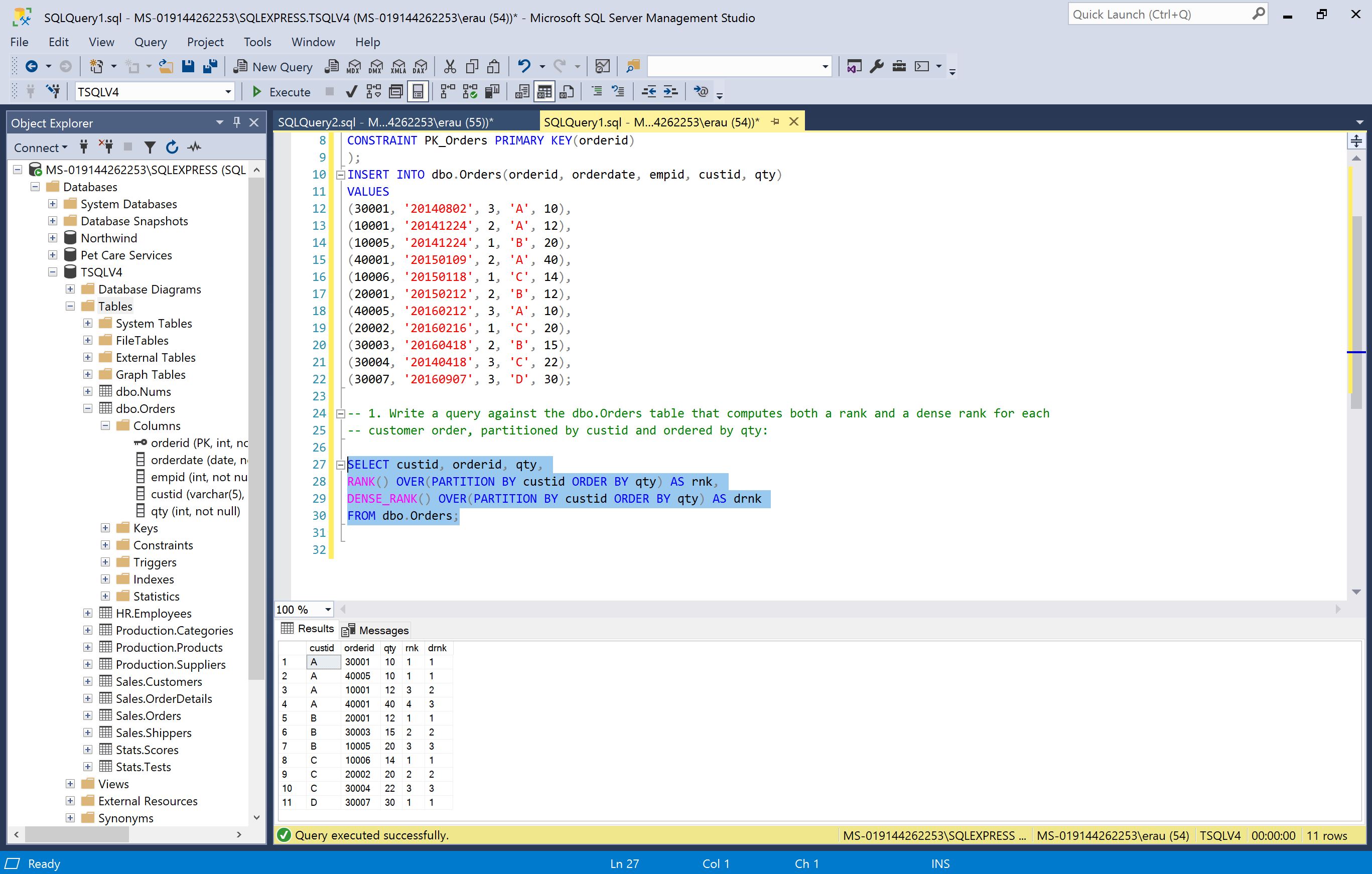
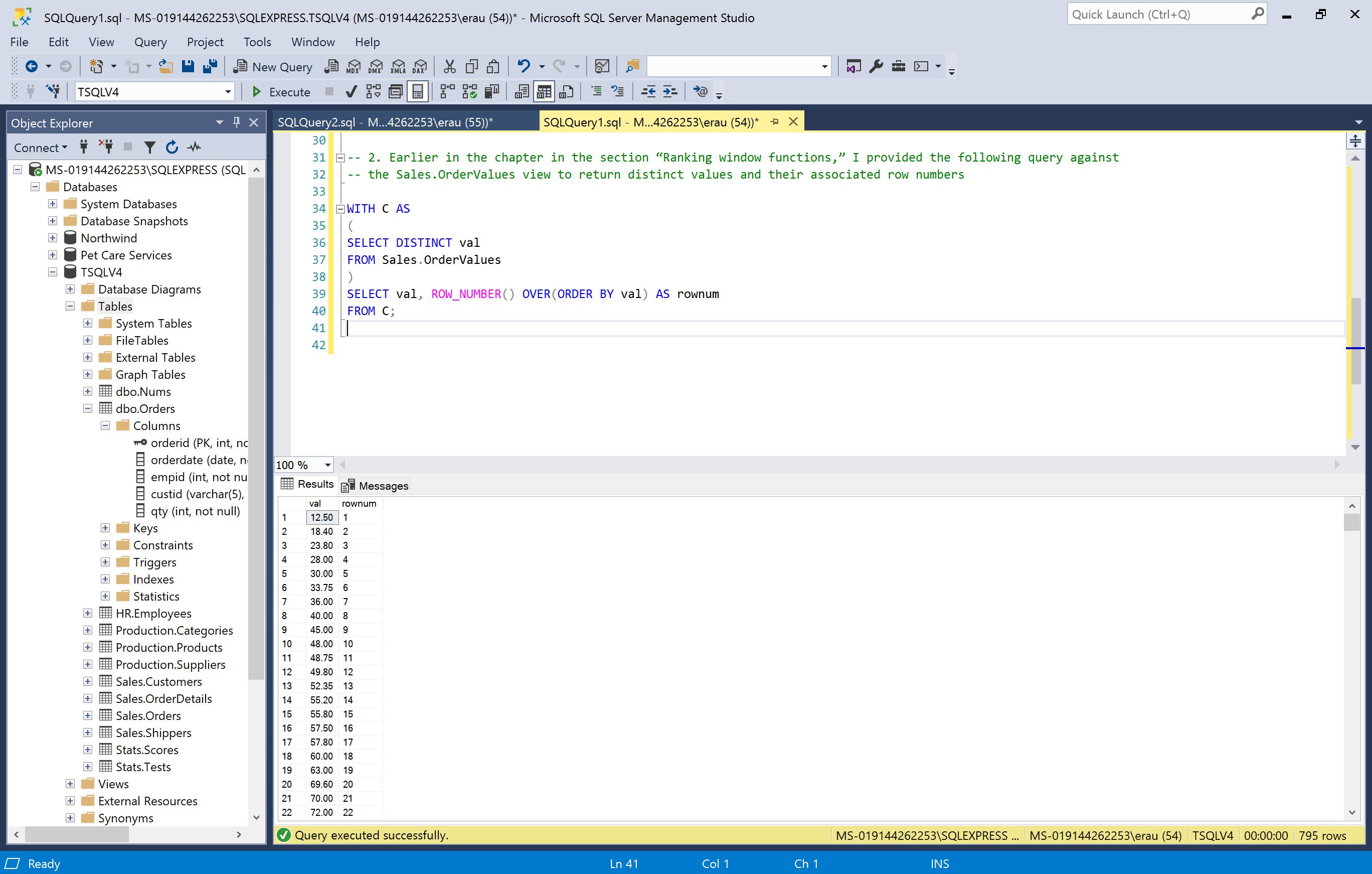
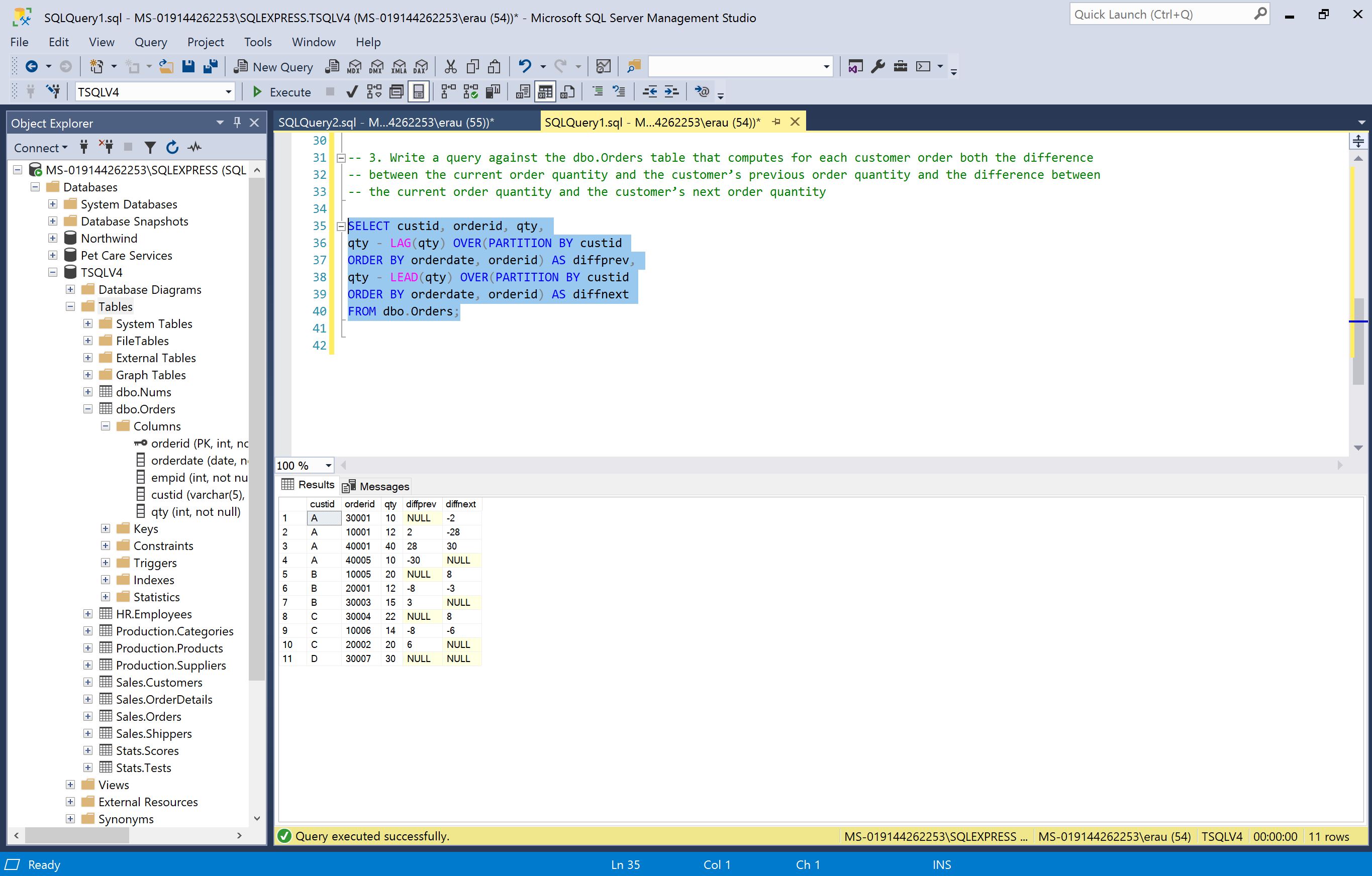
-- 1. Write a query against the dbo.Orders table that computes both a rank and a dense rank for each customer order, partitioned by custid and ordered by qty:



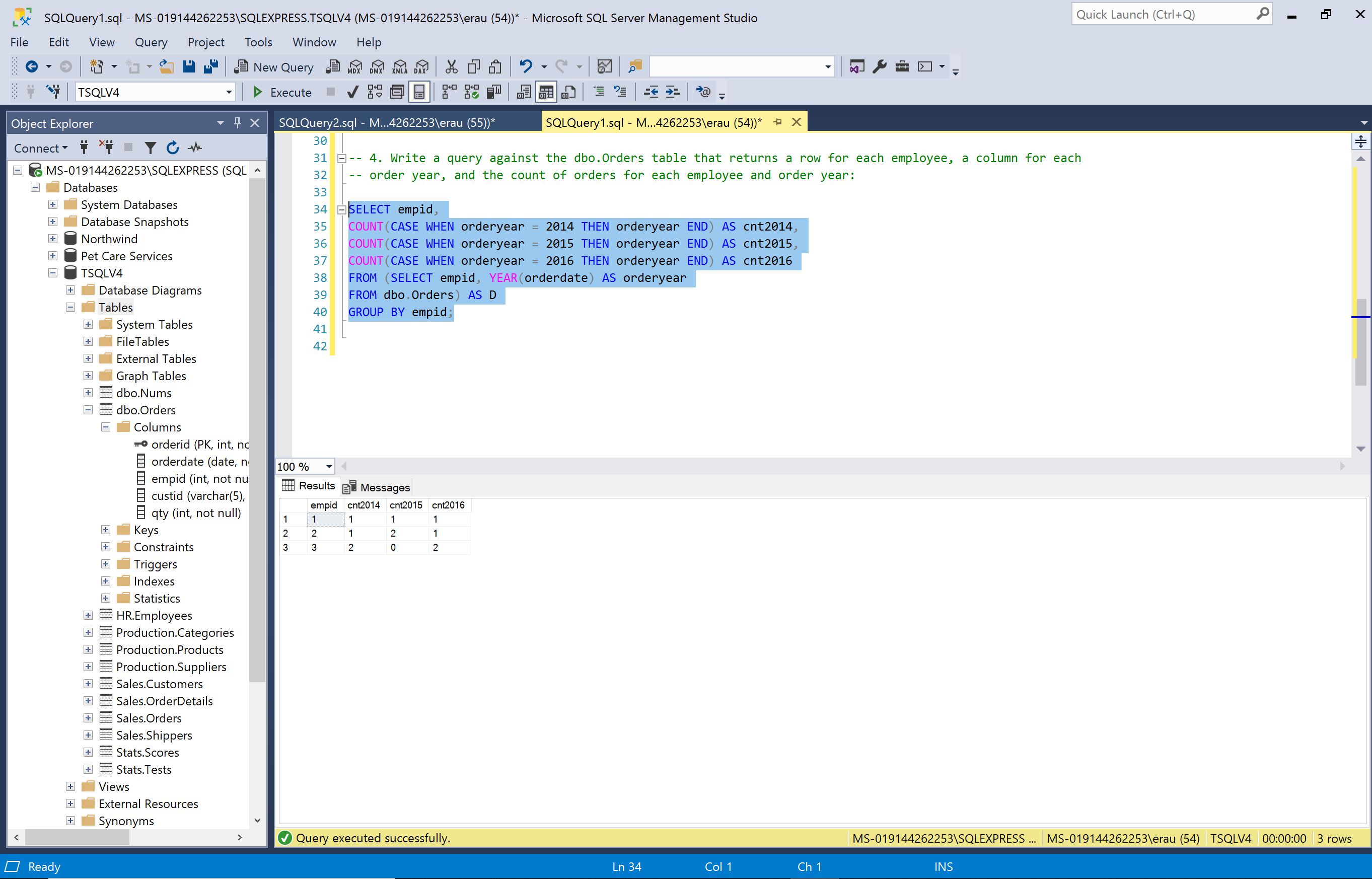
-- 2. Earlier in the chapter in the section “Ranking window functions,” I provided the following query against the Sales.OrderValues view to return distinct values and their associated row numbers



-- 3. Write a query against the dbo.Orders table that computes for each customer order both the difference between the current order quantity and the customer’s previous order quantity and the difference between the current order quantity and the customer’s next order quantity

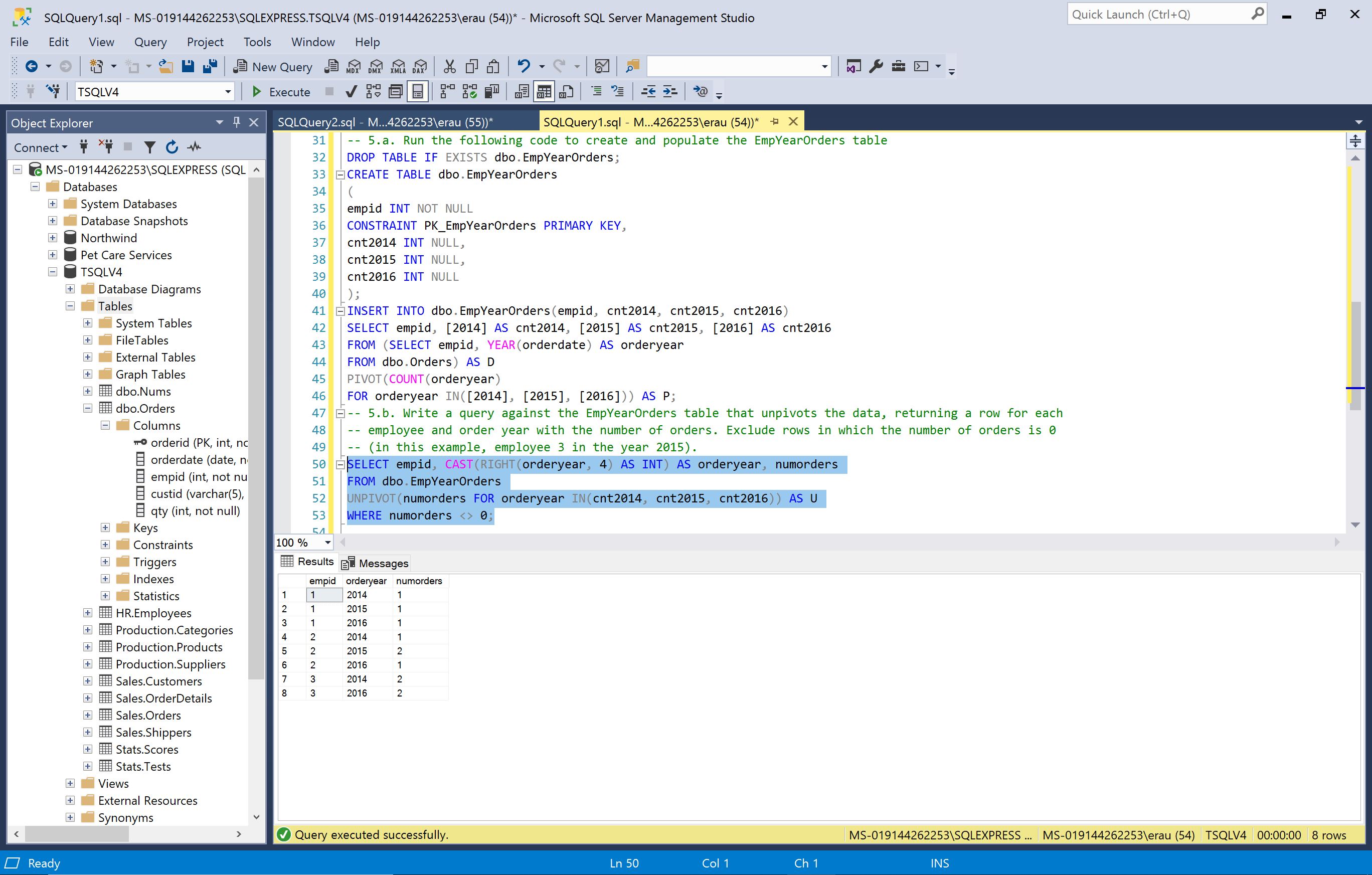


-- 4. Write a query against the dbo.Orders table that returns a row for each employee, a column for each order year, and the count of orders for each employee and order year:



-- 5.a. Run the code to create and populate the EmpYearOrders table

-- 5.b. Write a query against the EmpYearOrders table that unpivots the data, returning a row for each employee and order year with the number of orders. Exclude rows in which the number of orders is 0 (in this example, employee 3 in the year 2015).



-- 6. Write a query against the dbo.Orders table that returns the total quantities for each of the following: (employee, customer, and order year), (employee and order year), and (customer and order year). Include a result column in the output that uniquely identifies the grouping set with which the current row is associated:

