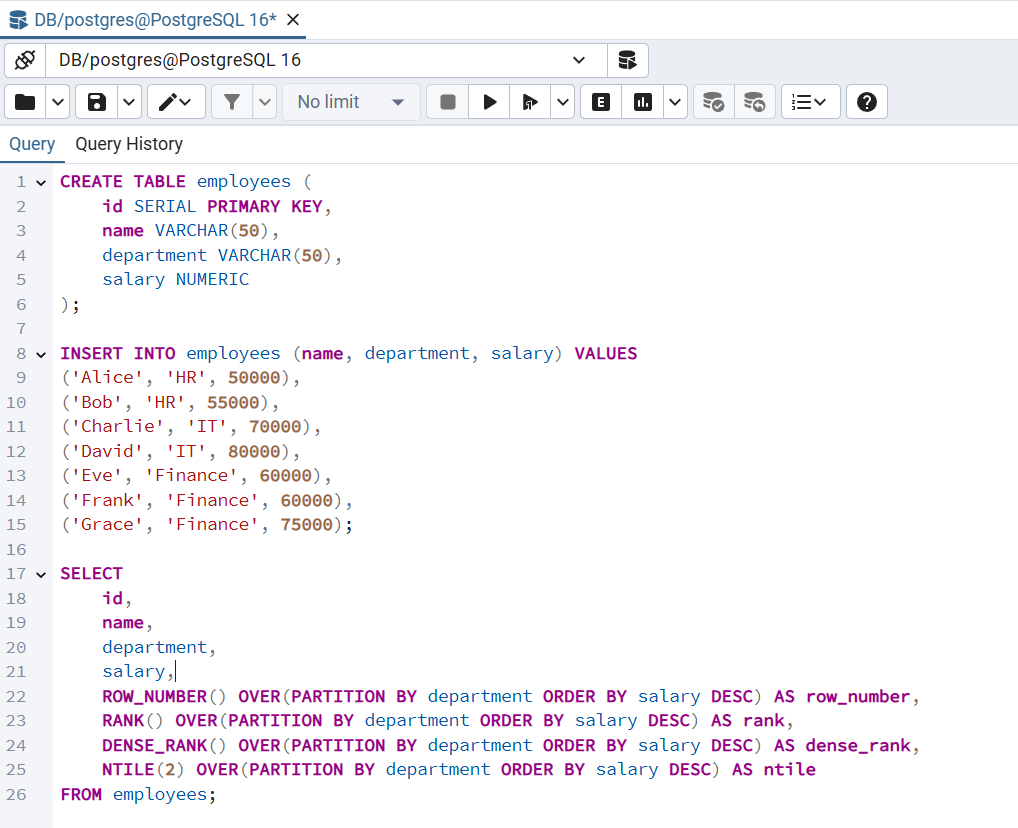
**Question 1:**

Write SQL queries using **Ranking and Window Functions** like:

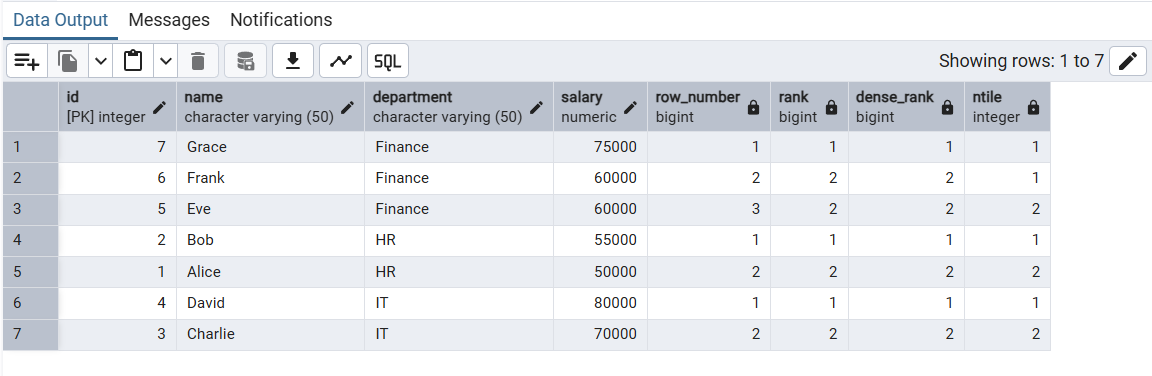
* ROW\_NUMBER()
* RANK()
* DENSE\_RANK()
* NTILE()

on a sample table of employees to rank them by their salary **within each department**.

**Code:**



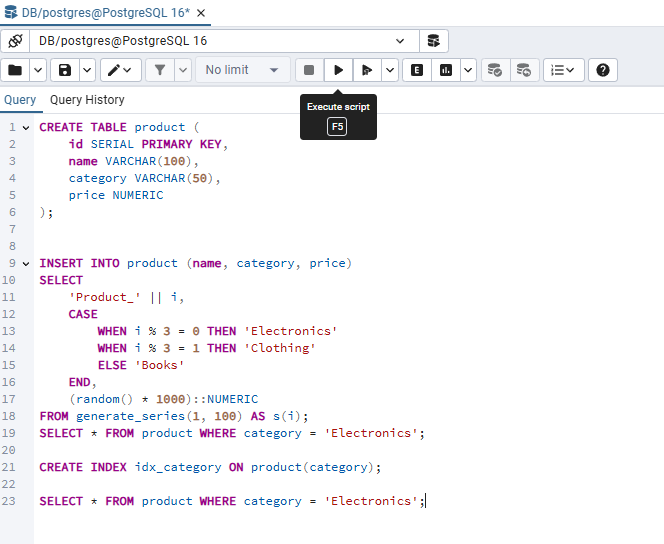
**Output:**



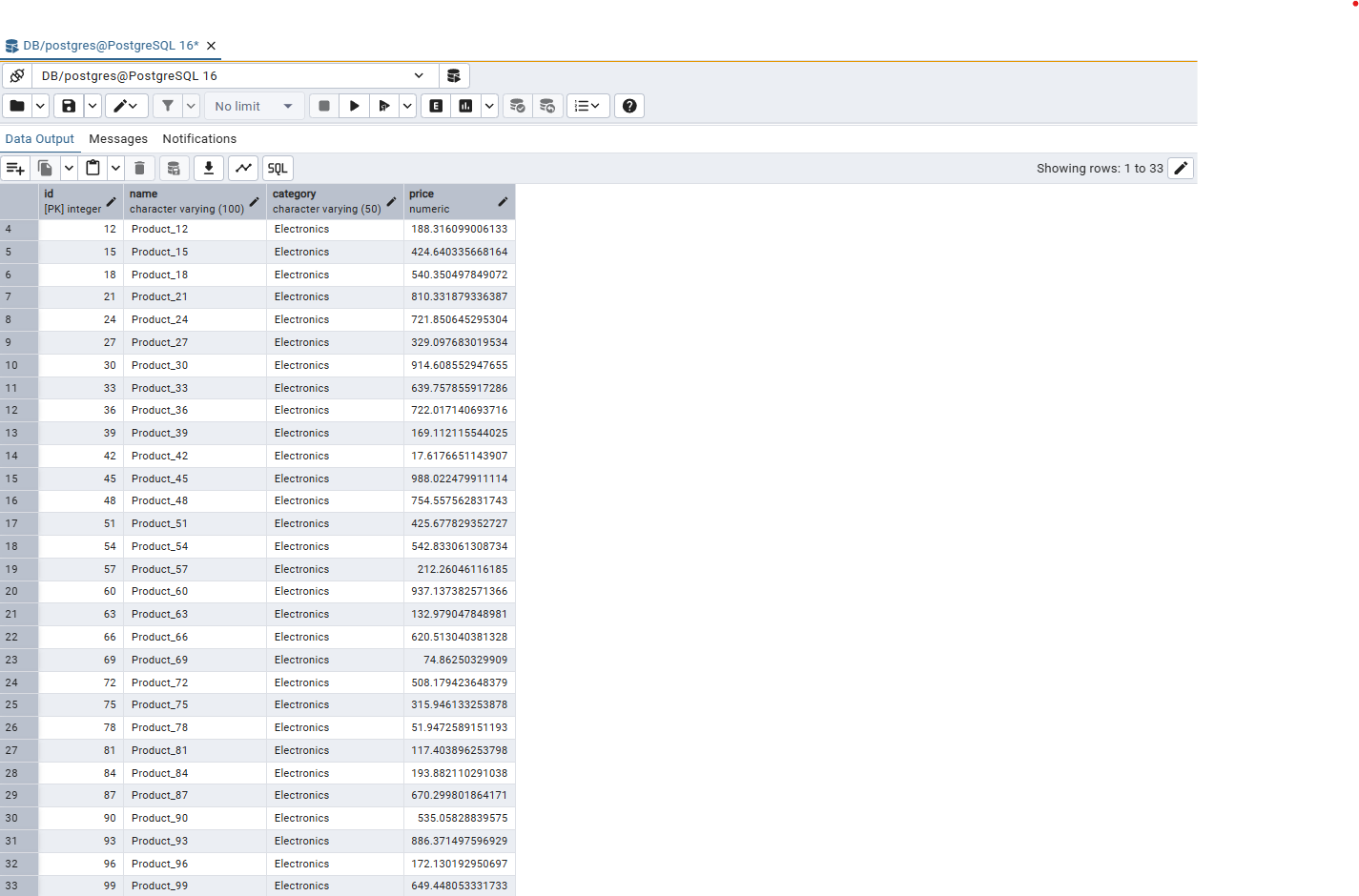
**Question 2:**

Create an **index** on a column in a table and show how it improves the performance of a query.

**Code:**



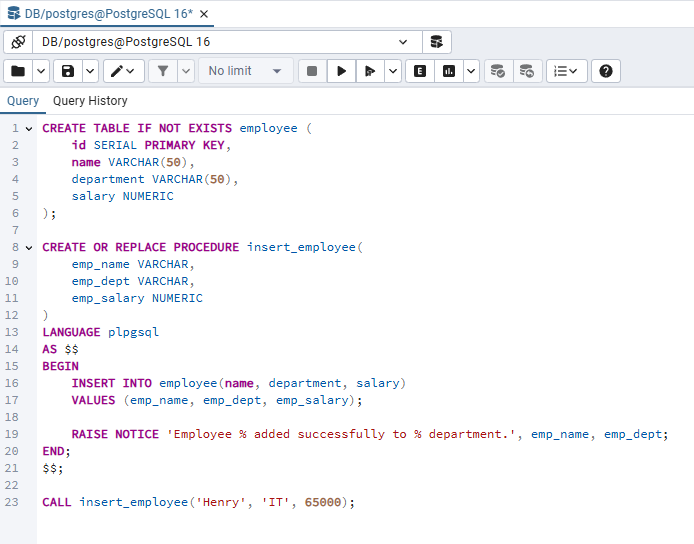
**Output:**

****

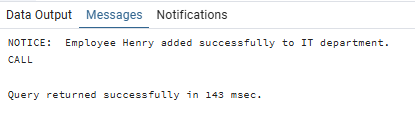
**Question 3:**

Write a stored procedure that inserts a new employee into the employee table and prints a message.

**Code:**

****

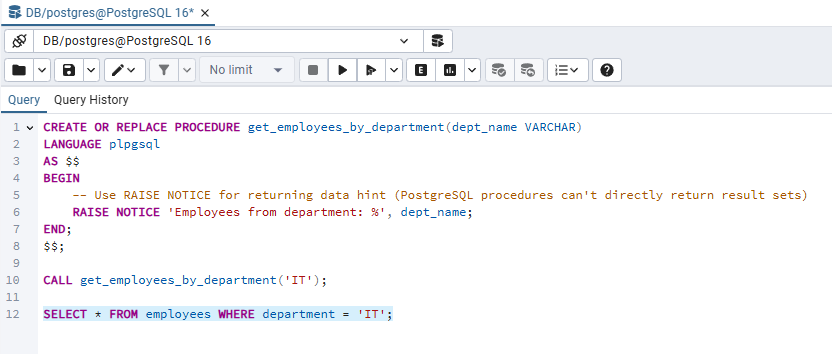
**Output:**

****

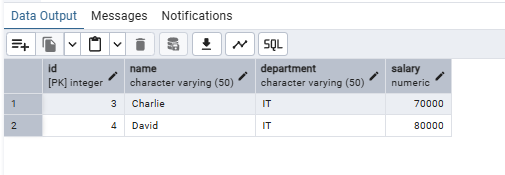
**Question 4:**

Write a stored procedure that returns a list of employees from a specific department.

**Code:**

****

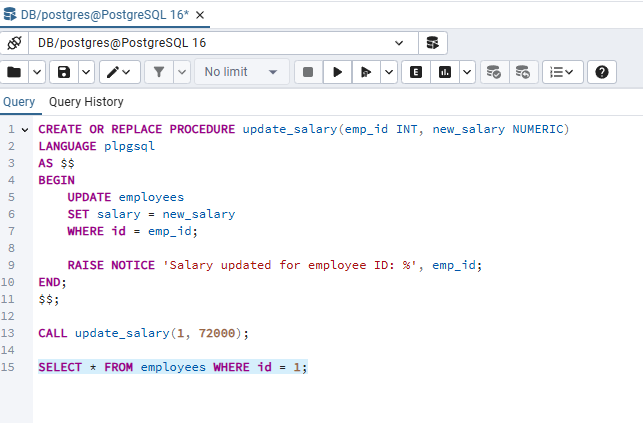
**Output:**

****

**Question 5:**

Execute a stored procedure that updates an employee's salary.

**Code:**

****

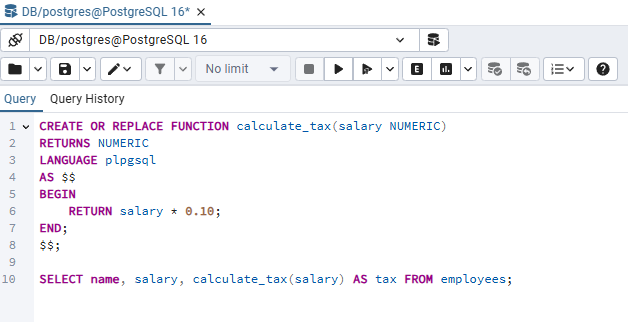
**Output:**

****

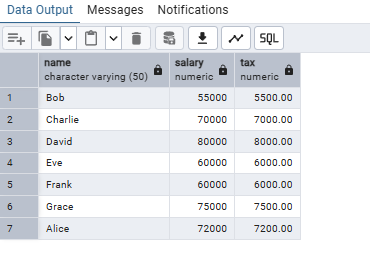
**Question 6:**

Write a user-defined function that takes employee salary and returns tax as 10% of the salary.

**Code:**

****

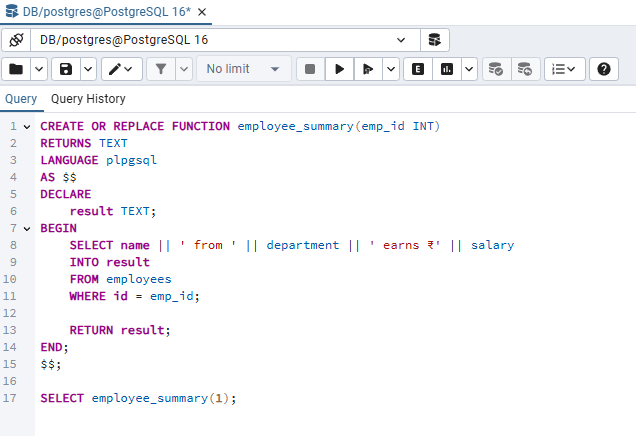
**Output:**

****

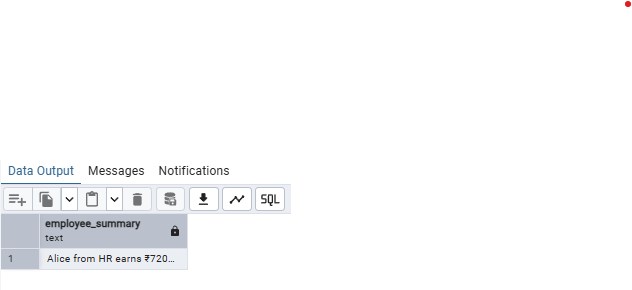
**Question 7:**

Write a function that returns full employee info as a concatenated string: “Name from Department earns Salary”

**Code:**

****

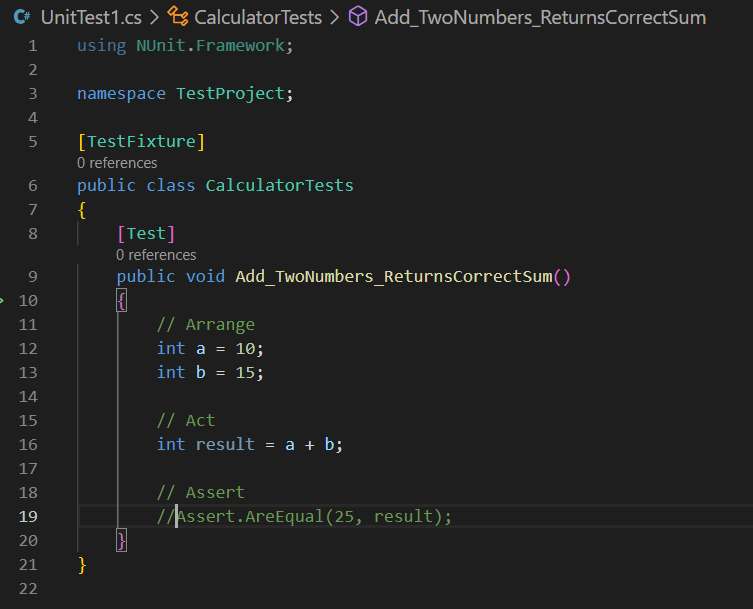
**Output:**

****

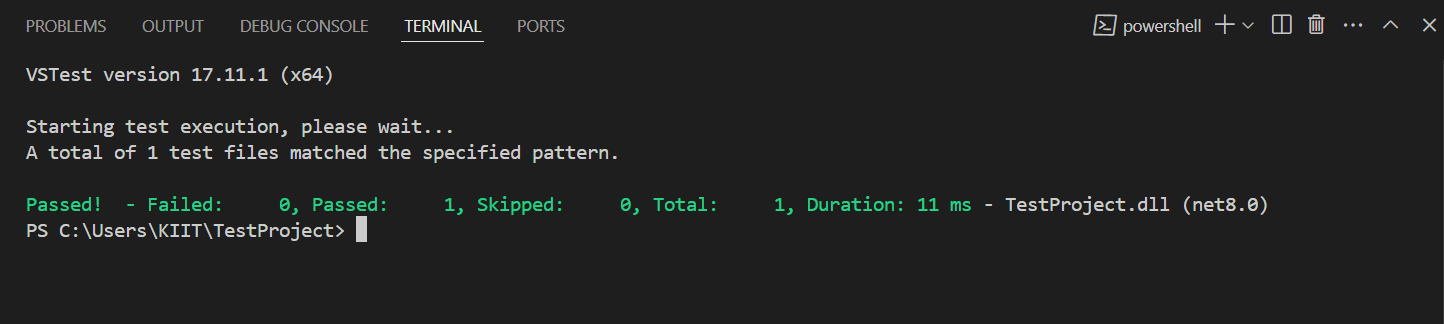
**Question 8:**

Write a unit test using **NUnit** to verify that the addition of two integers returns the correct result.  
Use the Assert.AreEqual() method to validate the expected output.

**Code:**

****

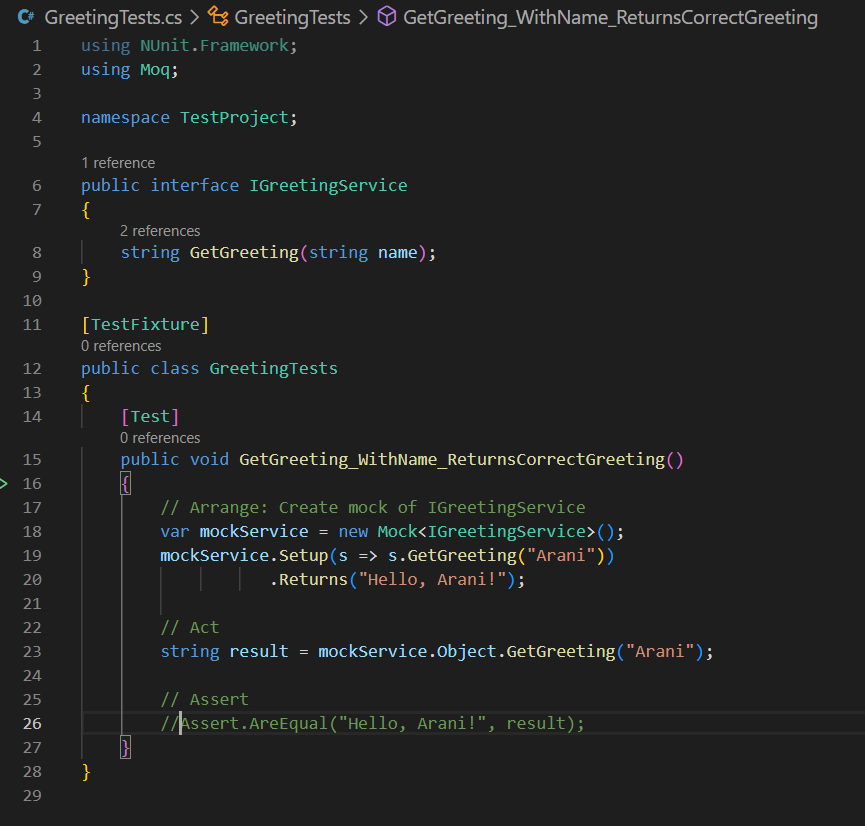
**Output:**

****

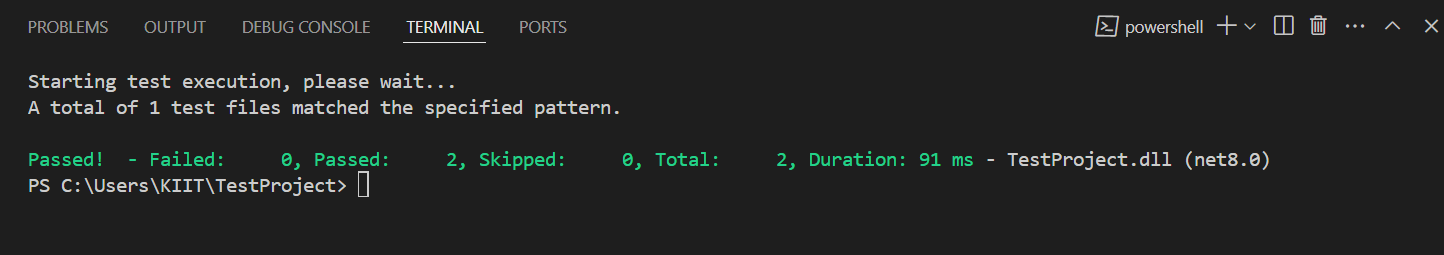
**Question 9:**

Write testable code using Moq – e.g., mock an interface and test behavior.

**Code:**

****

**Output:**

****