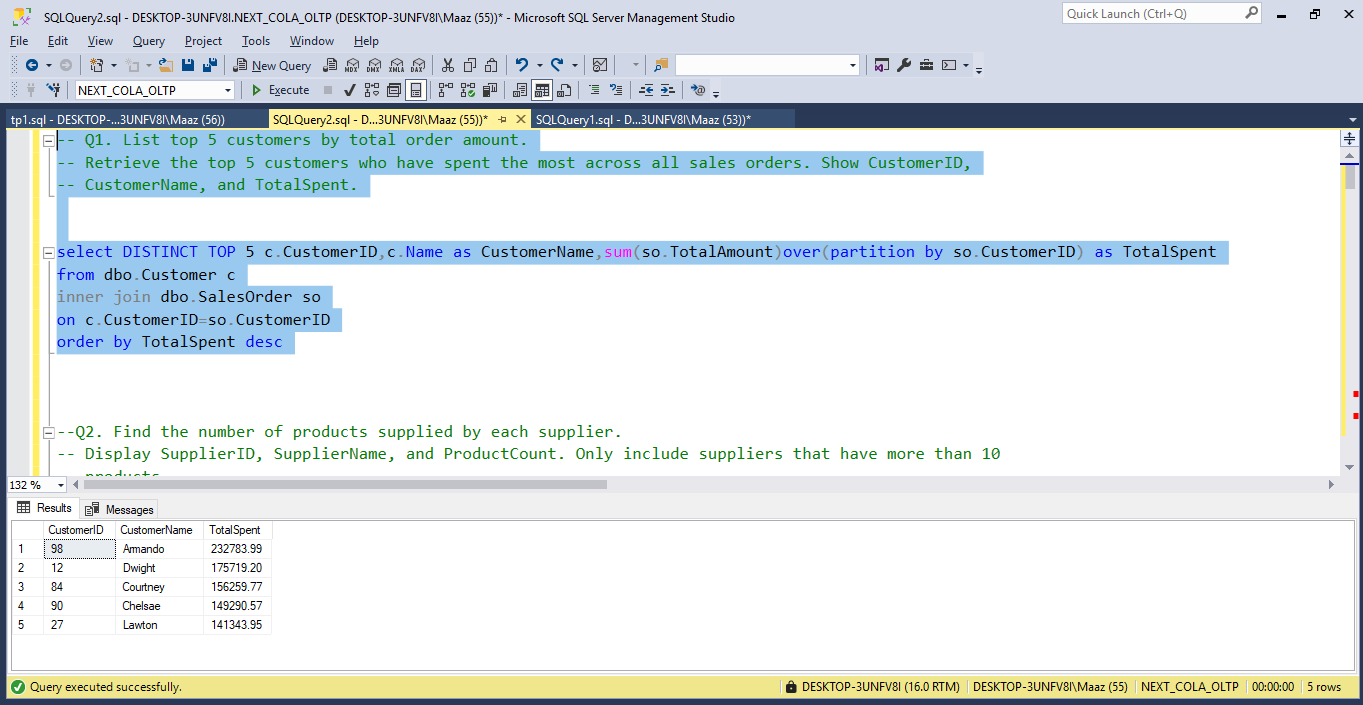
-- Q1. List top 5 customers by total order amount. -- Retrieve the top 5 customers who have spent the most across all sales orders. Show CustomerID, -- CustomerName, and TotalSpent.

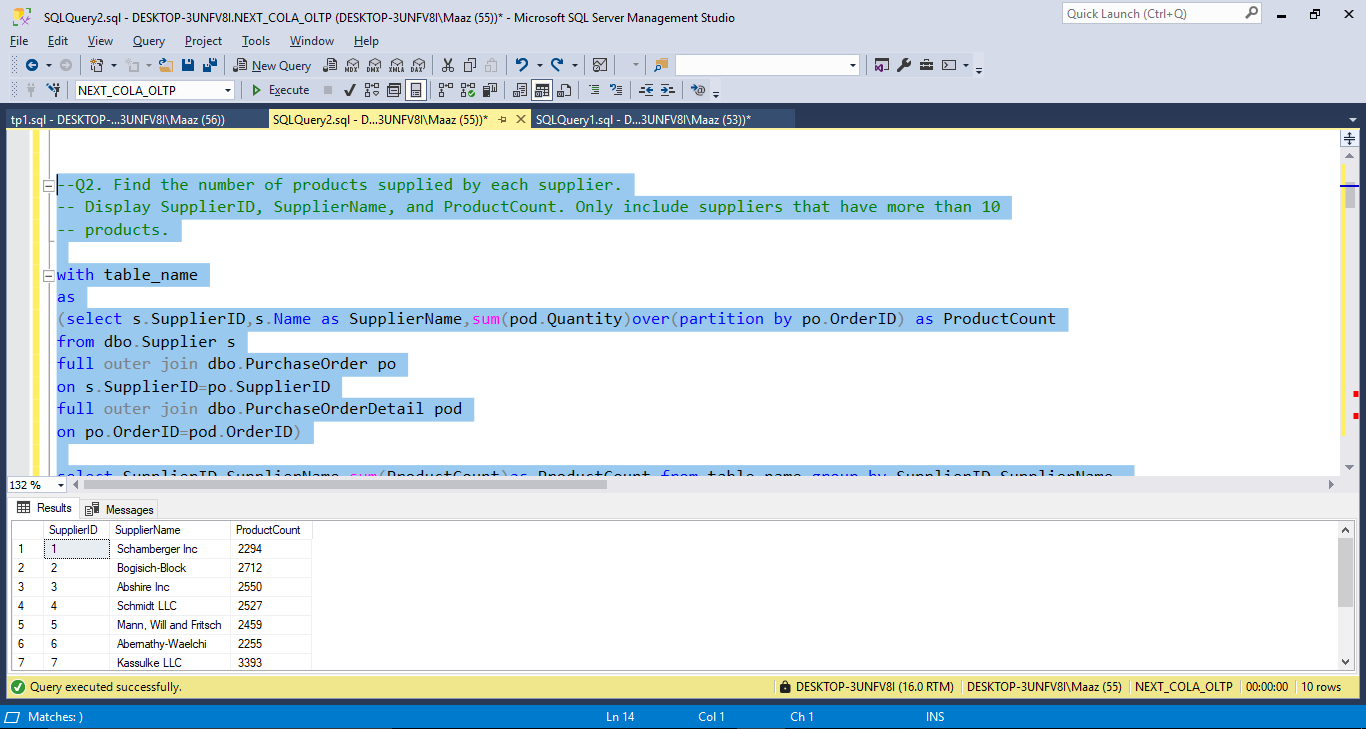
select DISTINCT TOP 5 c.CustomerID,c.Name as CustomerName,sum(so.TotalAmount)over(partition by so.CustomerID) as TotalSpent from dbo.Customer c inner join dbo.SalesOrder so on c.CustomerID=so.CustomerID order by TotalSpent desc



--Q2. Find the number of products supplied by each supplier. -- Display SupplierID, SupplierName, and ProductCount. Only include suppliers that have more than 10 -- products.

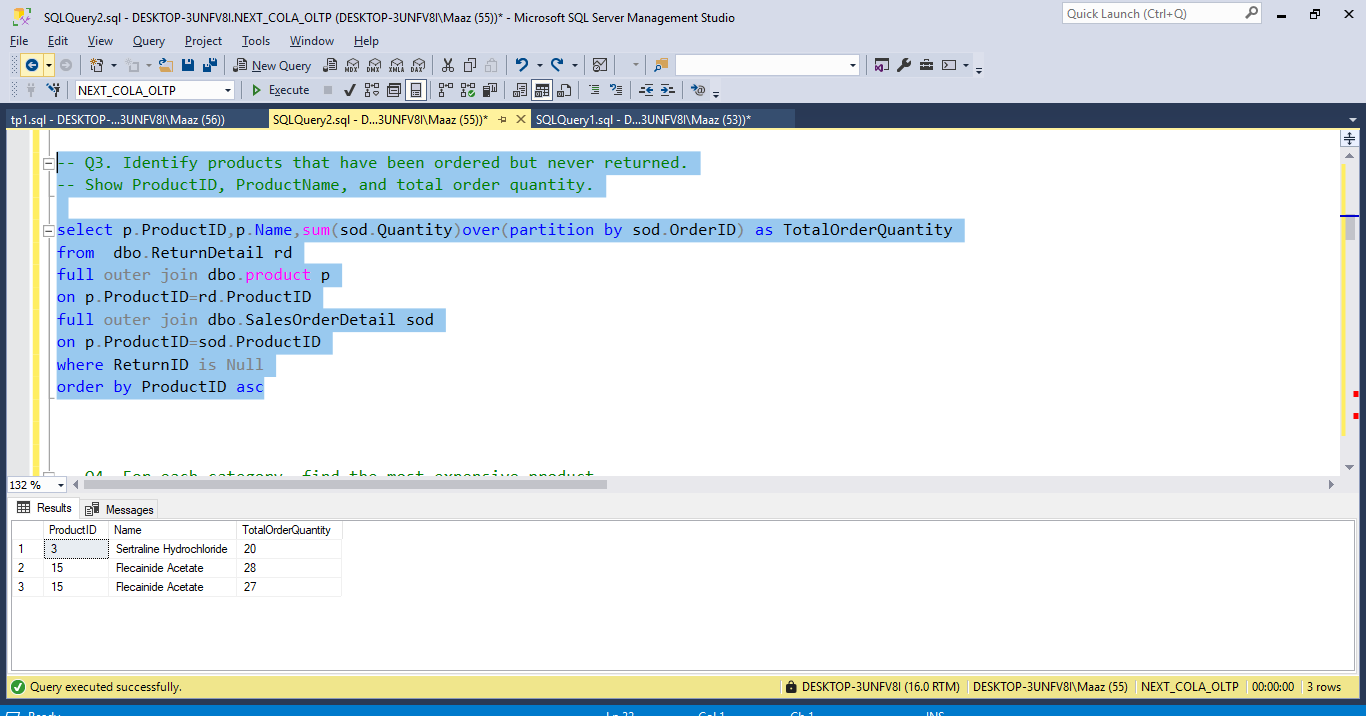
with table\_name as (select s.SupplierID,s.Name as SupplierName,sum(pod.Quantity)over(partition by po.OrderID) as ProductCount from dbo.Supplier s full outer join dbo.PurchaseOrder po on s.SupplierID=po.SupplierID full outer join dbo.PurchaseOrderDetail pod on po.OrderID=pod.OrderID)

select SupplierID,SupplierName,sum(ProductCount)as ProductCount from table\_name group by SupplierID,SupplierName having sum(ProductCount)>10 order by SupplierID asc



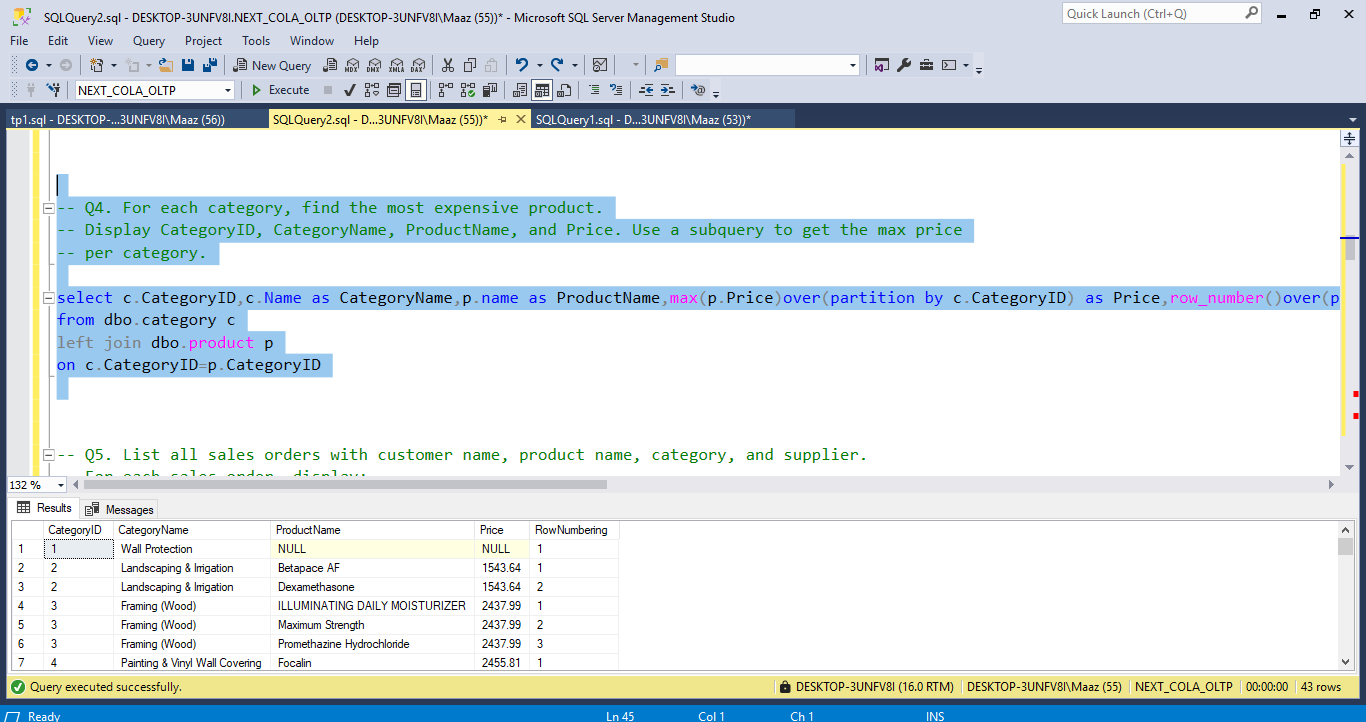
-- Q3. Identify products that have been ordered but never returned. -- Show ProductID, ProductName, and total order quantity.

select p.ProductID,p.Name,sum(sod.Quantity)over(partition by sod.OrderID) as TotalOrderQuantity from dbo.ReturnDetail rd full outer join dbo.product p on p.ProductID=rd.ProductID full outer join dbo.SalesOrderDetail sod on p.ProductID=sod.ProductID where ReturnID is Null order by ProductID asc



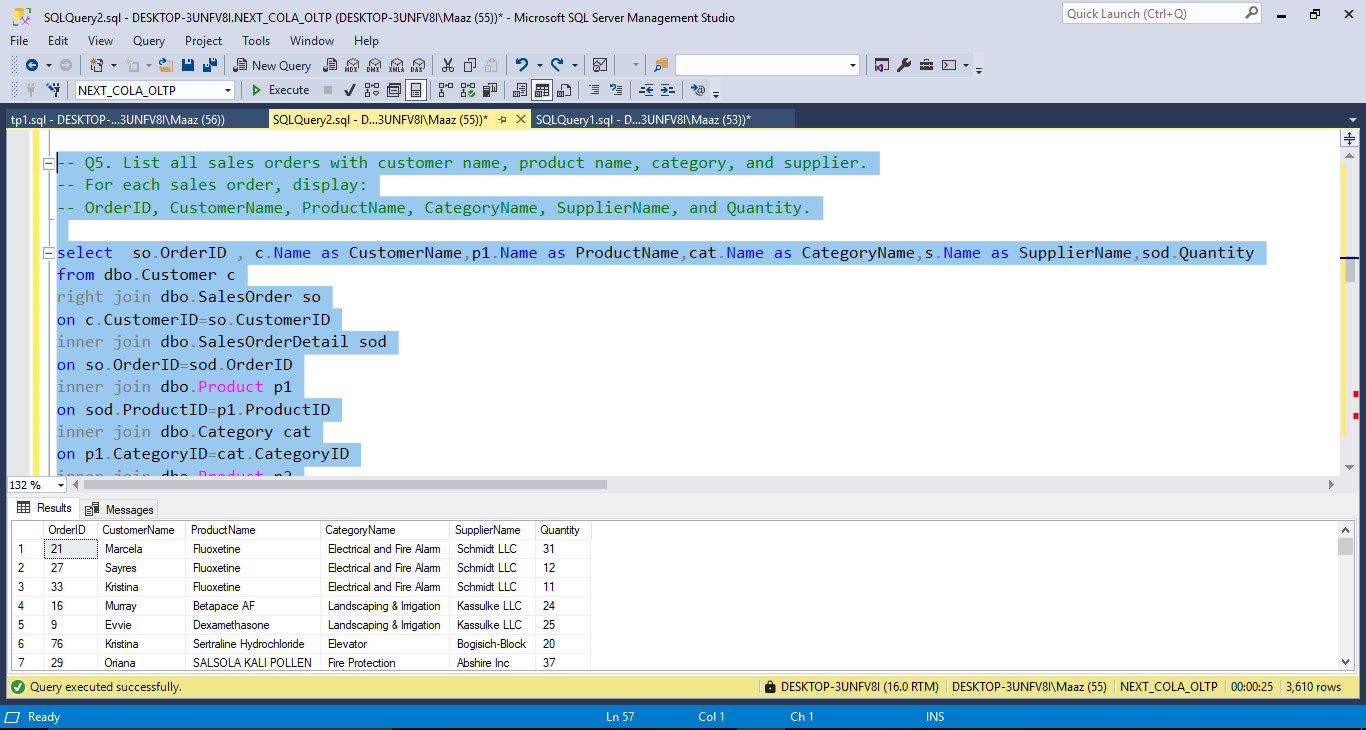
-- Q4. For each category, find the most expensive product. -- Display CategoryID, CategoryName, ProductName, and Price. Use a subquery to get the max price -- per category.

select c.CategoryID,c.Name as CategoryName,p.name as ProductName,max(p.Price)over(partition by c.CategoryID) as Price,row\_number()over(partition by c.CategoryID order by price desc) as RowNumbering from dbo.category c left join dbo.product p on c.CategoryID=p.CategoryID



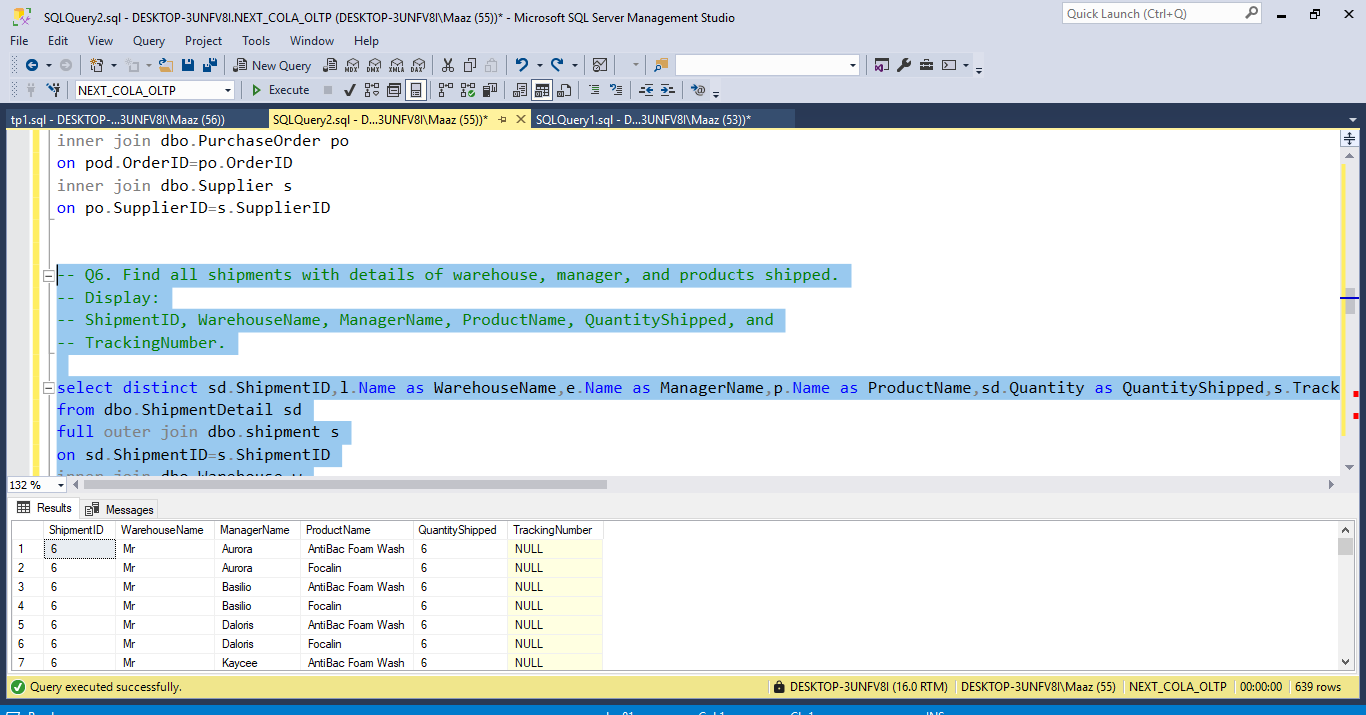
-- Q5. List all sales orders with customer name, product name, category, and supplier. -- For each sales order, display: -- OrderID, CustomerName, ProductName, CategoryName, SupplierName, and Quantity.

select so.OrderID , c.Name as CustomerName,p1.Name as ProductName,cat.Name as CategoryName,s.Name as SupplierName,sod.Quantity from dbo.Customer c right join dbo.SalesOrder so on c.CustomerID=so.CustomerID inner join dbo.SalesOrderDetail sod on so.OrderID=sod.OrderID inner join dbo.Product p1 on sod.ProductID=p1.ProductID inner join dbo.Category cat on p1.CategoryID=cat.CategoryID inner join dbo.Product p2 on cat.CategoryID=p2.CategoryID inner join dbo.PurchaseOrderDetail pod on p2.ProductID=pod.ProductID inner join dbo.PurchaseOrder po on pod.OrderID=po.OrderID inner join dbo.Supplier s on po.SupplierID=s.SupplierID



-- Q6. Find all shipments with details of warehouse, manager, and products shipped. -- Display: -- ShipmentID, WarehouseName, ManagerName, ProductName, QuantityShipped, and -- TrackingNumber.

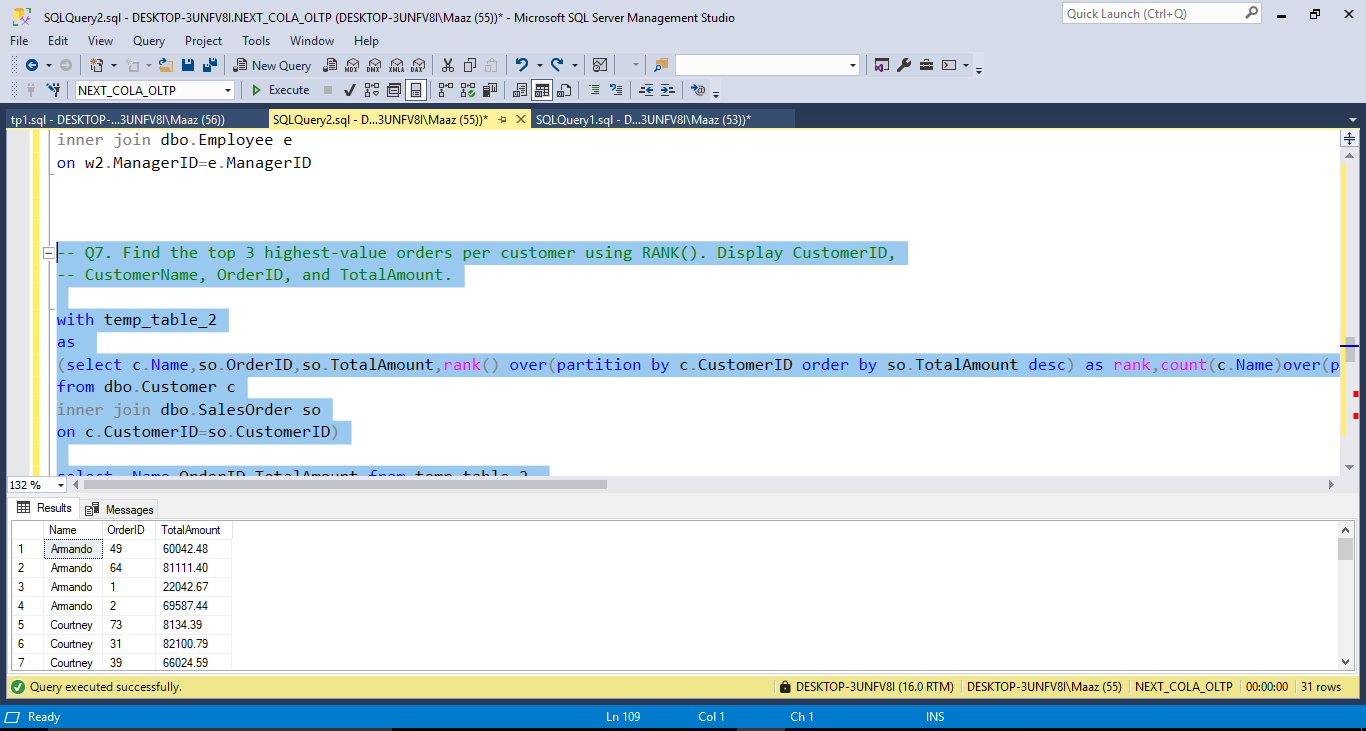
select distinct sd.ShipmentID,l.Name as WarehouseName,e.Name as ManagerName,p.Name as ProductName,sd.Quantity as QuantityShipped,s.TrackingNumber from dbo.ShipmentDetail sd full outer join dbo.shipment s on sd.ShipmentID=s.ShipmentID inner join dbo.Warehouse w on s.WarehouseID=w.WarehouseID inner join dbo.Inventory i1 on w.WarehouseID=i1.WarehouseID inner join dbo.Product p on i1.ProductID=p.ProductID inner join dbo.Inventory i2 on i2.ProductID=p.ProductID inner join dbo.Warehouse w1 on w1.WarehouseID=i2.WarehouseID inner join dbo.Location l on w1.LocationID=l.LocationID inner join dbo.Warehouse w2 on l.LocationID=w2.LocationID inner join dbo.Employee e on w2.ManagerID=e.ManagerID



-- Q7. Find the top 3 highest-value orders per customer using RANK(). Display CustomerID, -- CustomerName, OrderID, and TotalAmount.

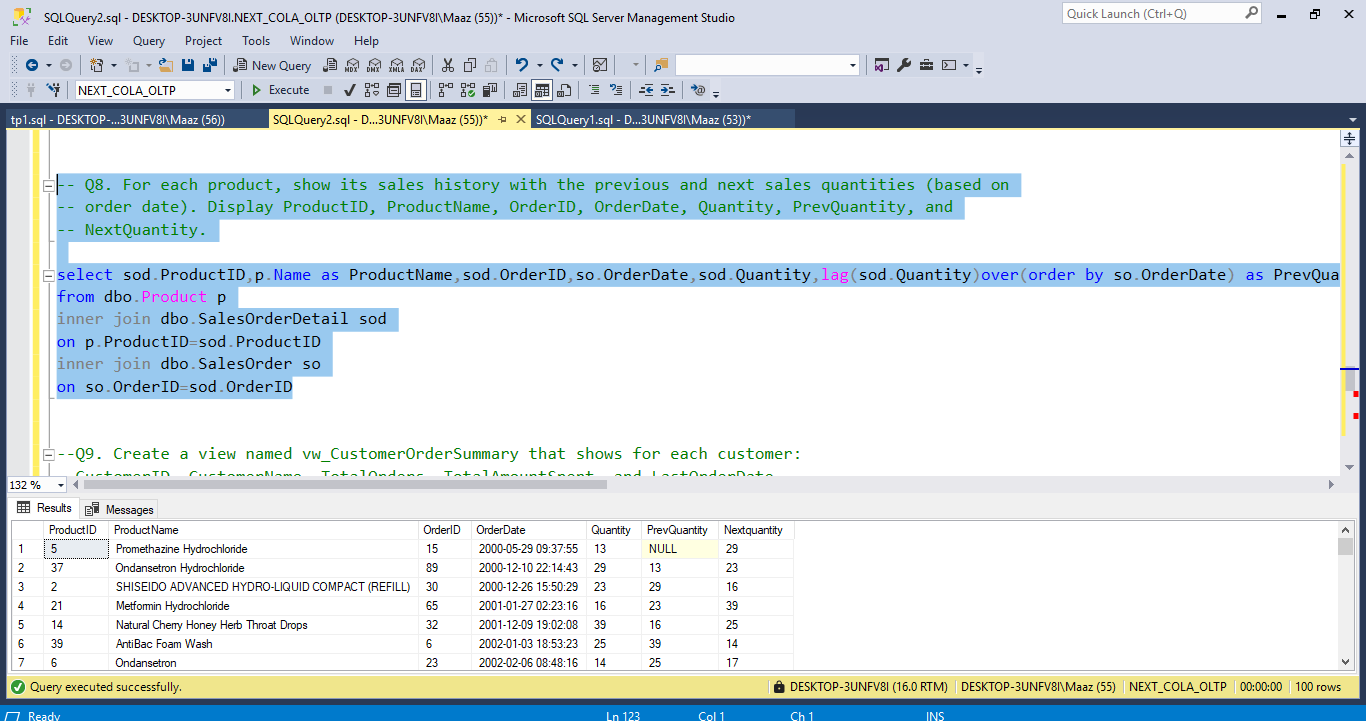
with temp\_table\_2 as (select c.Name,so.OrderID,so.TotalAmount,rank() over(partition by c.CustomerID order by so.TotalAmount desc) as rank,count(c.Name)over(partition by c.Name) as Count from dbo.Customer c inner join dbo.SalesOrder so on c.CustomerID=so.CustomerID)

select Name,OrderID,TotalAmount from temp\_table\_2 where Count>=3



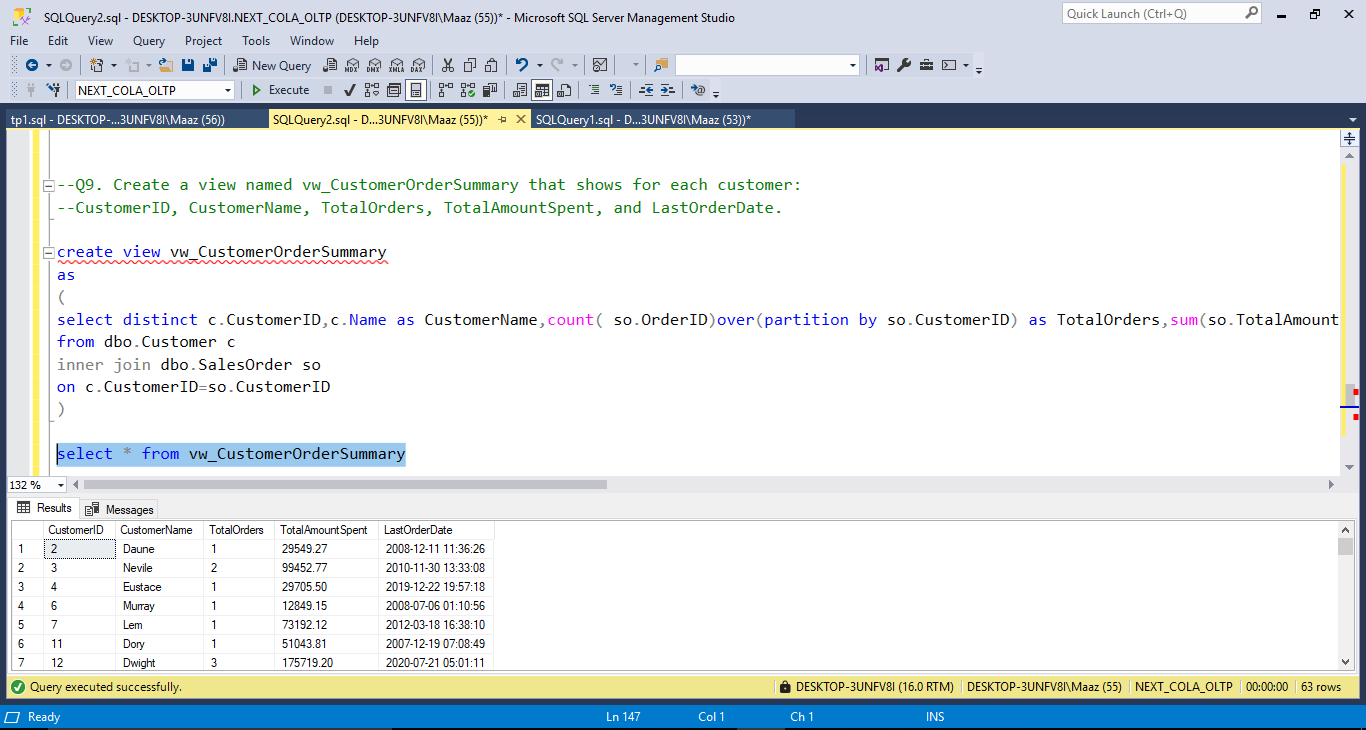
-- Q8. For each product, show its sales history with the previous and next sales quantities (based on -- order date). Display ProductID, ProductName, OrderID, OrderDate, Quantity, PrevQuantity, and -- NextQuantity.

select sod.ProductID,p.Name as ProductName,sod.OrderID,so.OrderDate,sod.Quantity,lag(sod.Quantity)over(order by so.OrderDate) as PrevQuantity,lead(sod.Quantity)over(order by so.OrderDate) as Nextquantity from dbo.Product p inner join dbo.SalesOrderDetail sod on p.ProductID=sod.ProductID inner join dbo.SalesOrder so on so.OrderID=sod.OrderID



--Q9. Create a view named vw\_CustomerOrderSummary that shows for each customer: --CustomerID, CustomerName, TotalOrders, TotalAmountSpent, and LastOrderDate.

create view vw\_CustomerOrderSummary as ( select distinct c.CustomerID,c.Name as CustomerName,count( so.OrderID)over(partition by so.CustomerID) as TotalOrders,sum(so.TotalAmount)over(partition by so.CustomerID) as TotalAmountSpent,FIRST\_VALUE(so.OrderDate)over(partition by so.CustomerID order by so.OrderDate desc) as LastOrderDate from dbo.Customer c inner join dbo.SalesOrder so on c.CustomerID=so.CustomerID )



--Q10. Write a stored procedure sp\_GetSupplierSales that takes a SupplierID as input and returns the --total sales amount for all products supplied by that supplier.

create procedure sp\_GetSupplierSales @SupplierID int as begin select SupplierID,sum(TotalAmount) as TotalSalesAmount from dbo.PurchaseOrder group by SupplierID having SupplierId=@SupplierID end

exec sp\_GetSupplierSales @SupplierID=3;

