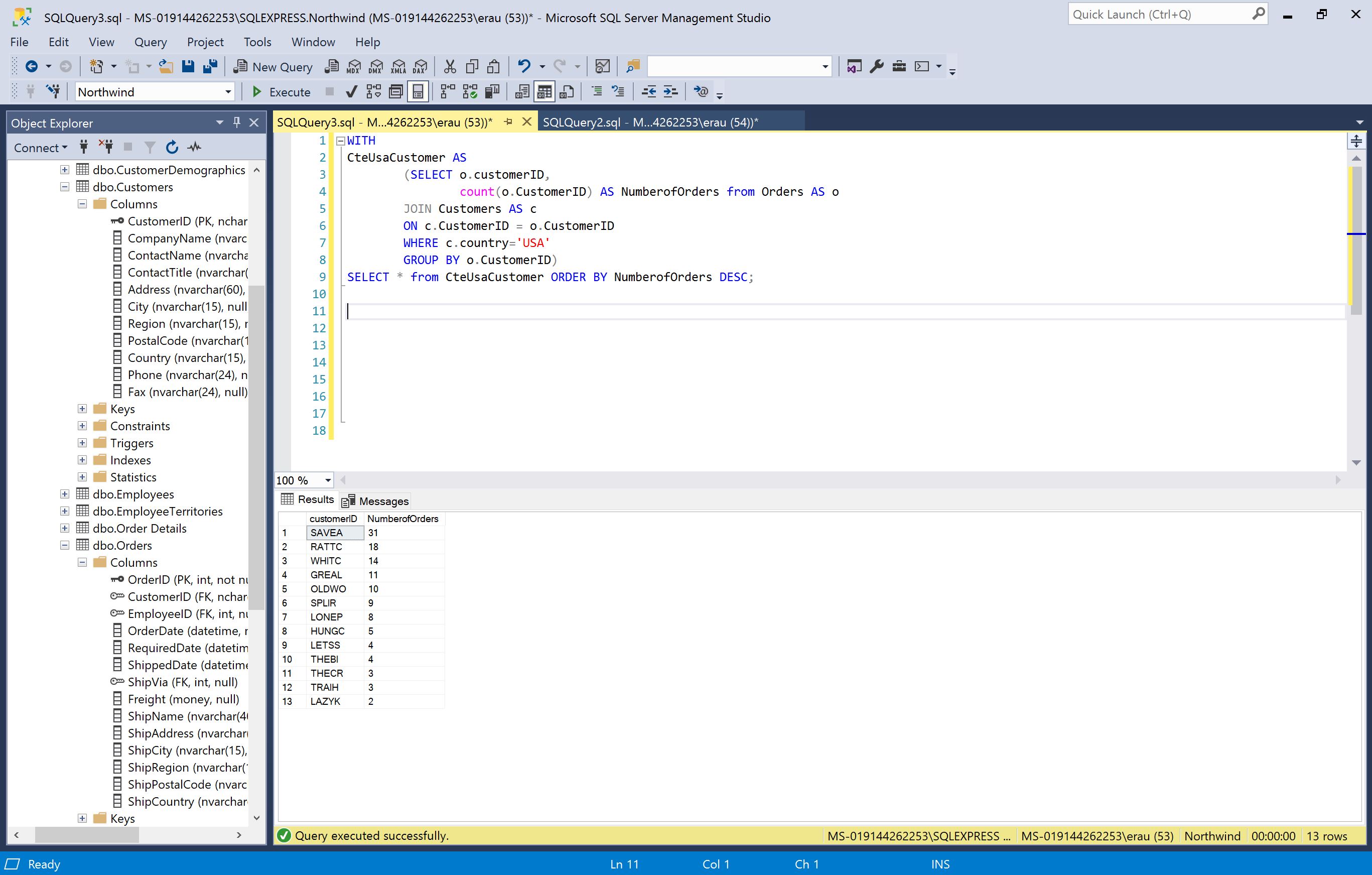
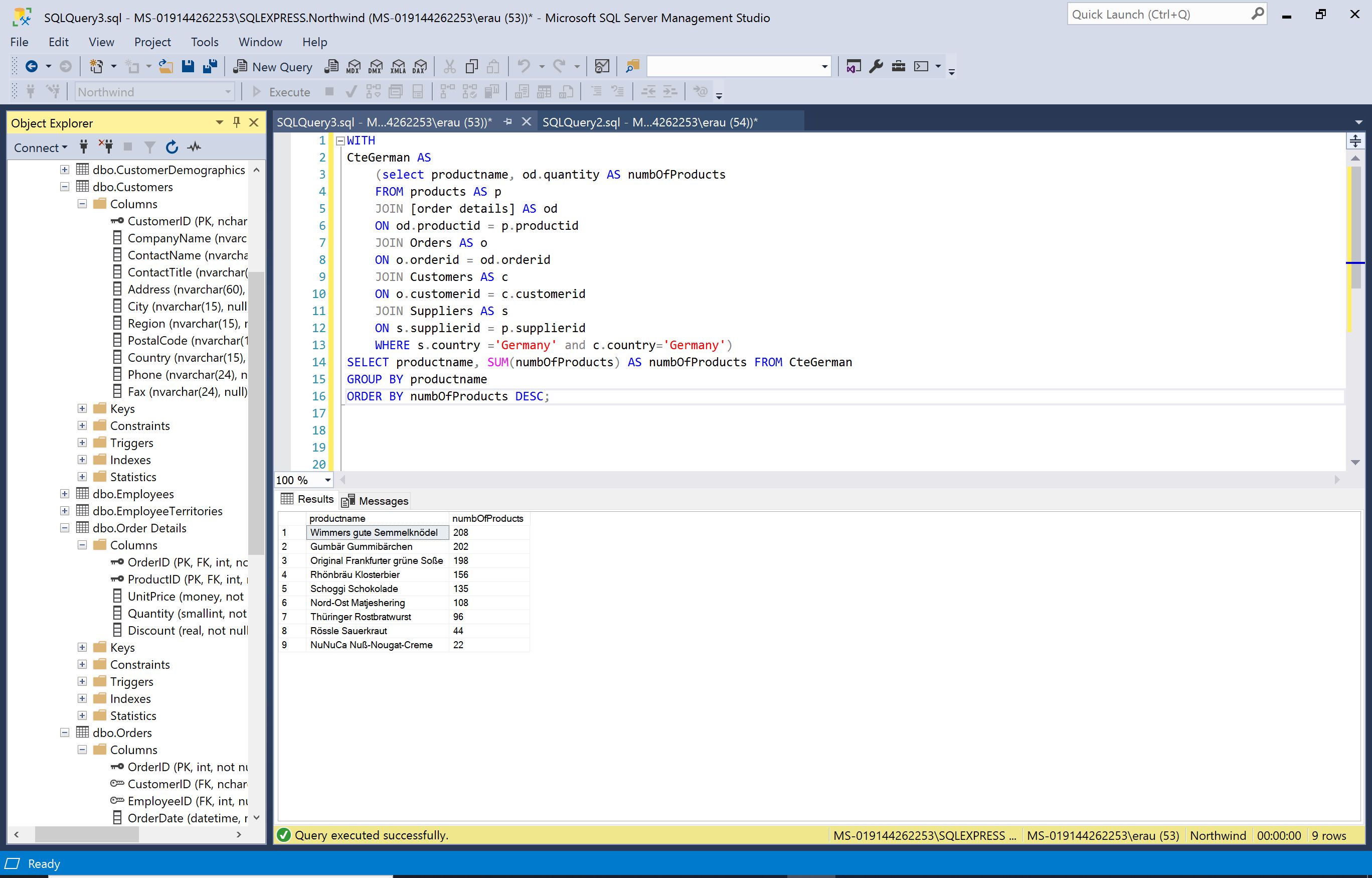
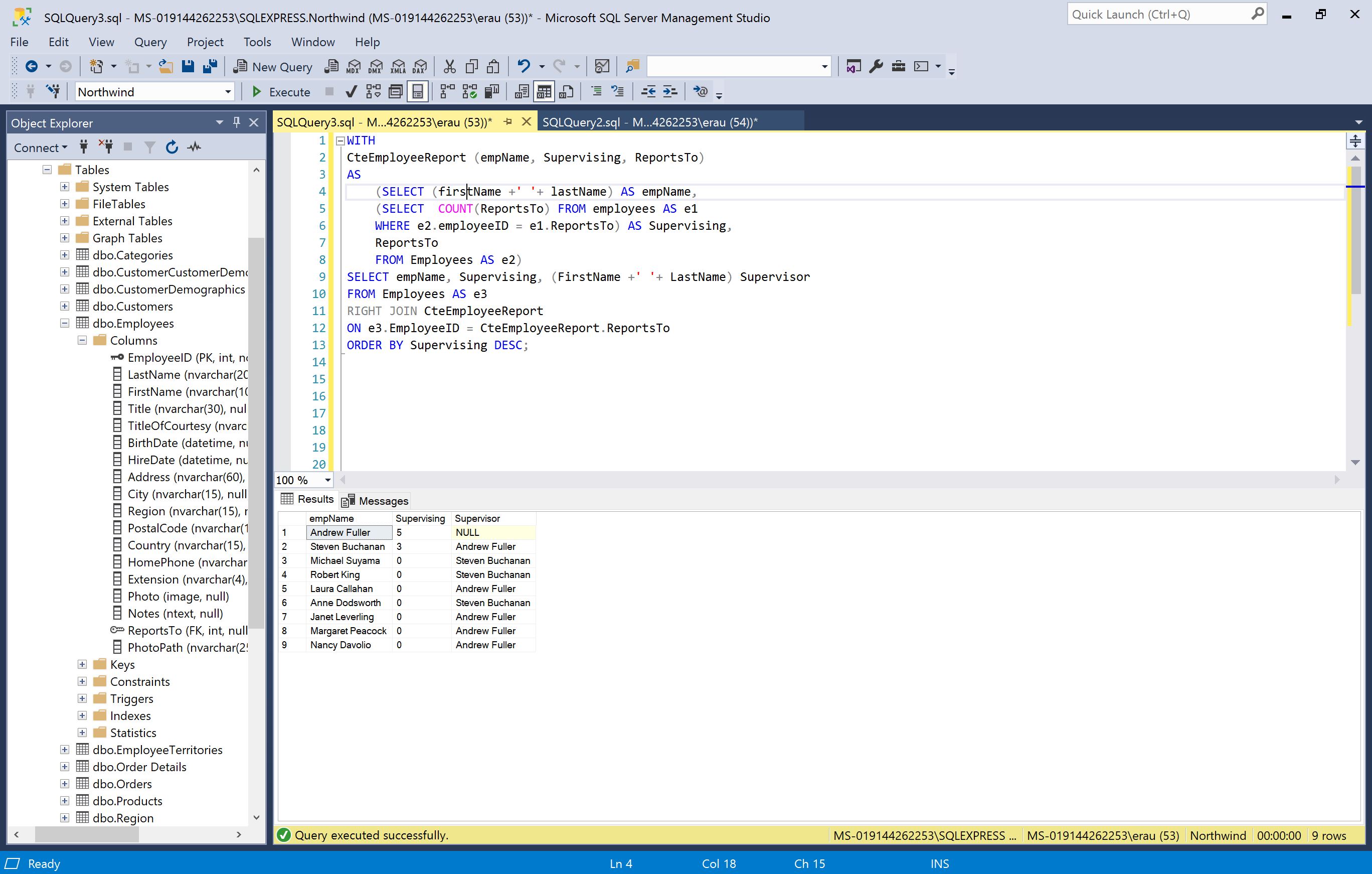
1. List the number of orders by each customer who lives in the United States using a CTE. Sort from highest to lowest.



1. List the product name and the number of each product from a German supplier sold to a customer in Germany using a CTE. Sort from highest to lowest.

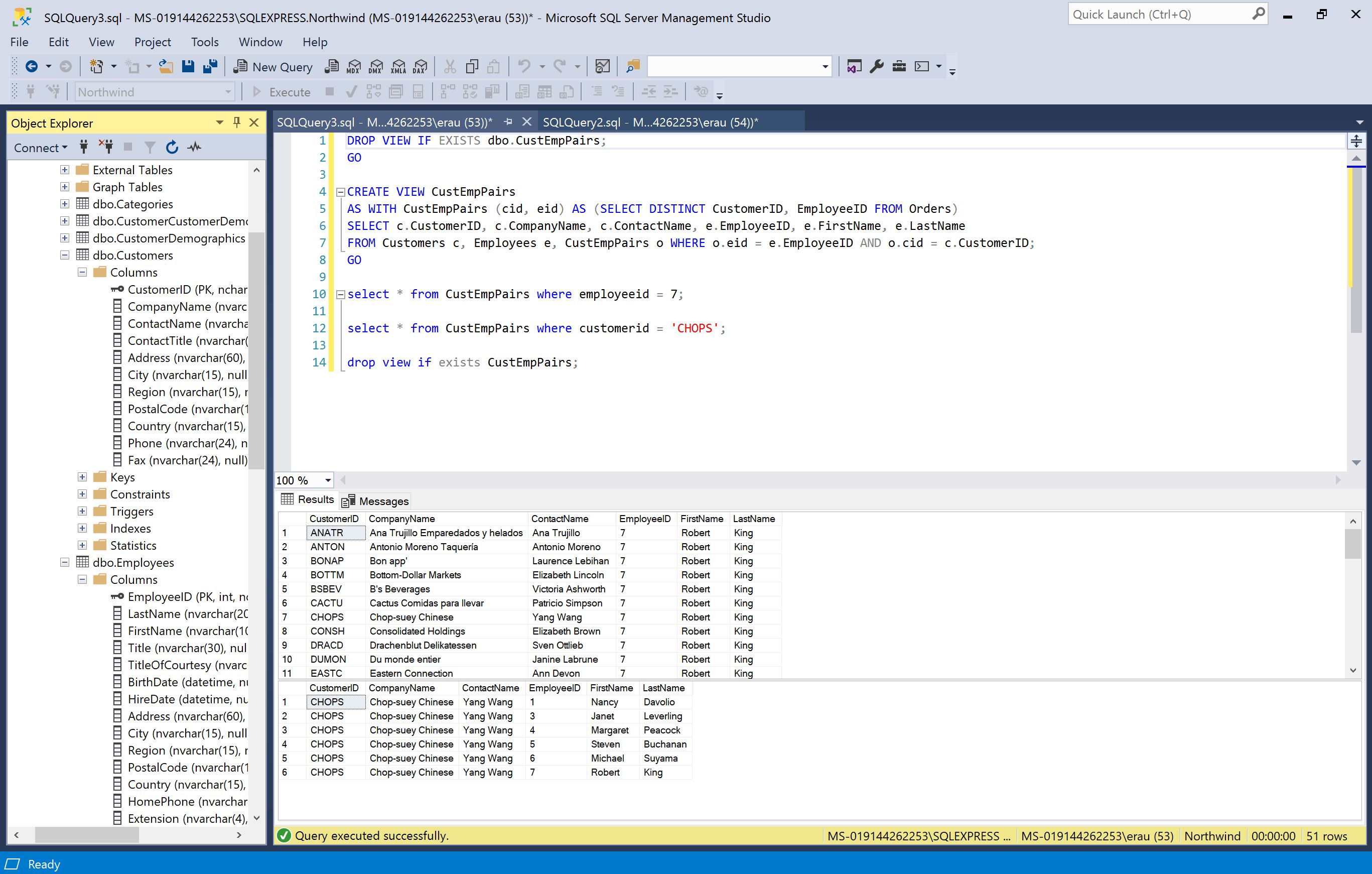


1. Prepare an employee report showing the name of each employee, the number of employees they supervise, and the name of their supervisor using a CTE. Sort by the number of employees supervised.



1. One purpose of views is to denormalize databases for the purpose of efficiency, both machine efficiency and programmer efficiency. Creating denormalized objects can turn complex queries into simple ones.   
     
   For example, suppose you needed a list of all employees who took orders for a specific customer, or all customers who were served by a specific employee. You can create a “table” as a view that contains distinct pairs of customers and employees. This is somewhat complex, so do this in steps:
   1. Create a query that returns every distinct customer/employee pair.
   2. Use that query to write another query turning the customerid, customername, and customercontact, and the employeeid, firstname, and lastname.
   3. Make sure you drop any view that might exist.
   4. Create a view based on your query.
   5. Write a report listing all customers served by employee 7, Robert King.
   6. Write a report listing all employees who served customer CHOPS, Chop-suey Chinese.
   7. Drop the view.

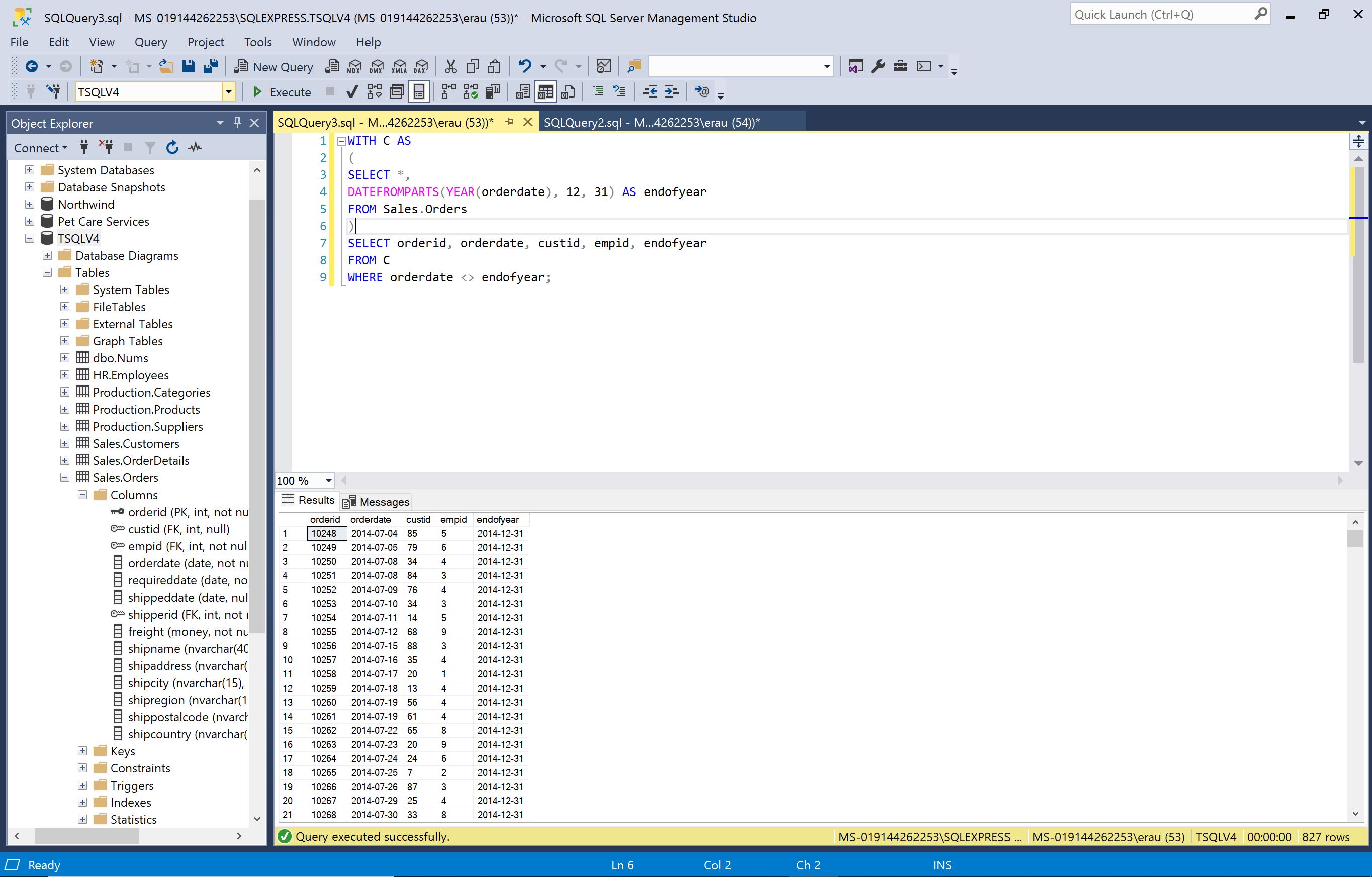
**\*See Next Page for Solution\***



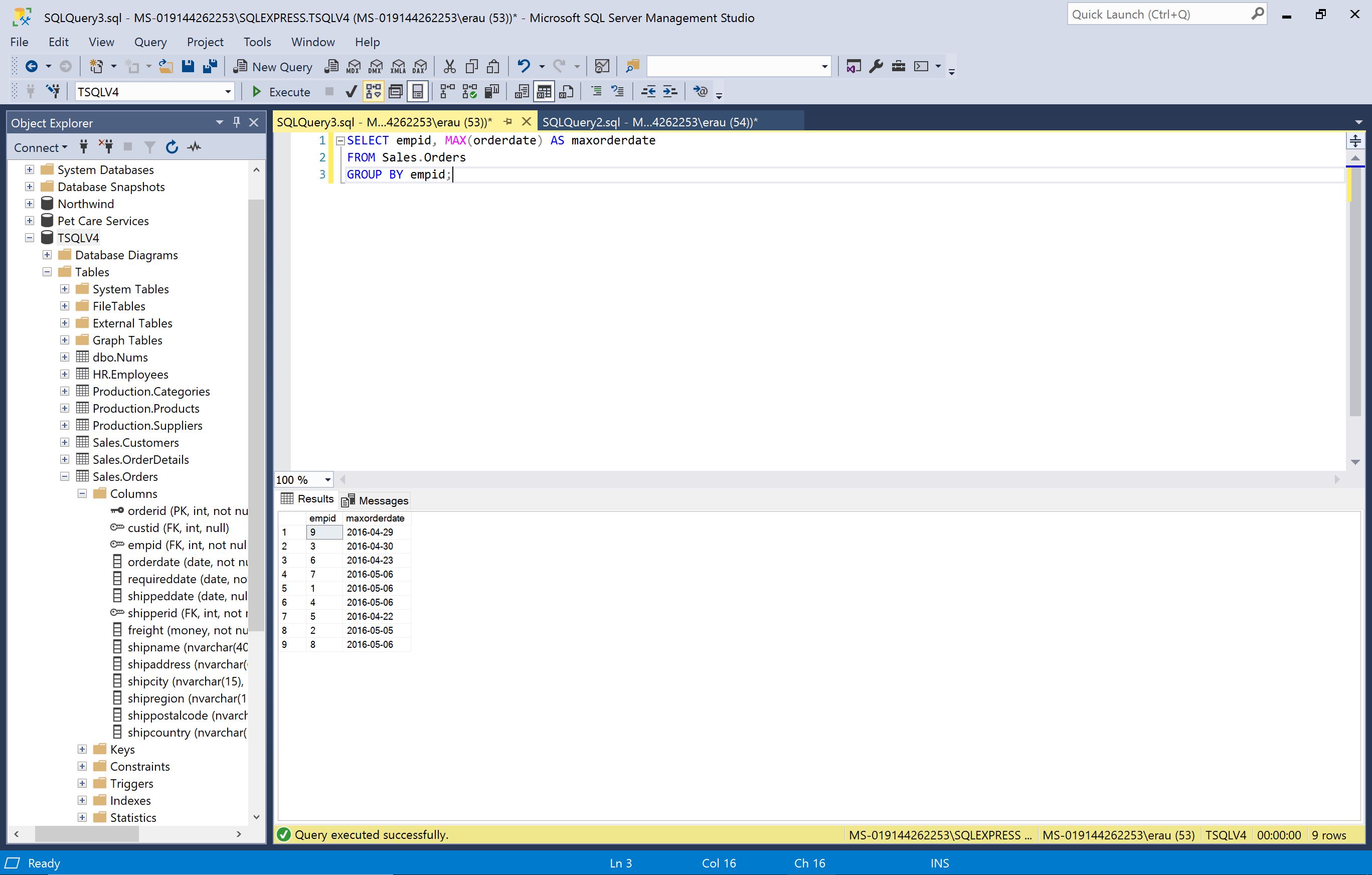
**TSQLV4**

1. When you try to run this query, you get an error. Explain the problem and suggest a solution.

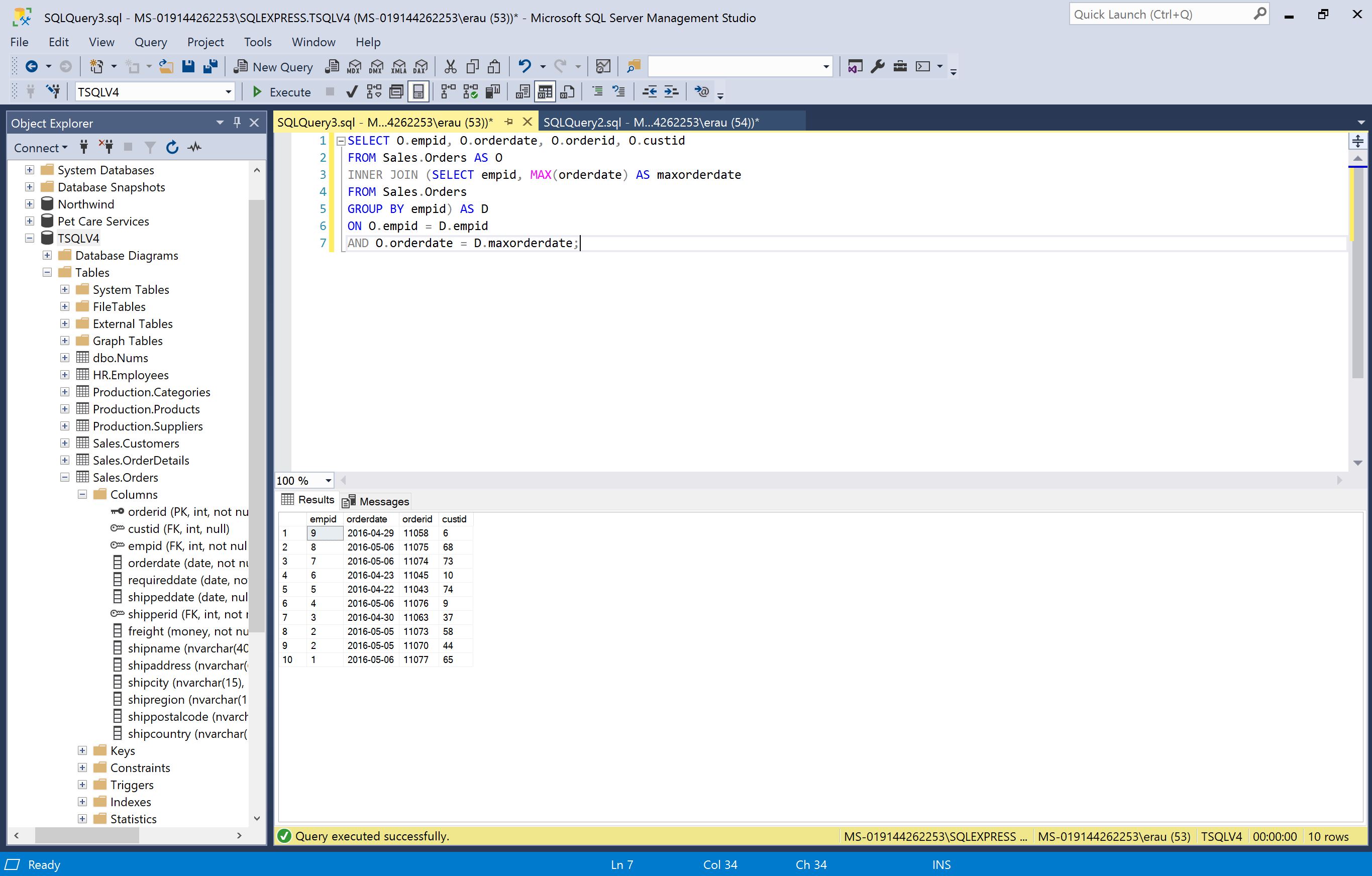
The SELECT clause is evaluated after the WHERE clause. This means you’re not allowed to refer to an alias you create in the SELECT clause within the WHERE clause.



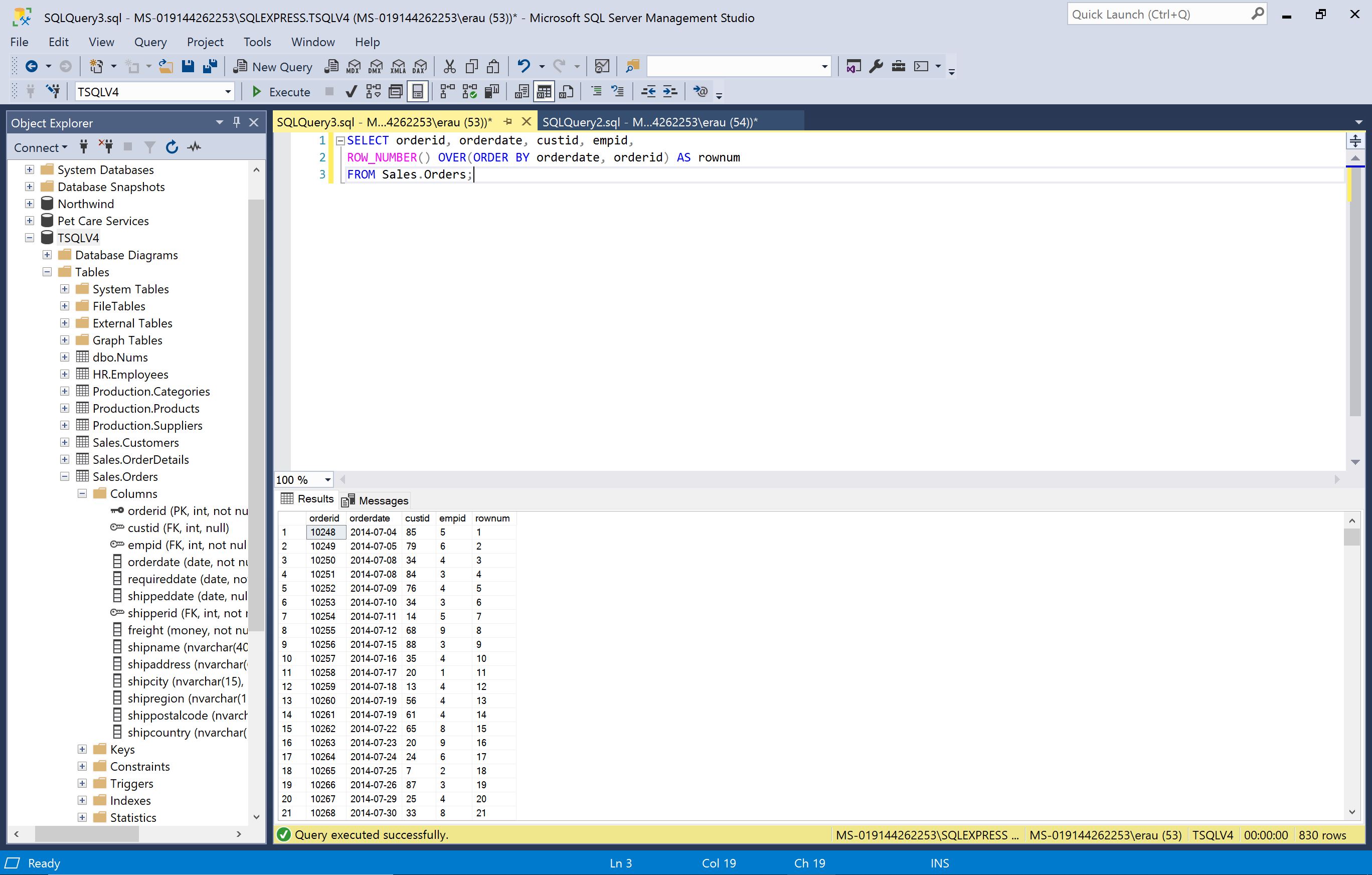
2-1. Return the maximum value in the orderdate column for each employee



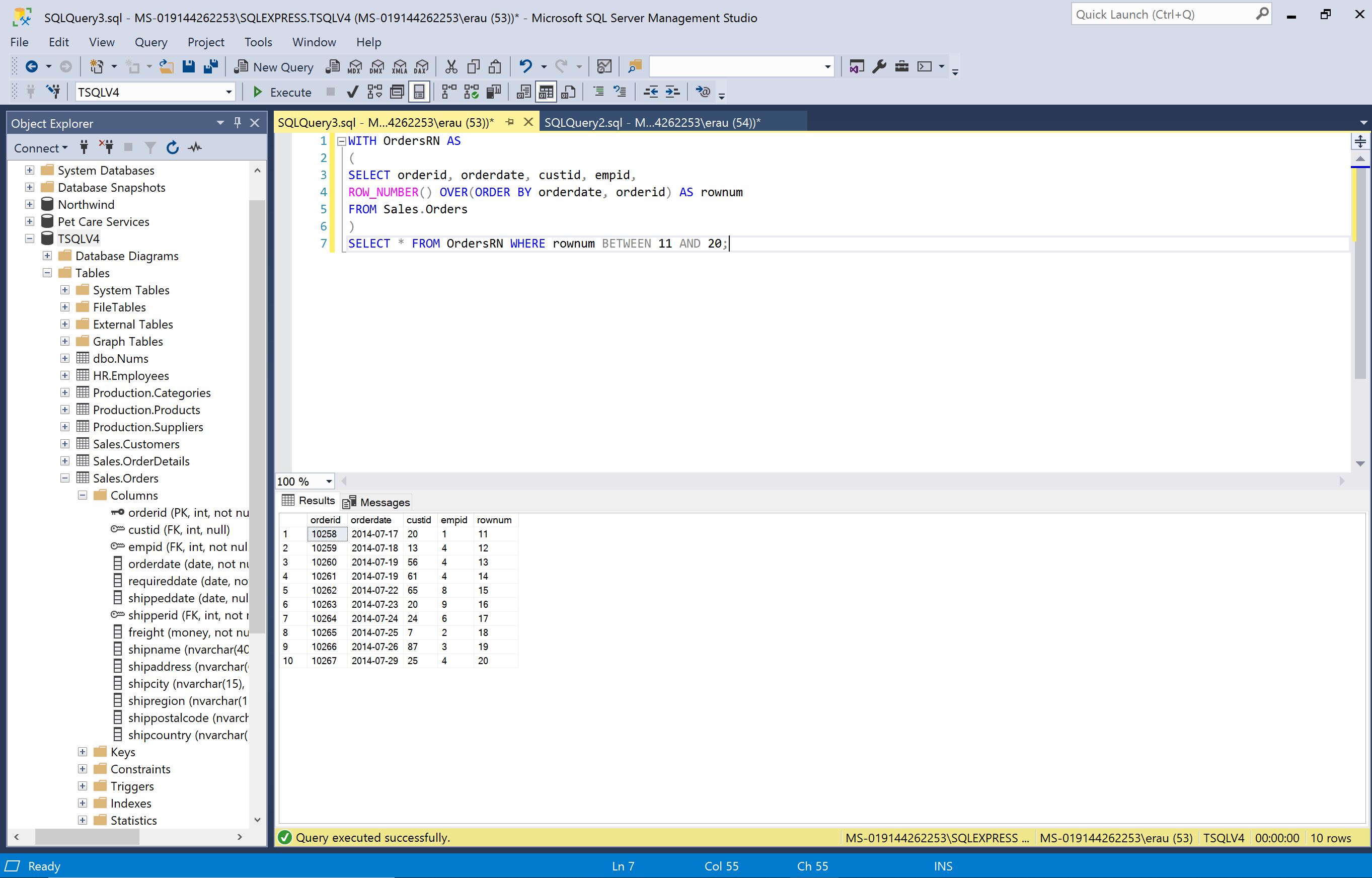
2-2. Encapsulate the query from Exercise 2-1 in a derived table. Write a join query between the derived table and the Orders table to return the orders with the maximum order date for each employee.



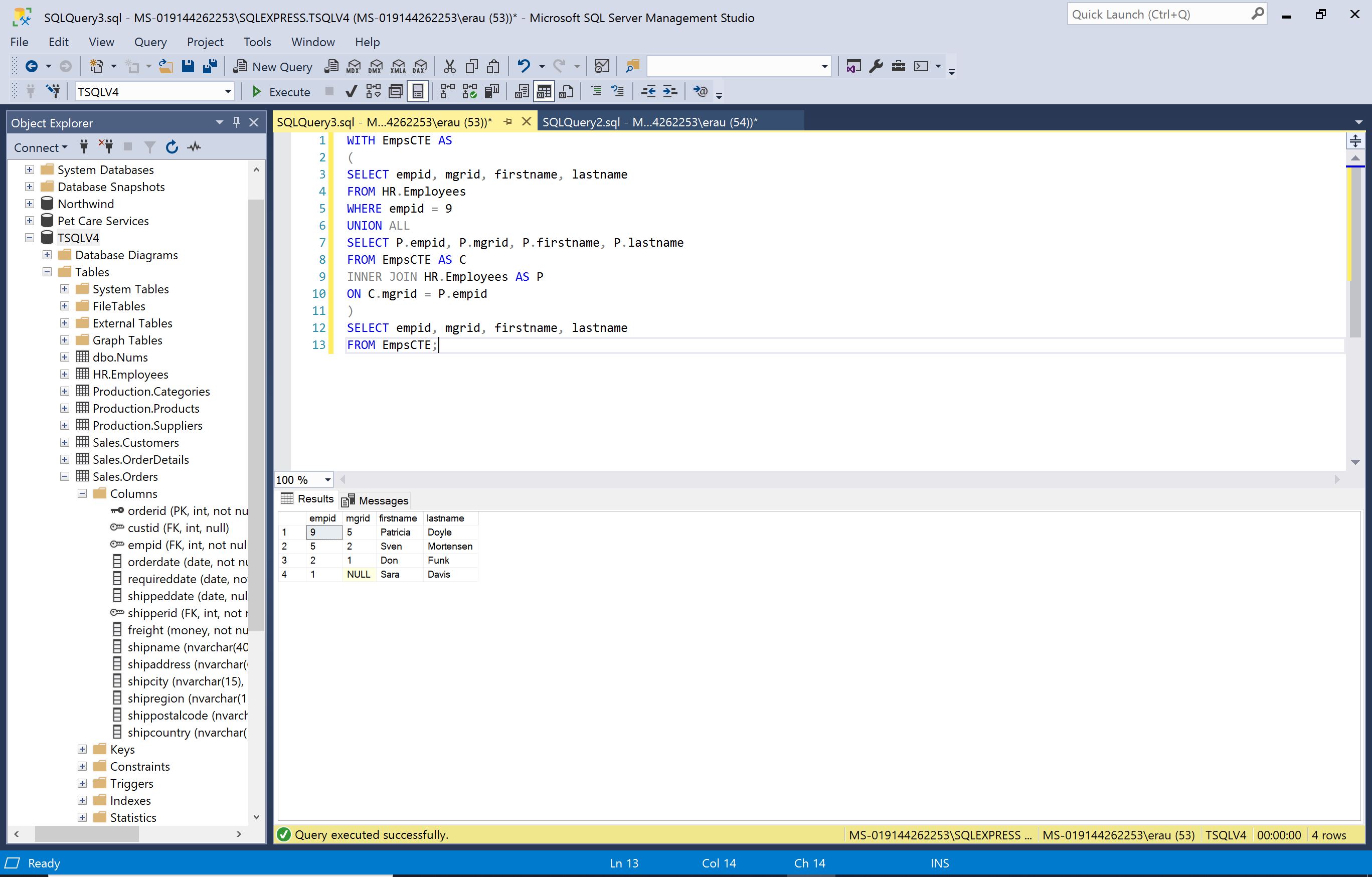
3-1. Calculate a row number for each order based on orderdate, orderid ordering.



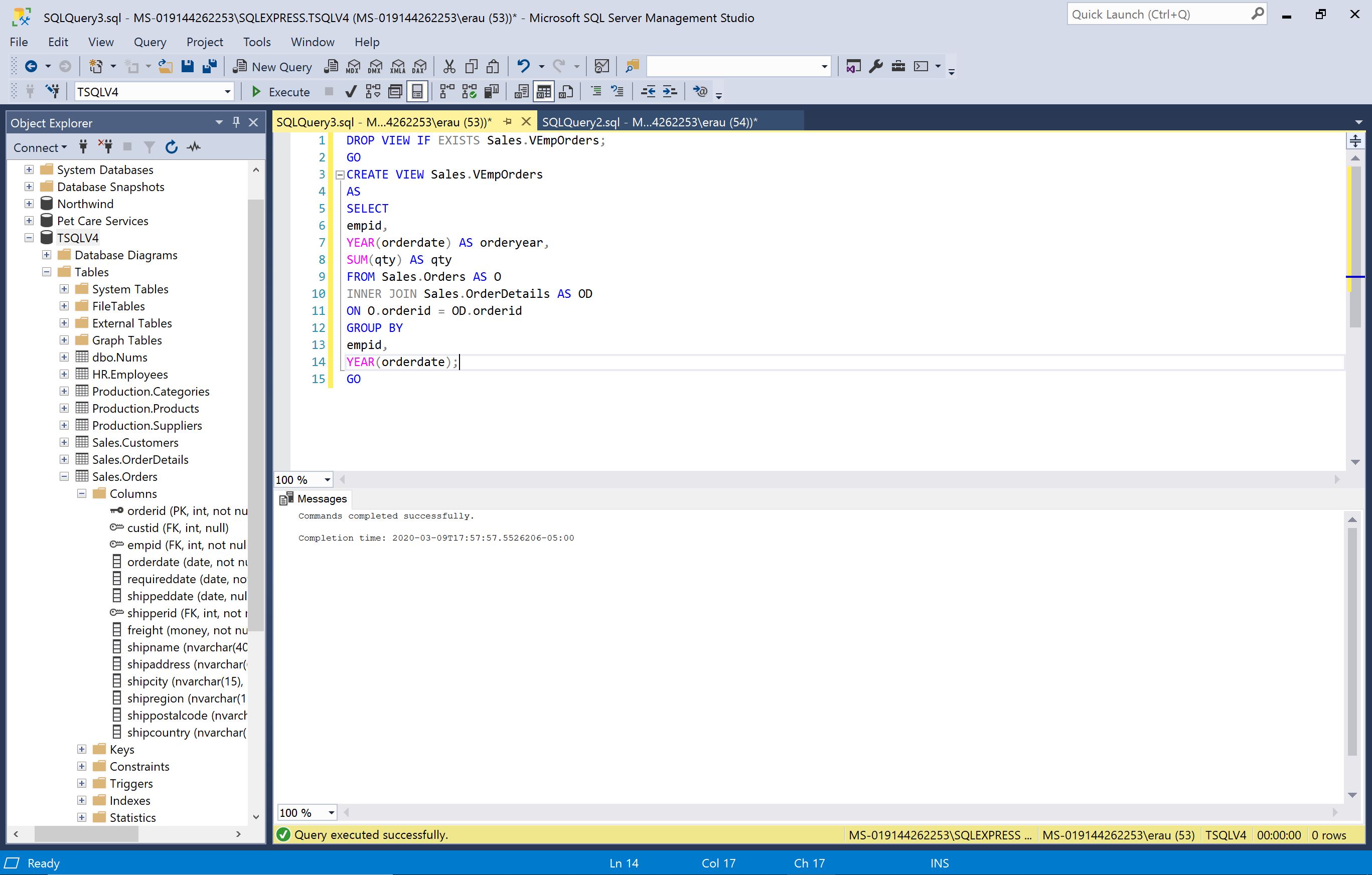
3-2. Return rows with row numbers 11 through 20 based on the row-number definition in Exercise 3-1. Use a CTE to encapsulate the code from Exercise 3-1:



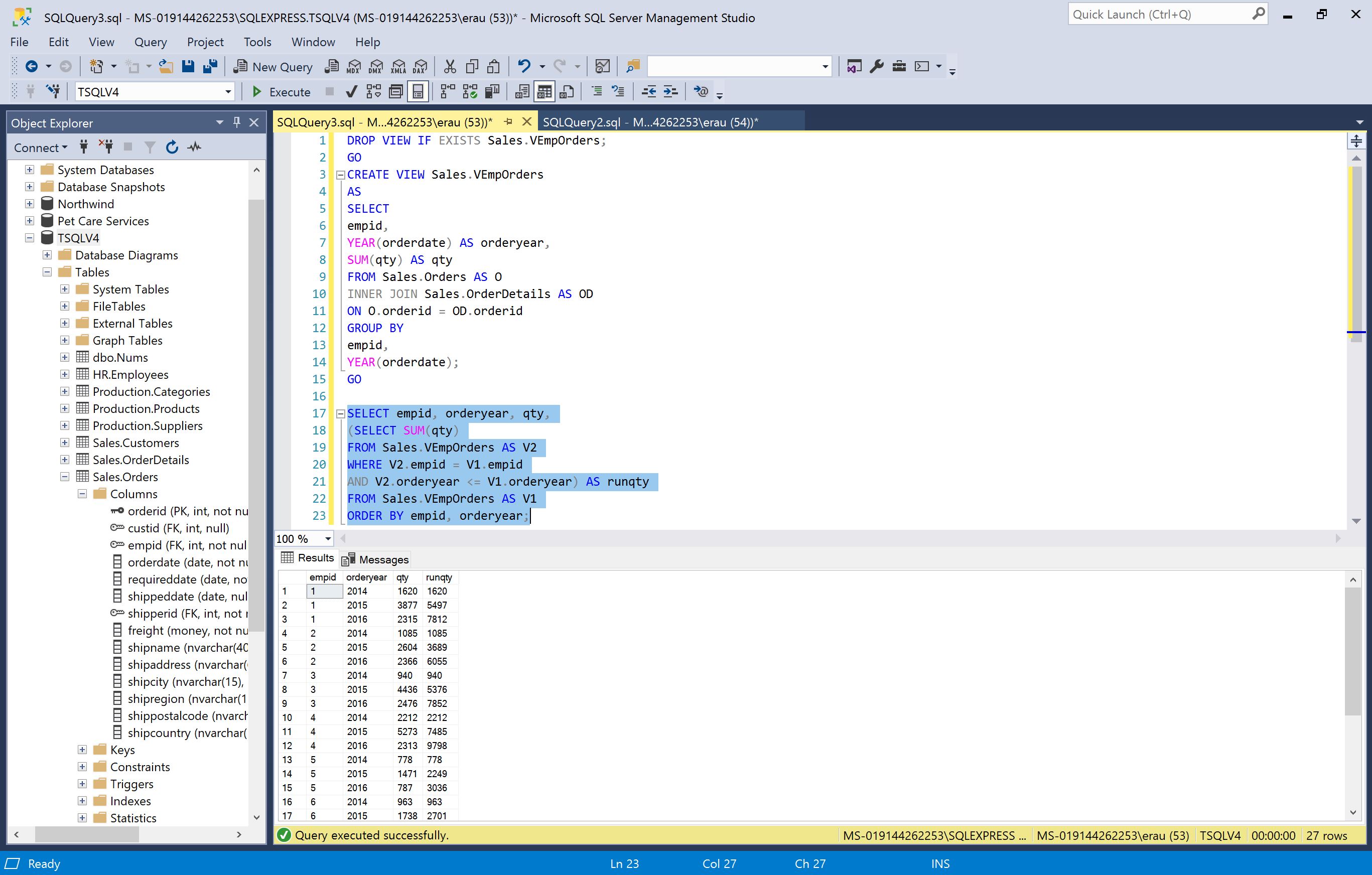
4. Write a solution using a recursive CTE that returns the management chain leading to Patricia Doyle (employee ID 9):



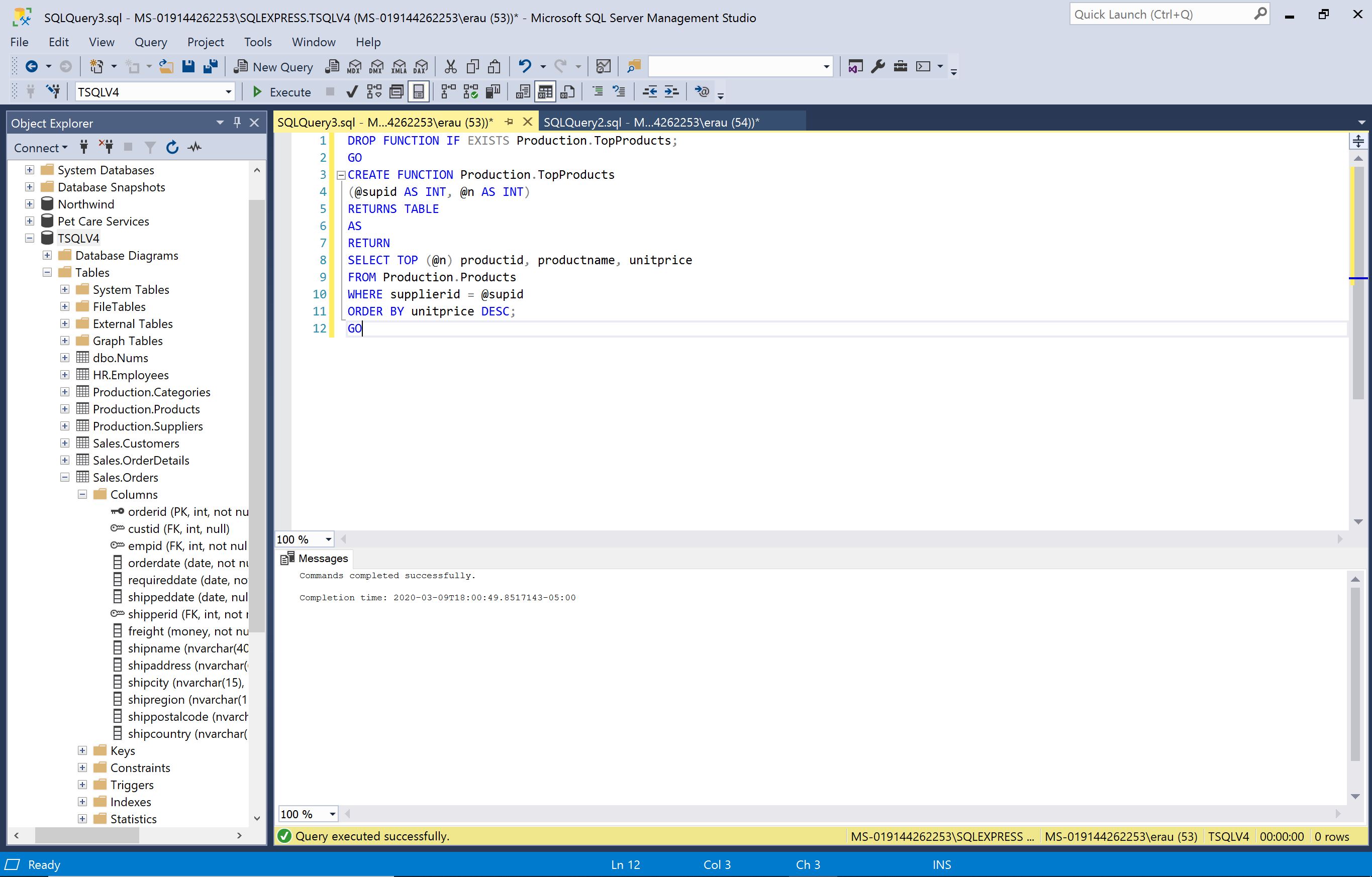
5-1. Create a view that returns the total quantity for each employee and year:



5-2. Write a query against Sales.VEmpOrders that returns the running total quantity for each employee and year.



6-1. Create an inline TVF that accepts as inputs a supplier ID (@supid AS INT) and a requested number of products (@n AS INT).



6-2. Using the CROSS APPLY operator and the function you created in Exercise 6-1, return the two most expensive products for each supplier:

