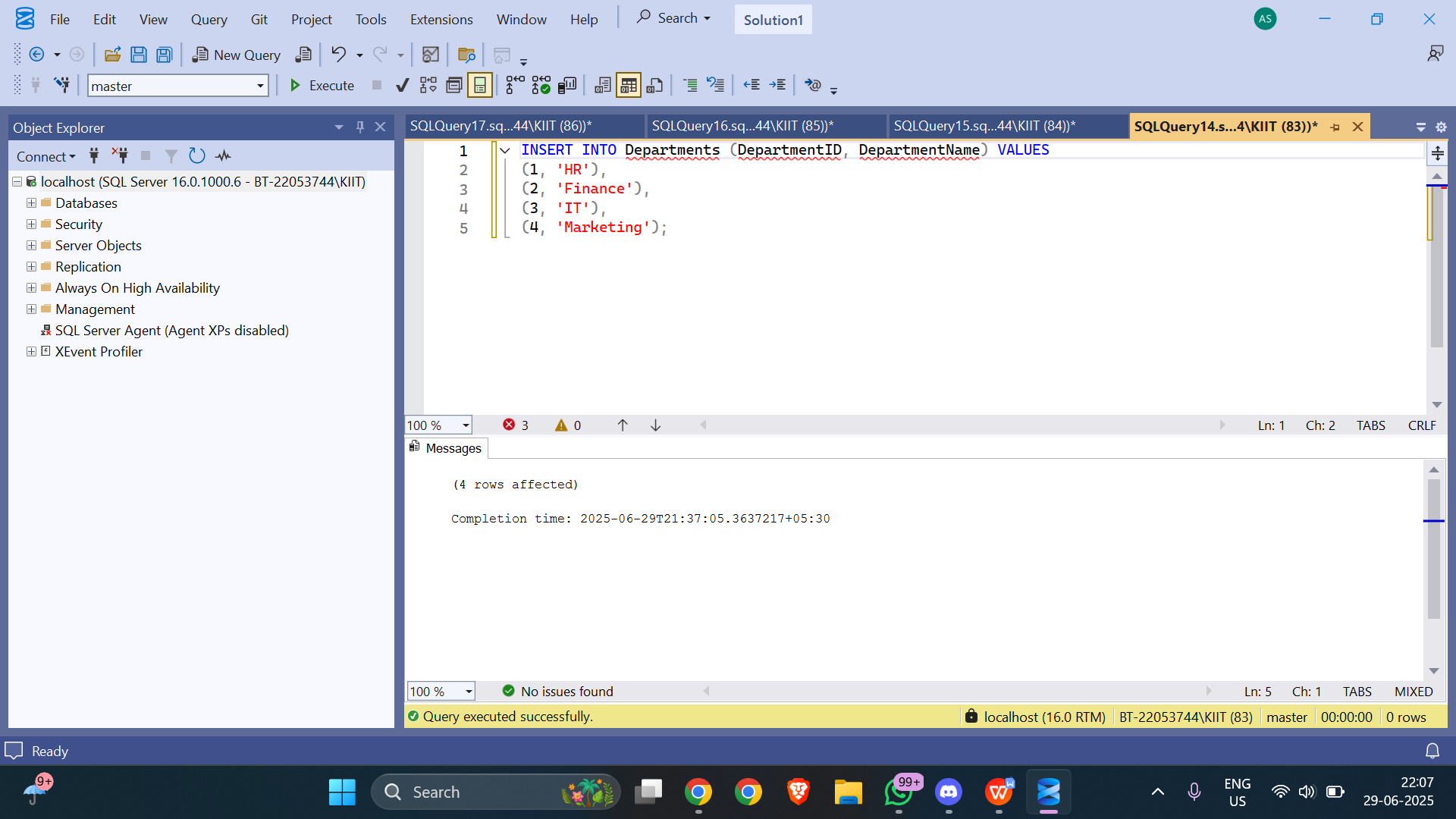
Employee Management System - SQL Exercises

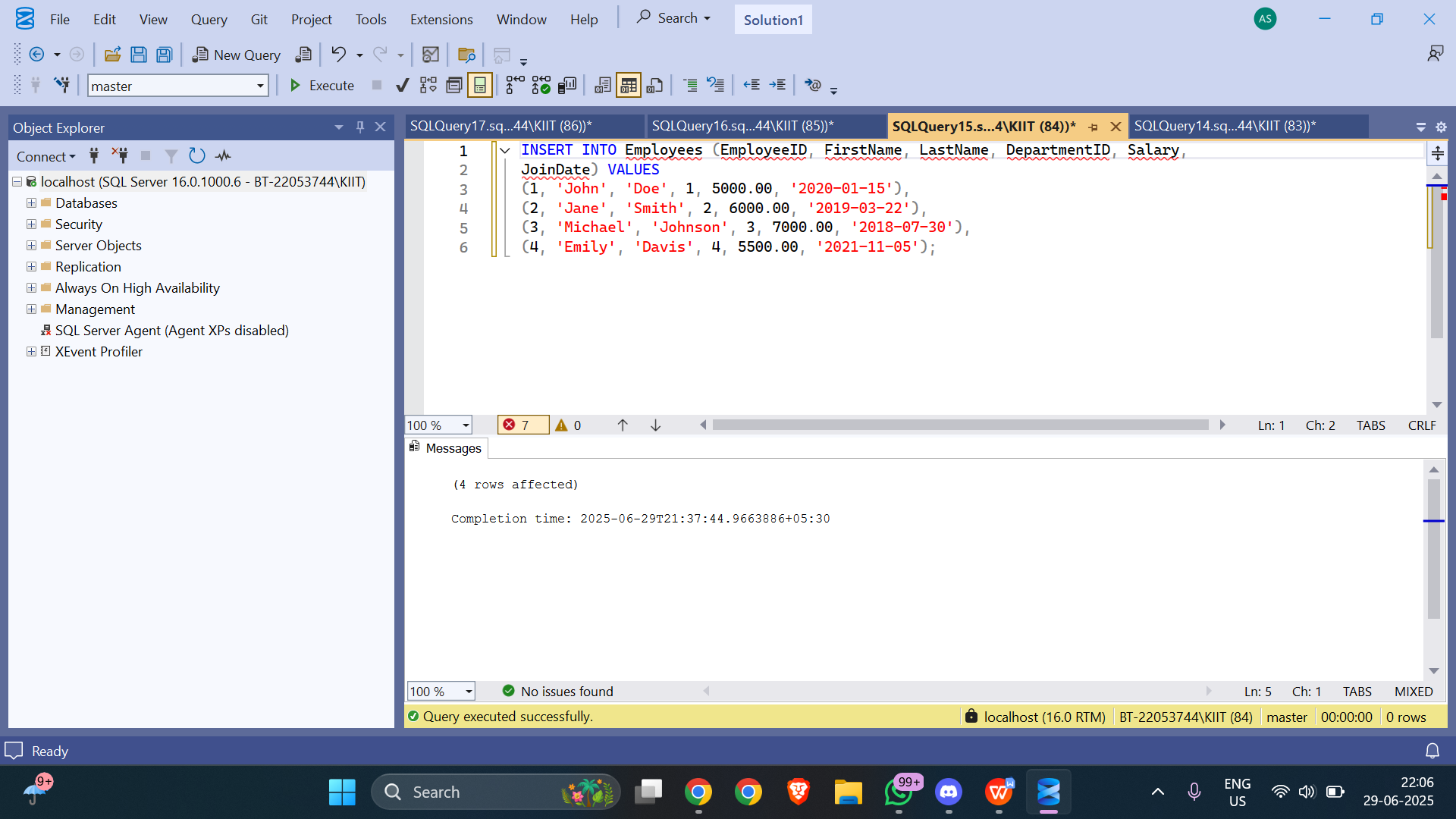
# Exercise 1: Create a Stored Procedure

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

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

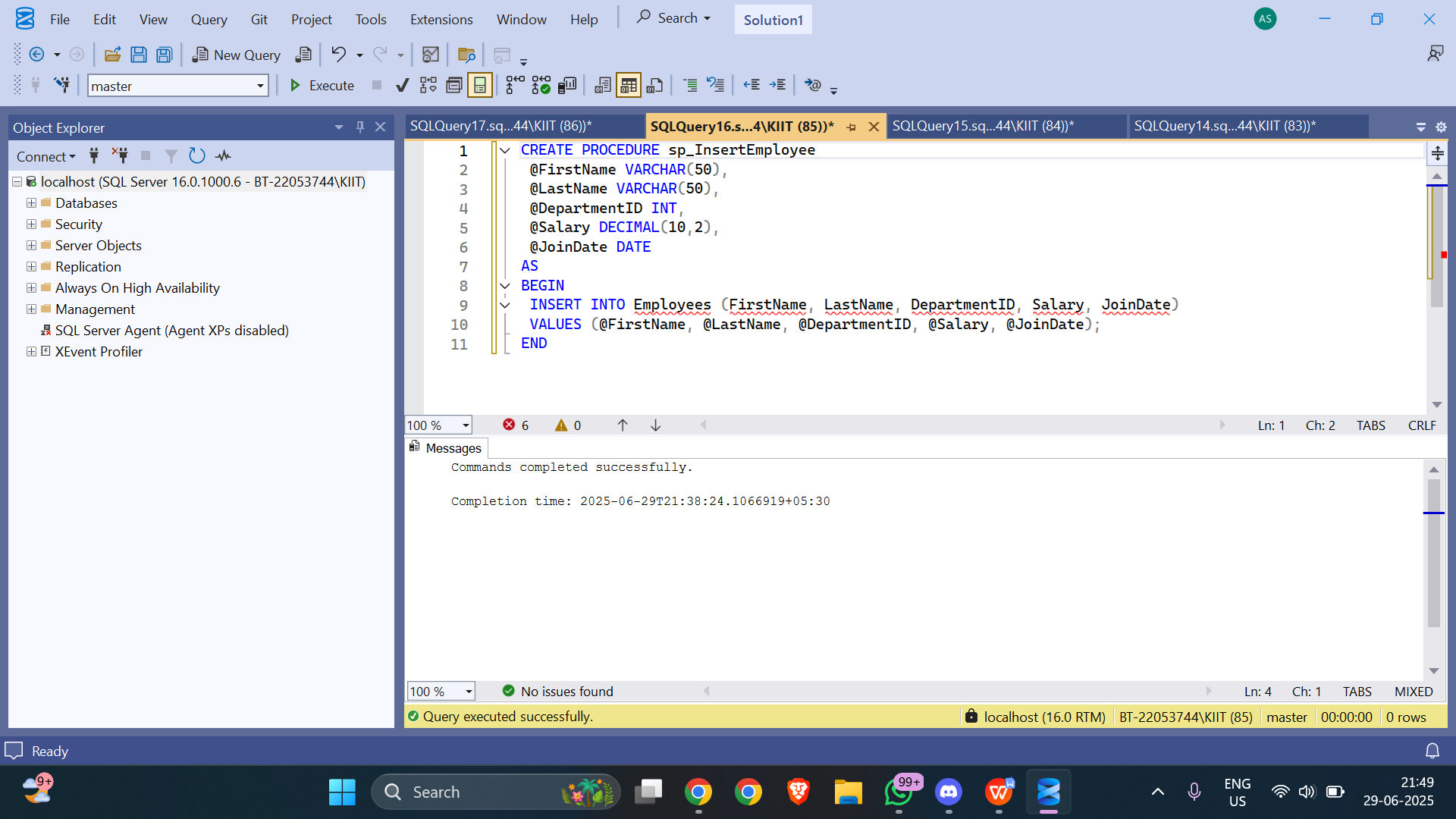
Step 2: Write the SQL query to select employee details based on the DepartmentID.





Step 3: 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;



# Exercise 5: Return Data from a Stored Procedure

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

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

Step 2: Write the SQL query to count the number of employees in the specified department.

Step 3: Save the stored procedure by executing the Stored procedure content.

CREATE PROCEDURE sp\_CountEmployeesByDepartment  
 @DepartmentID INT  
AS  
BEGIN  
 SELECT COUNT(\*) AS TotalEmployees  
 FROM Employees  
 WHERE DepartmentID = @DepartmentID;  
END;

