---------------------------------------------------------------------

-- LAB 11

--

-- Exercise 1

---------------------------------------------------------------------

USE TSQL;

Go

---------------------------------------------------------------------

-- Task 1

--

-- Write a SELECT statement to return the productid, productname, supplierid, unitprice,

-- and discontinued columns from the Production.Products table. Filter the results to

-- include only products that belong to the category Beverages (categoryid equals 1).

-- Observe and compare the results that you got with the desired results shown in the file 52 - Lab Exercise 1 - Task 1 Result.txt.

-- Modify the T-SQL code to include the following supplied T-SQL statement.

-- Put this statement before the SELECT clause:

-- Execute the complete T-SQL statement. This will create an object view named

-- ProductBeverages under the Production schema.

---------------------------------------------------------------------

Select pp.productid,pp.productname,pp.supplierid,pp.unitprice,pp.discontinued

From Production.Products pp

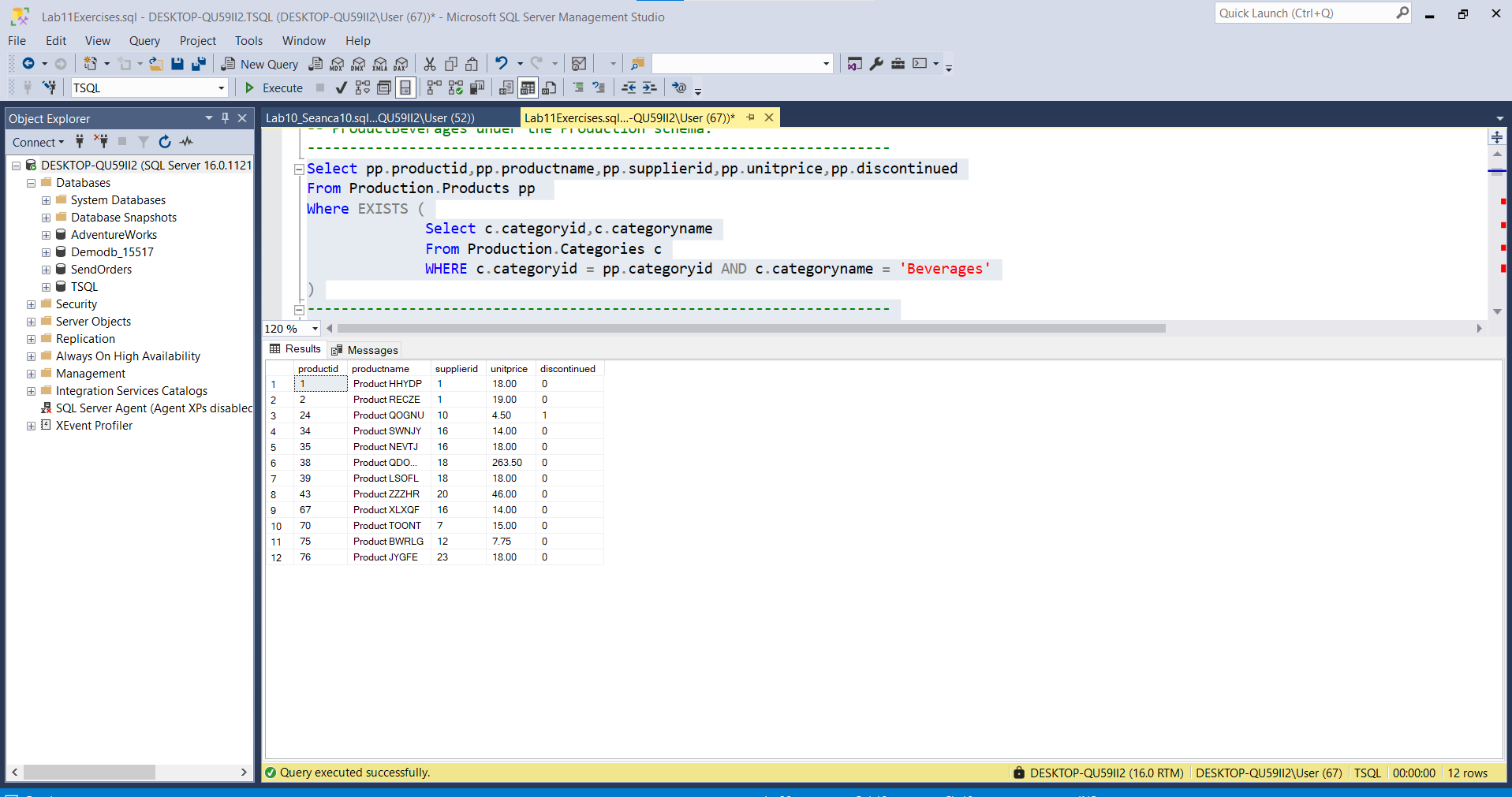
Where EXISTS (

Select c.categoryid,c.categoryname

From Production.Categories c

WHERE c.categoryid = pp.categoryid AND c.categoryname = 'Beverages'

)



-- Task 2

--

-- Write a SELECT statement to return the productid and productname columns from the Production.ProductsBeverages view.

-- Filter the results to include only products where supplierid equals 1.

-- Execute the written statement and compare the results that you got with

-- the desired results shown in the file 53 - Lab Exercise 1 - Task 2 Result.txt.

---------------------------------------------------------------------

Create VIEW [Production].[ProductsBeveragess]

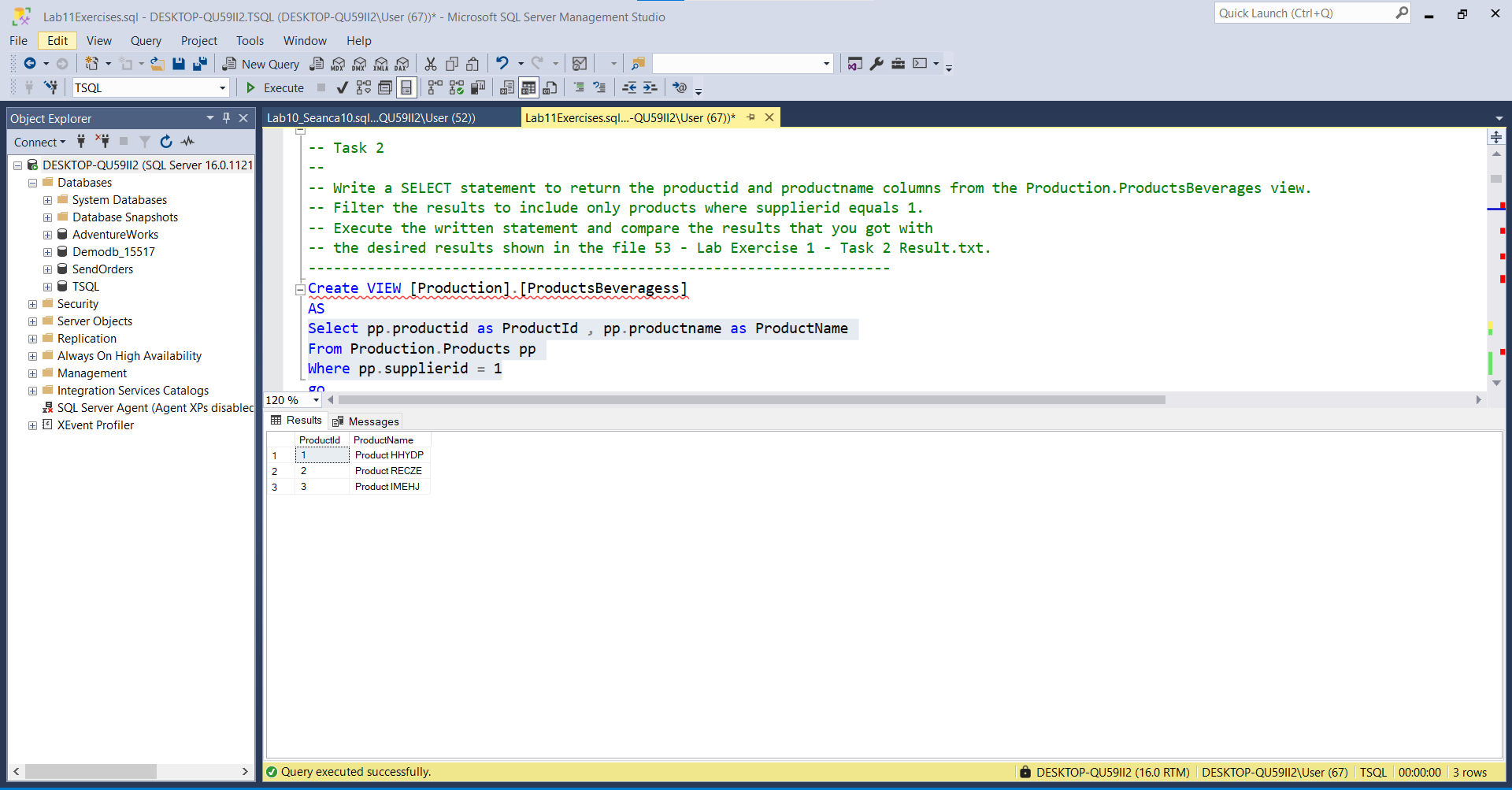
AS

Select pp.productid as ProductId , pp.productname as ProductName

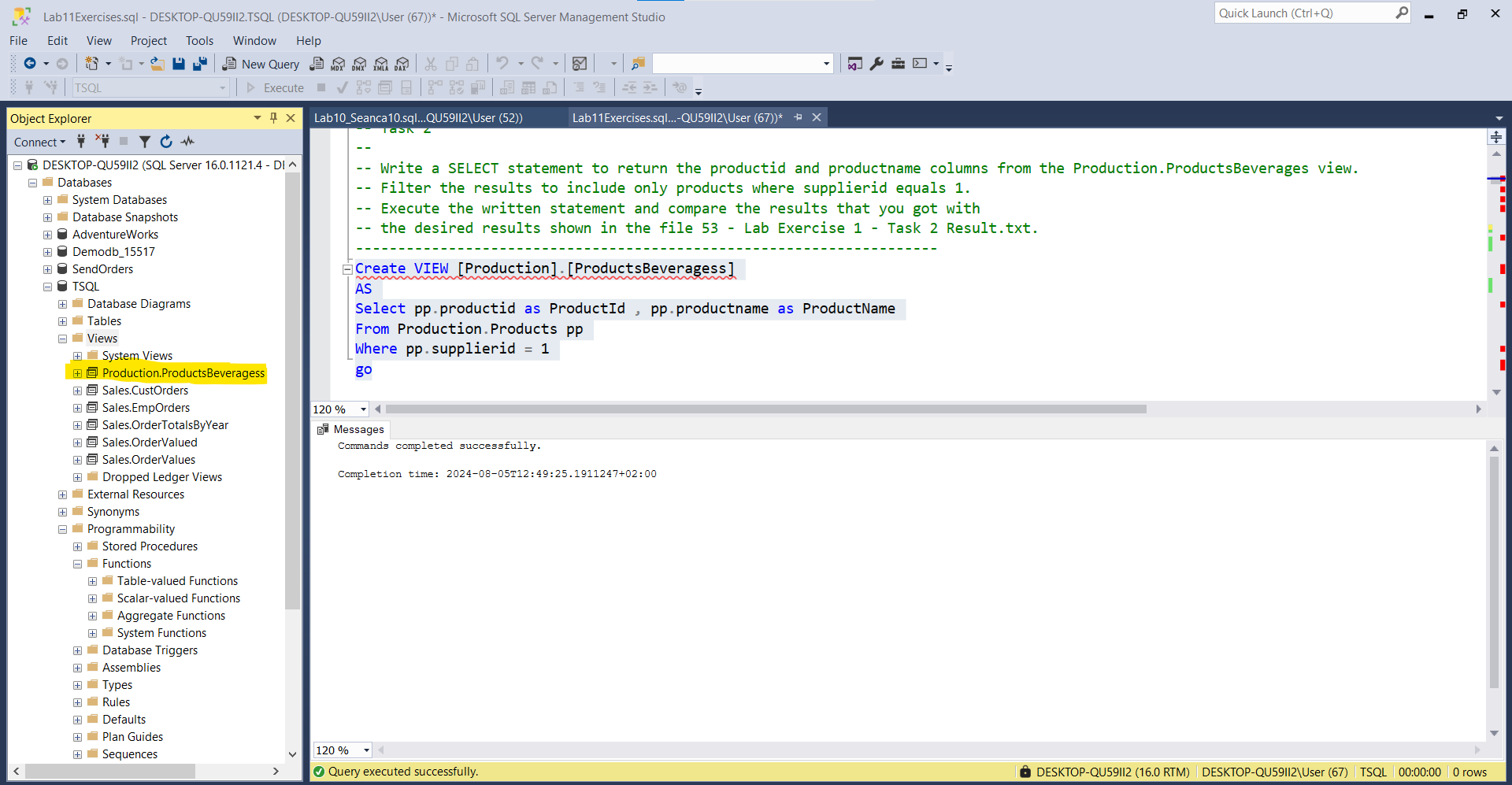
From Production.Products pp

Where pp.supplierid = 1

go



--Ekzekutimi me sukses i VIEWS si më poshtë:



-- Task 3

--

-- The IT department has written a T-SQL statement that adds an ORDER BY clause to the view created in task 1.

--

-- Execute the provided code. What happened? What is the error message? Why did the query fail?

--

-- Modify the supplied T-SQL statement by including the TOP (100) PERCENT option. The query should look like this:

-- Execute the modified T-SQL statement. By applying the needed changes, you have altered the existing view.

-- Notice that you are still using still use the ORDER BY clause.

--

-- If you write a query against the modified Production.ProductsBeverages view,

-- will it be guaranteed that the retrieved rows will be sorted by productname? Please explain.

---------------------------------------------------------------------

ALTER VIEW Production.ProductsBeverages

AS

SELECT top(100) percent p.productid, p.productname, p.supplierid, p.unitprice,p.discontinued

FROM Production.Products p

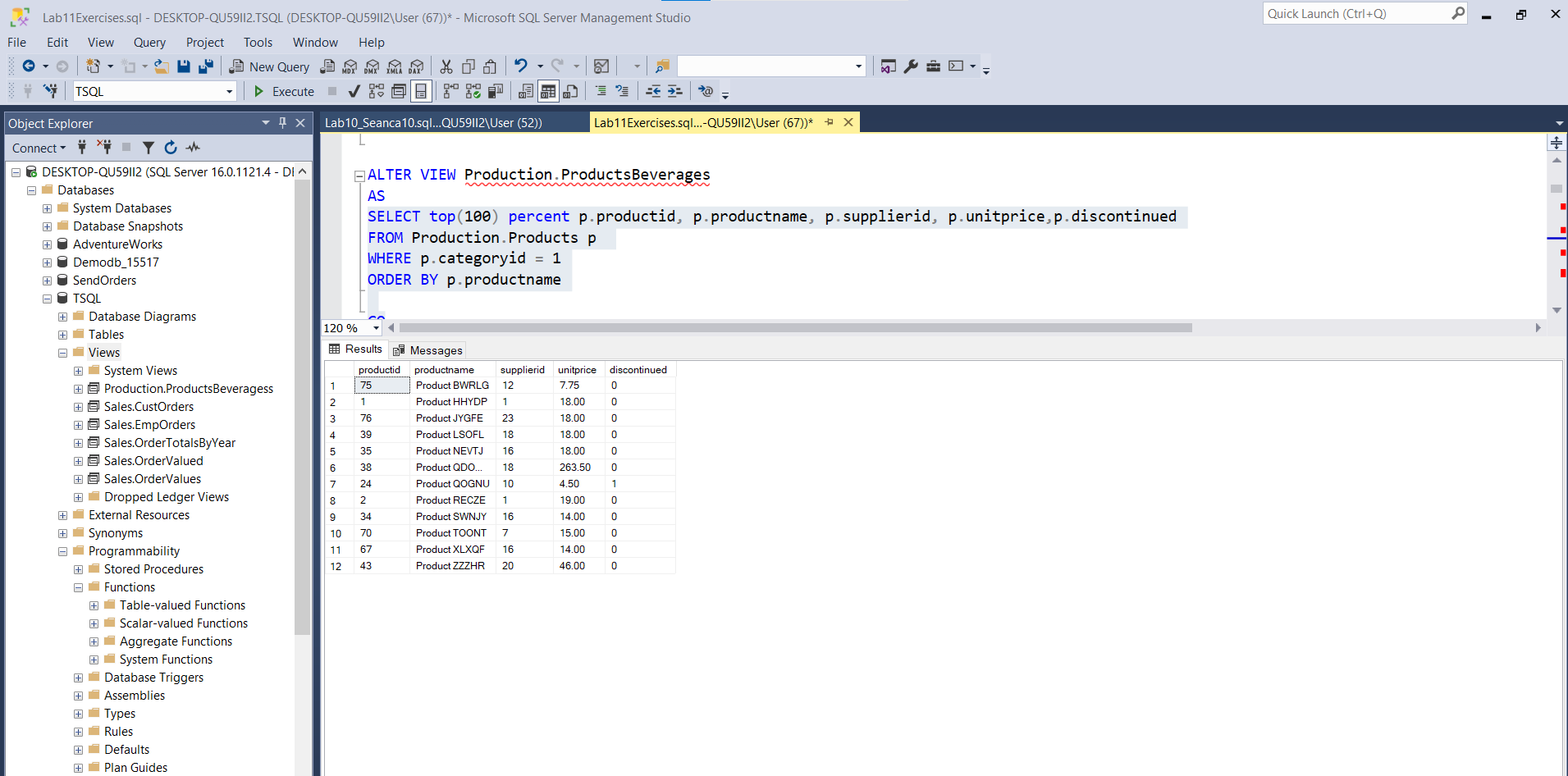
WHERE p.categoryid = 1

ORDER BY p.productname

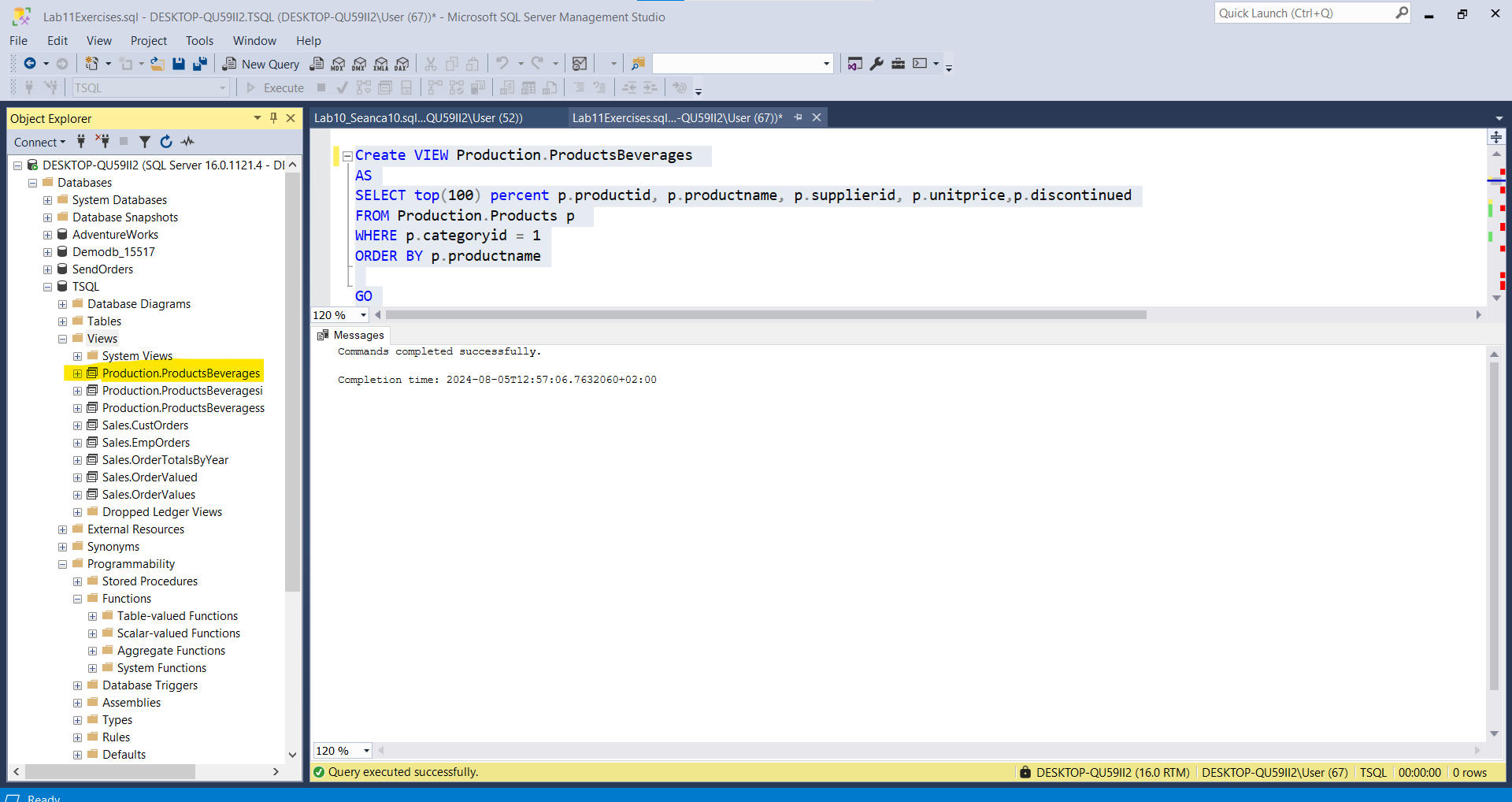
GO

--\*\*Order By clause can be used only if we have used in parallel 'TOP' and/or 'OFFSET/FETCH'

--Ekzekutimi me sukses i querisë dhe views



Ekzekutimi me sukses i VIEWS :



---------------------------------------------------------------------

-- Task 4

--

-- The IT department has written a T-SQL statement that adds an additional calculated column to the view created in task 1.

--

-- Execute the provided query. What happened? What is the error message? Why did the query fail?

-- Apply the changes needed to get the T-SQL statement to execute properly.

---------------------------------------------------------------------

Create VIEW Production.ProductsBeveragesi AS

SELECT

p.productid, p.productname, p.supplierid, p.unitprice, p.discontinued,

productStatus\_by\_unitprice=(CASE WHEN p.unitprice > 100. THEN N'high' ELSE N'normal' END)

FROM Production.Products p

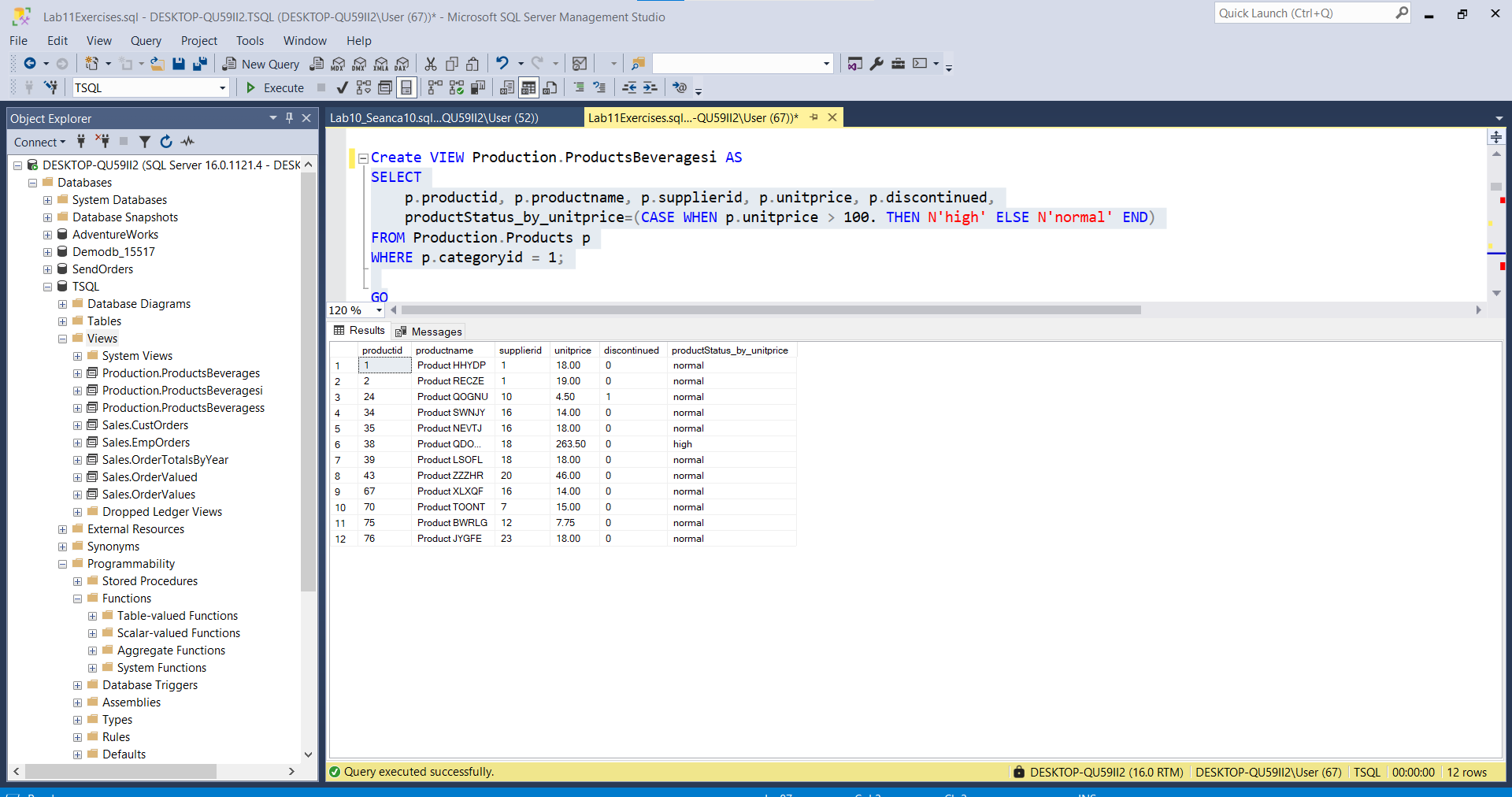
WHERE p.categoryid = 1;

GO

--Ekzekutimi me sukses i VIEW



Ekzekutimi me sukses i querisë :



---------------------------------------------------------------------

-- Task 5

-- Remove the created view by executing the provided T-SQL statement.

-- Execute this code exactly as written inside a query window.

---------------------------------------------------------------------

IF OBJECT\_ID(N'Production.ProductsBeverages', N'V') IS NOT NULL

DROP VIEW Production.ProductsBeverages;

--------------------------------------------------------------------

Commands completed successfully.

Completion time: 2024-08-04T15:00:49.7958250+02:00---------------------------------------------------------------------

-- LAB 11

--

-- Exercise 2

---------------------------------------------------------------------

USE TSQL;

GO

---------------------------------------------------------------------

-- Task 1

--

-- Write a SELECT statement against a derived table and retrieve the productid and productname columns.

-- Filter the results to include only the rows in which the pricetype column value is equal to high.

-- Use the SELECT statement from exercise 1, task 4 as the inner query that defines the derived table.

-- Do not forget to use an alias for the derived table. (You can use the alias p.)

-- Execute the written statement and compare the results that you got with the desired results

-- shown in the file 62 - Lab Exercise 2 - Task 1 Result.txt.

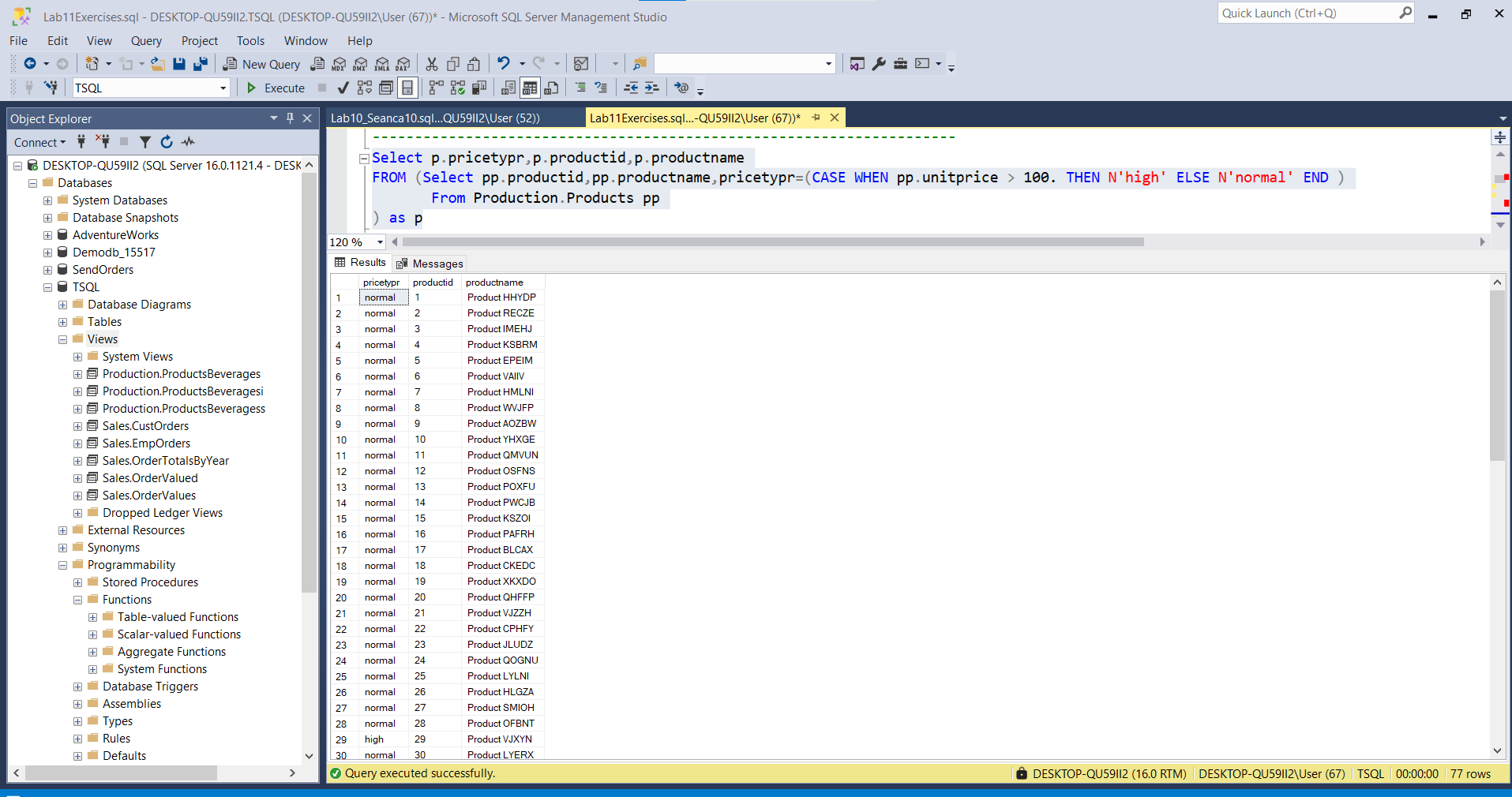
---------------------------------------------------------------------

Select p.pricetypr,p.productid,p.productname

FROM (Select pp.productid,pp.productname,pricetypr=(CASE WHEN pp.unitprice > 100. THEN N'high' ELSE N'normal' END )

From Production.Products pp

) as p



-- Task 2

--

-- Write a SELECT statement to retrieve the custid column and two calculated columns:

-- totalsalesamount, which returns the total sales amount per customer, and avgsalesamount,

-- which returns the average sales amount of orders per customer. To correctly calculate

-- the average sales amount of orders per customer, you will first have to calculate the total sales amount per order.

-- You can do so by defining a derived table based on a query that joins the Sales.Orders

-- and Sales.OrderDetails tables.You can use the custid and orderid columns

-- from the Sales.Orders table and the qty and unitprice columns from the Sales.OrderDetails table.

-- Execute the written statement and compare the results that you got with the recommended result

-- shown in the file 63 - Lab Exercise 2 - Task 2 Result.txt.

---------------------------------------------------------------------

Select orderss.custid,Sum(orderss.totalsalesamount) as TotalOrderSales\_per\_Customer,avg(orderss.avgsalesamount)

as AverageOrderSales\_per\_Customer

From ( Select o.custid,o.orderid,Sum((od.unitprice\*od.qty)\*(1-od.discount)) as totalsalesamount ,

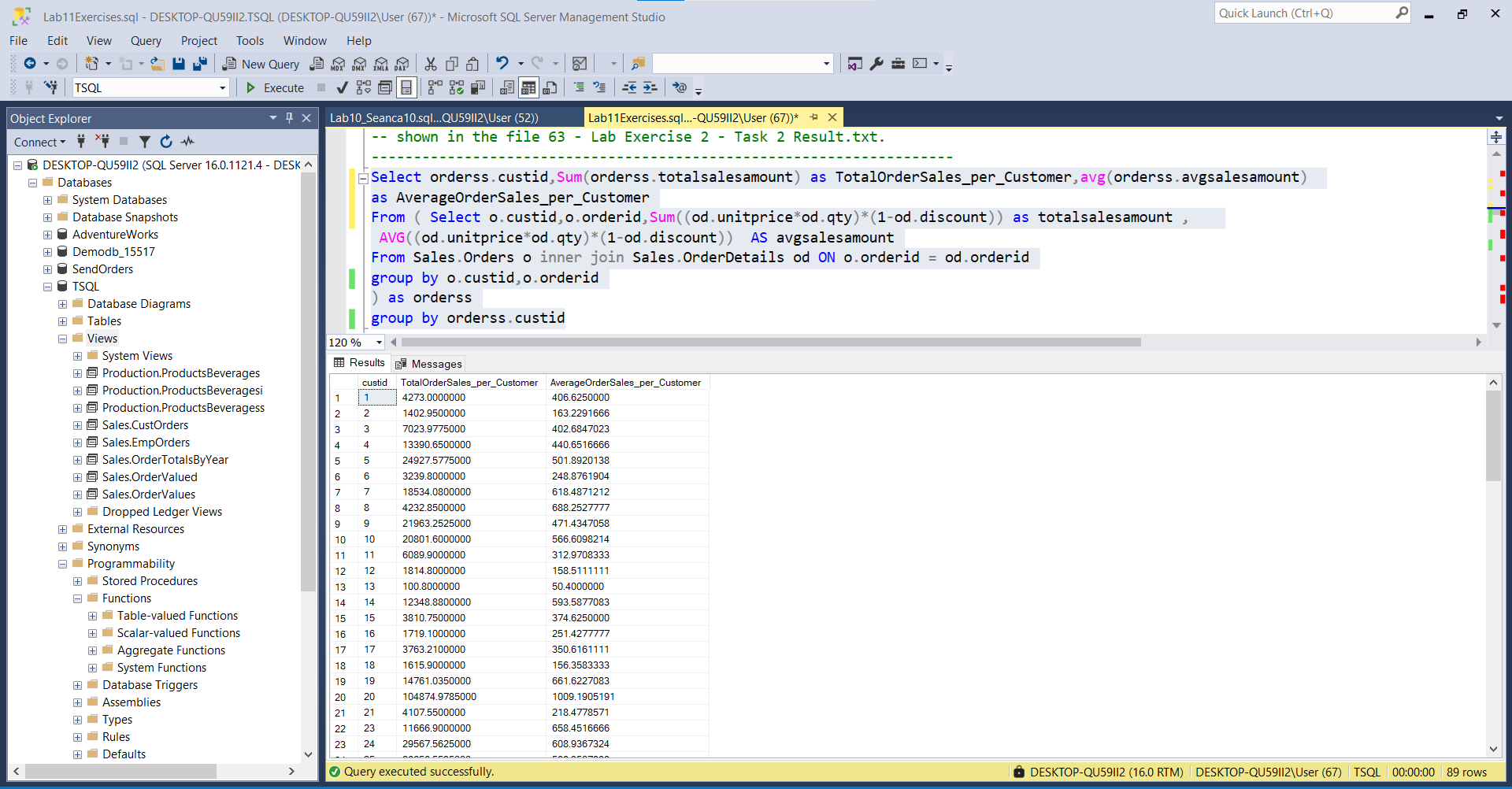
AVG((od.unitprice\*od.qty)\*(1-od.discount)) AS avgsalesamount

From Sales.Orders o inner join Sales.OrderDetails od ON o.orderid = od.orderid

group by o.custid,o.orderid

) as orderss

group by orderss.custid



----------------------------------------------------------------------------------------------------------------

-- Task 3

--

-- Write a SELECT statement to retrieve the following columns:

-- orderyear, representing the year of the order date

-- currentTotalSales, representing the total sales amount for the current order year

-- prevTotalSales, representing the total sales amount for the previous order year

-- percentGrowth, representing the percentage of sales growth in the current order year compared to the previous order year

-- You will have to write a T-SQL statement using two derived tables. To get the order year and total sales columns for each

-- SELECT statement, you can query an already existing view named Sales.OrderValues. The val column represents the sales amount.

-- Do not forget that the order year 2006 does not have a previous order year in the database,

-- but it should still be retrieved by the query.

-- Execute the T-SQL code and compare the results that you got with the recommended result shown

-- in the file 64 - Lab Exercise 2 - Task 3 Result.txt.

---------------------------------------------------------------------

Create view [Sales].[OrderValued]

AS

Select val1.pTotalSales2006,val1.cTotalSales2007,val1.percentGrowth

FROM( Select SUM(val.previewsTotalSales2006) AS pTotalSales2006,SUM(val.currentTotalSales2007) AS cTotalSales2007,

percentGrowth = ((Select SUM((od.qty\*od.unitprice)\*(1-od.discount)) AS TotalSales

From Sales.Orders o inner join Sales.OrderDetails od ON o.orderid = od.orderid

Where year(o.orderdate) = 2006)/(Select SUM((od.qty\*od.unitprice)\*(1-od.discount)) AS TotalSales

From Sales.Orders o inner join Sales.OrderDetails od ON o.orderid = od.orderid

Where year(o.orderdate) = 2007)\*100)

FROM(Select (Select SUM((od.qty\*od.unitprice)\*(1-od.discount)) AS TotalSales

From Sales.OrderDetails od

Where EXISTS (Select o.orderdate ,o.orderid

From Sales.Orders o

Where o.orderid = od.orderid And year(o.orderdate) = 2006

Group by o.orderid,o.orderdate)) as previewsTotalSales2006,

(Select SUM((od.qty\*od.unitprice)\*(1-od.discount)) AS TotalSales

From Sales.OrderDetails od

Where EXISTS (Select o.orderdate ,o.orderid

From Sales.Orders o

Where o.orderid = od.orderid And year(o.orderdate) = 2007

Group by o.orderid,o.orderdate)) as currentTotalSales2007

) as val

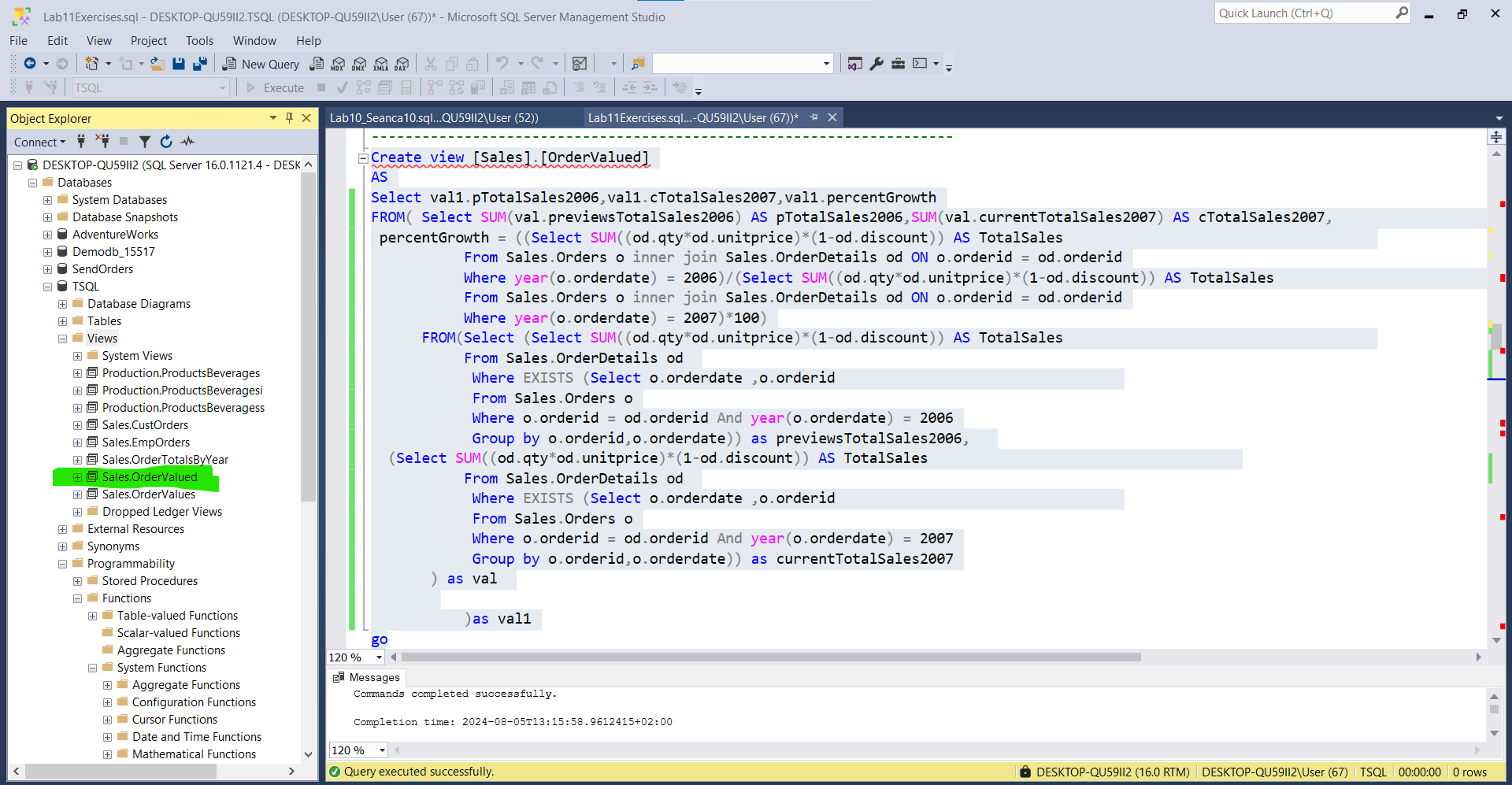
)as val1

go

Ekzekutimi i CTE-expression:



Ekzekutimi me sukses i te gjithe kodit:



------------------------------------------------------------------------------------------------------------------------------------------

-- LAB 11

--

-- Exercise 3

---------------------------------------------------------------------

USE TSQL;

GO

---------------------------------------------------------------------

-- Task 1

--

-- Write a SELECT statement like that in exercise 2, task 1, but use a CTE instead of a derived table.

-- Use inline column aliasing in the CTE query and name the CTE ProductBeverages.

-- Execute the T-SQL code and compare the results that you got with

-- the recommended result shown in the file 72 - Lab Exercise 3 - Task 1 Result.txt.

---------------------------------------------------------------------

WITH CTE\_ProductBeverages

AS

(Select o.custid,Sum((od.unitprice\*od.qty)\*(1-od.discount)) as totalsalesamount ,

AVG((od.unitprice\*od.qty)\*(1-od.discount)) AS avgsalesamount

From Sales.Orders o inner join Sales.OrderDetails od ON o.orderid = od.orderid

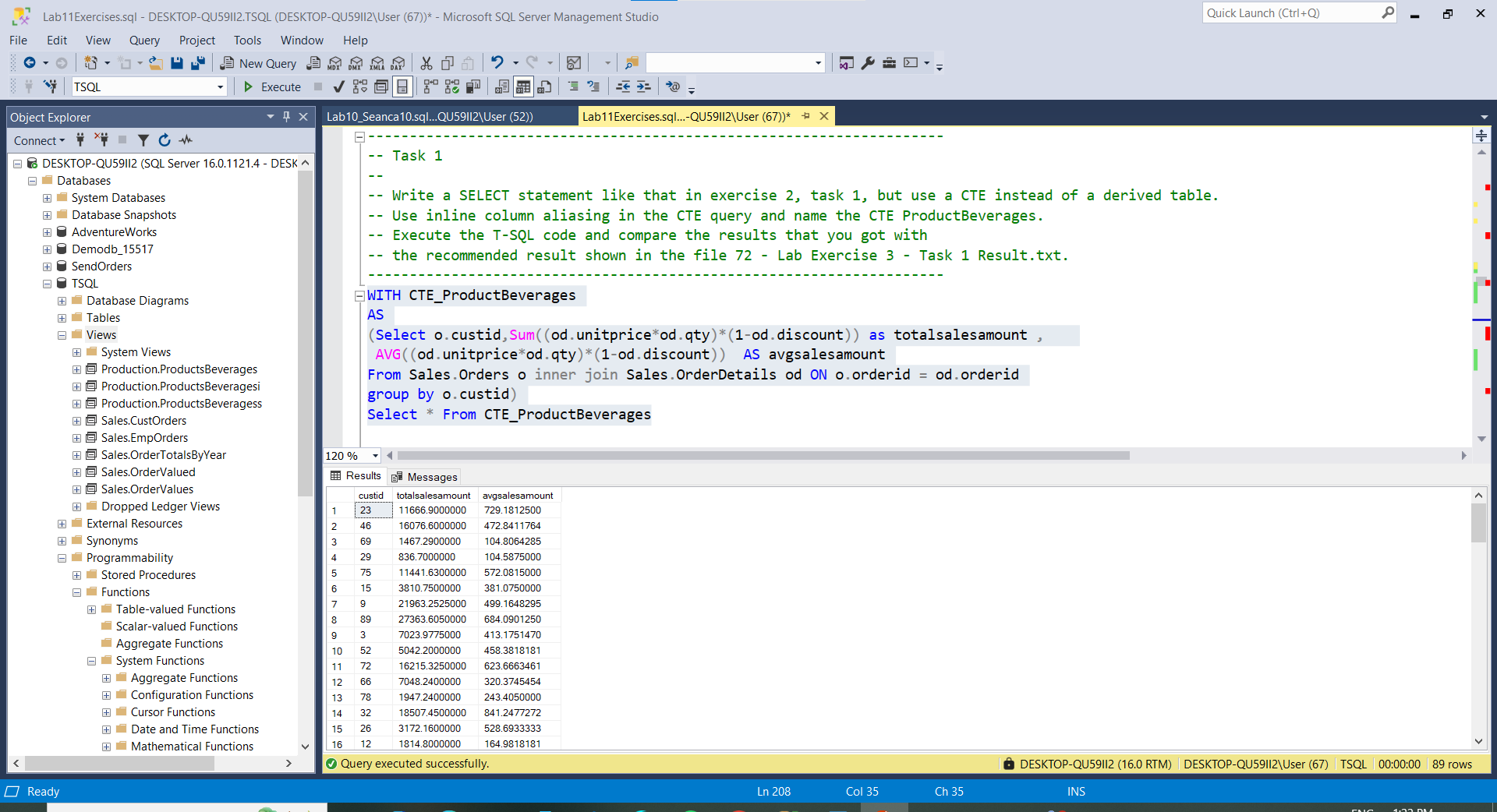
group by o.custid)

Select \* From CTE\_ProductBeverages ---------------------------------------------------------------------

Ekzekutimi me sukses i querisë



Ekzekutimi me sukses I CTE-së



-- Task 2

--

-- Write a SELECT statement against Sales.OrderValues to retrieve each customer’s ID and

-- total sales amount for the year 2008. Define a CTE named c2008 based on this query

-- using the external aliasing form to name the CTE columns custid and salesamt2008.

-- Join the Sales.Customers table and the c2008 CTE, returning the custid and contactname columns

-- from the Sales.Customers table and the salesamt2008 column from the c2008 CTE.

-- Execute the T-SQL code and compare the results that you got with the recommended result

-- shown in the file 73 - Lab Exercise 3 - Task 2 Result.txt.

---------------------------------------------------------------------

With CTE\_c2008

AS

(Select c.custid , SUM((od.qty\*od.unitprice)\*(1-od.discount)) as salesamt2008 , c.contactname

From Sales.OrderDetails od inner join Sales.Orders o ON o.orderid = od.orderid

inner join Sales.Customers c ON o.custid = c.custid

Where year(o.orderdate) = 2008

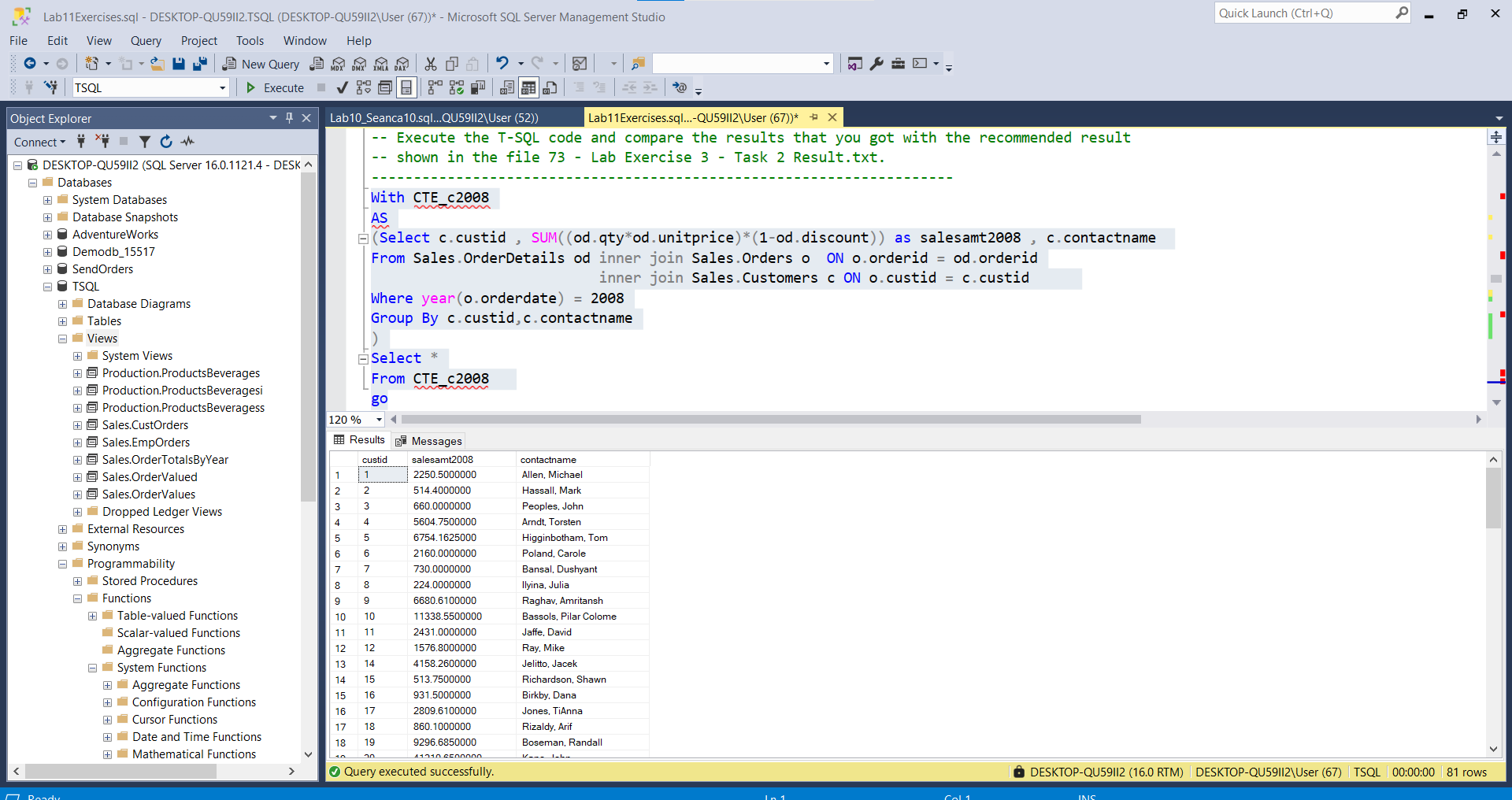
Group By c.custid,c.contactname

)

Select \*

From CTE\_c2008

Go



---------------------------------------------------------------------

-- Task 3

--

-- Write a SELECT statement to retrieve the custid and contactname columns from the Sales.Customers table.

-- Also retrieve the following calculated columns:

-- salesamt2008, representing the total sales amount for the year 2008

-- salesamt2007, representing the total sales amount for the year 2007

-- percentgrowth, representing the percentage of sales growth between the year 2007 and 2008

-- If percentgrowth is NULL, then display the value 0.

--

-- You can use the CTE from the previous task and add another CTE for the year 2007.

-- Then join both of them with the Sales.Customers table. Order the result by the percentgrowth column.

-- Execute the T-SQL code and compare the results that you got with the recommended result

-- shown in the file 74 - Lab Exercise 3 - Task 3 Result.txt.

---------------------------------------------------------------------

With CTE\_C2007

AS

(Select c.custid,c.contactname,SUM((od.qty\*od.unitprice)\*(1-od.discount)) AS previewTotalSales2007

From Sales.OrderDetails od inner join Sales.Orders o ON od.orderid = o.orderid

inner join Sales.Customers c ON o.custid = c.custid

WHERE year(o.orderdate) = 2007

group by c.custid,c.contactname

)

, CTE\_C2008

AS

(Select c.custid,SUM((od.qty\*od.unitprice)\*(1-od.discount)) AS currentTotalSales2008

From Sales.OrderDetails od inner join Sales.Orders o ON od.orderid = o.orderid

inner join Sales.Customers c ON o.custid = c.custid

WHERE year(o.orderdate) = 2008

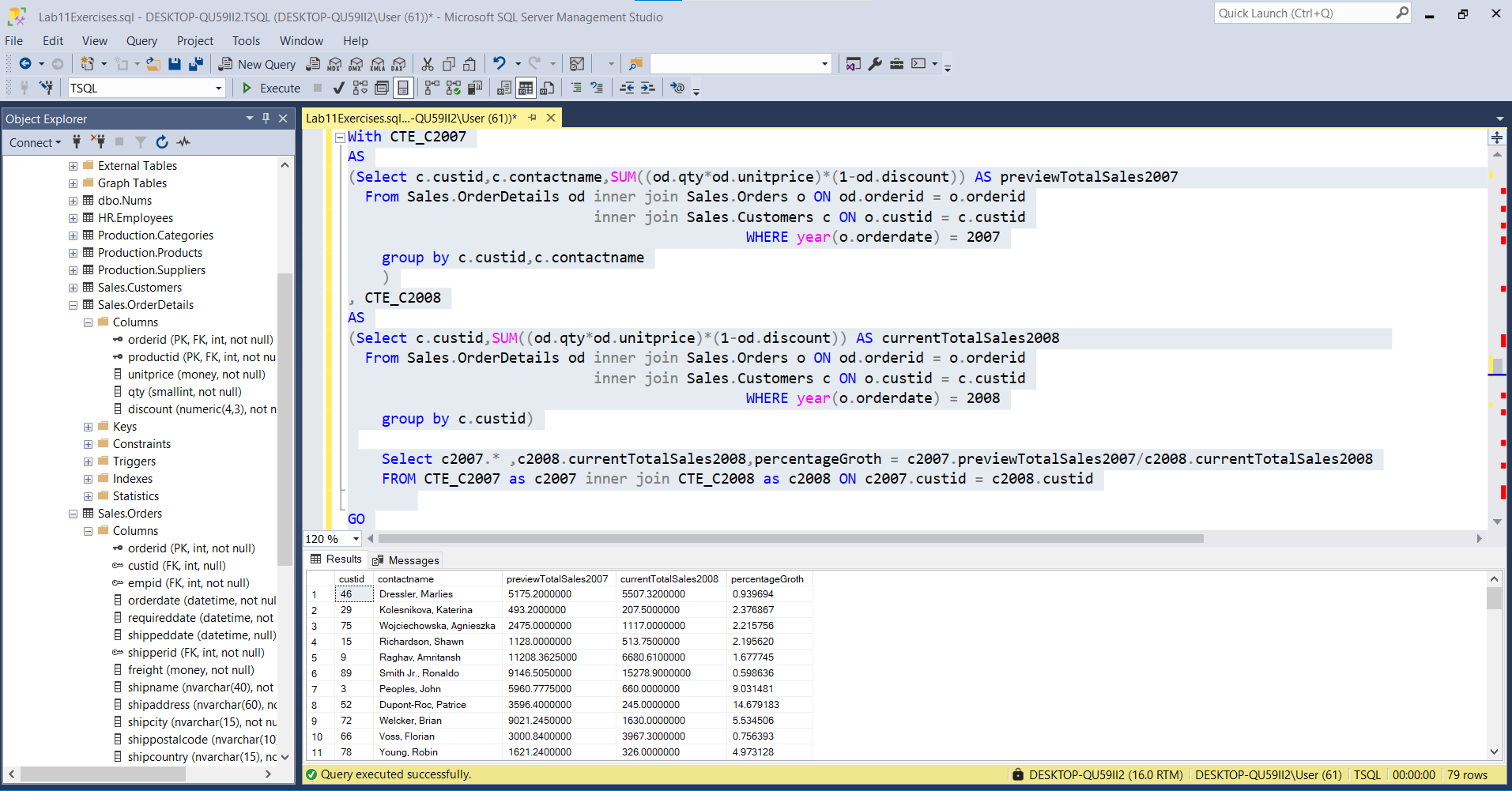
group by c.custid)

Select c2007.\* ,c2008.currentTotalSales2008, percentageGroth = c2007.previewTotalSales2007/c2008.currentTotalSales2008

FROM CTE\_C2007 as c2007 inner join CTE\_C2008 as c2008 ON c2007.custid = c2008.custid

GO

Ekzekutimi i kodit eshte si më poshtë:



---- LAB 11

--------------------------------

---- Exercise 4

---------------------------------------------------------------------

USE TSQL;

GO

---------------------------------------------------------------------

-- Task 1

--

-- Write a SELECT statement against the Sales.OrderValues view and retrieve the custid and totalsalesamount columns

-- as a total of the val column. Filter the results to include orders only for the order year 2007.

-- Execute the written statement and compare the results that you got with the recommended result

-- shown in the file 82 - Lab Exercise 4 - Task 1 Result.txt.

-- Define an inline table-valued function using the following function header and add

-- your previous query after the RETURN clause.

-- Modify the query by replacing the constant year value 2007 in the

-- WHERE clause with the parameter @orderyear.

-- Highlight the complete code and execute it. This will create an

-- inline table-valued function named dbo.fnGetSalesByCustomer.

---------------------------------------------------------------------

-- initial SQL statement

CREATE FUNCTION dbo.fnGetSalesByCustomersi

(@orderyear AS INT) RETURNS TABLE

AS

RETURN

Select c.custid , Sum((od.qty\*od.unitprice)\*(1 - od.discount)) as totalsalesamount

From Sales.Customers c inner join Sales.Orders o ON c.custid = o.custid

inner join Sales.OrderDetails od ON o.orderid = od.orderid

Where year(o.orderdate) = @orderyear

group by c.custid

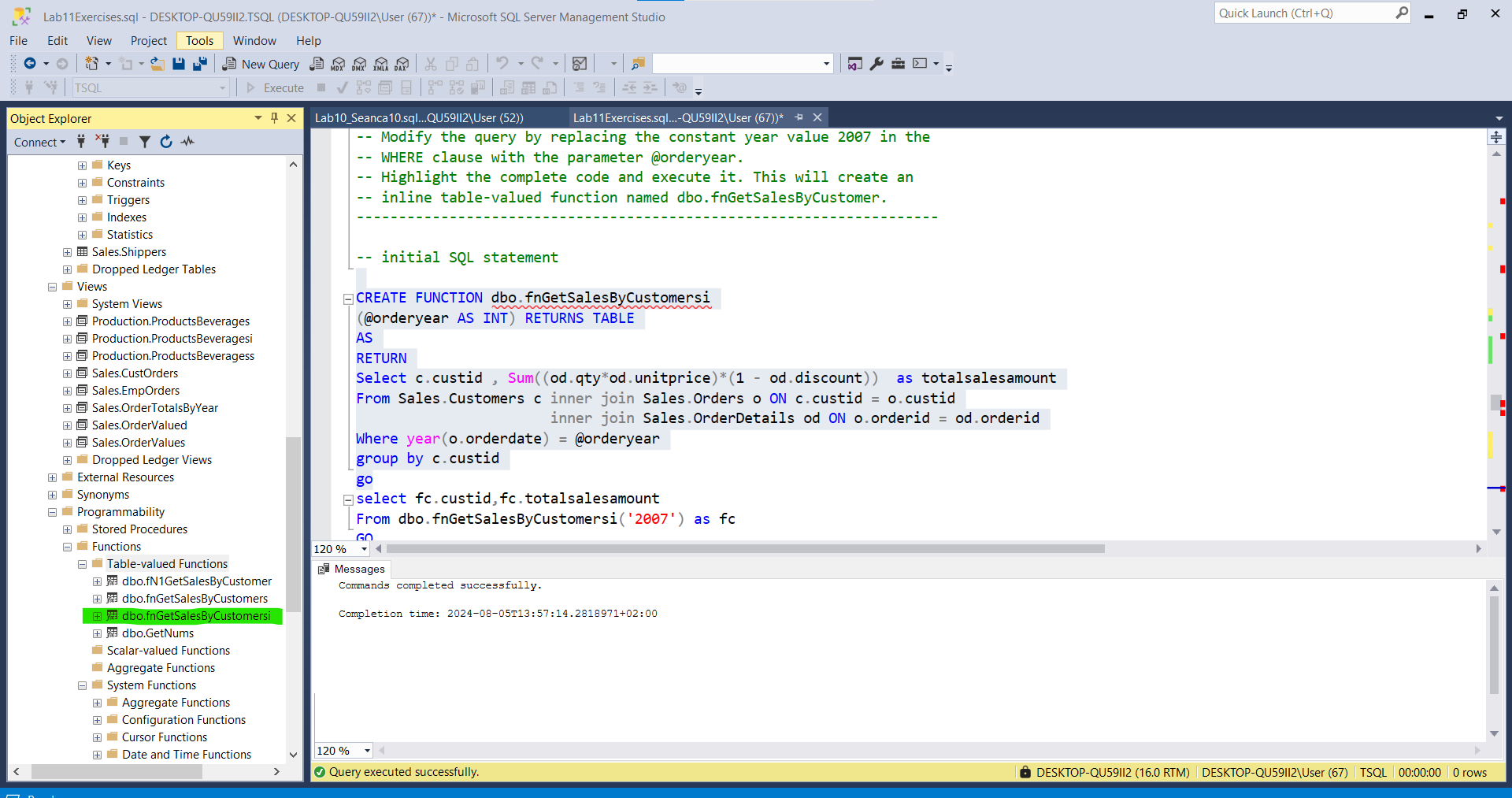
go

select fc.custid,fc.totalsalesamount

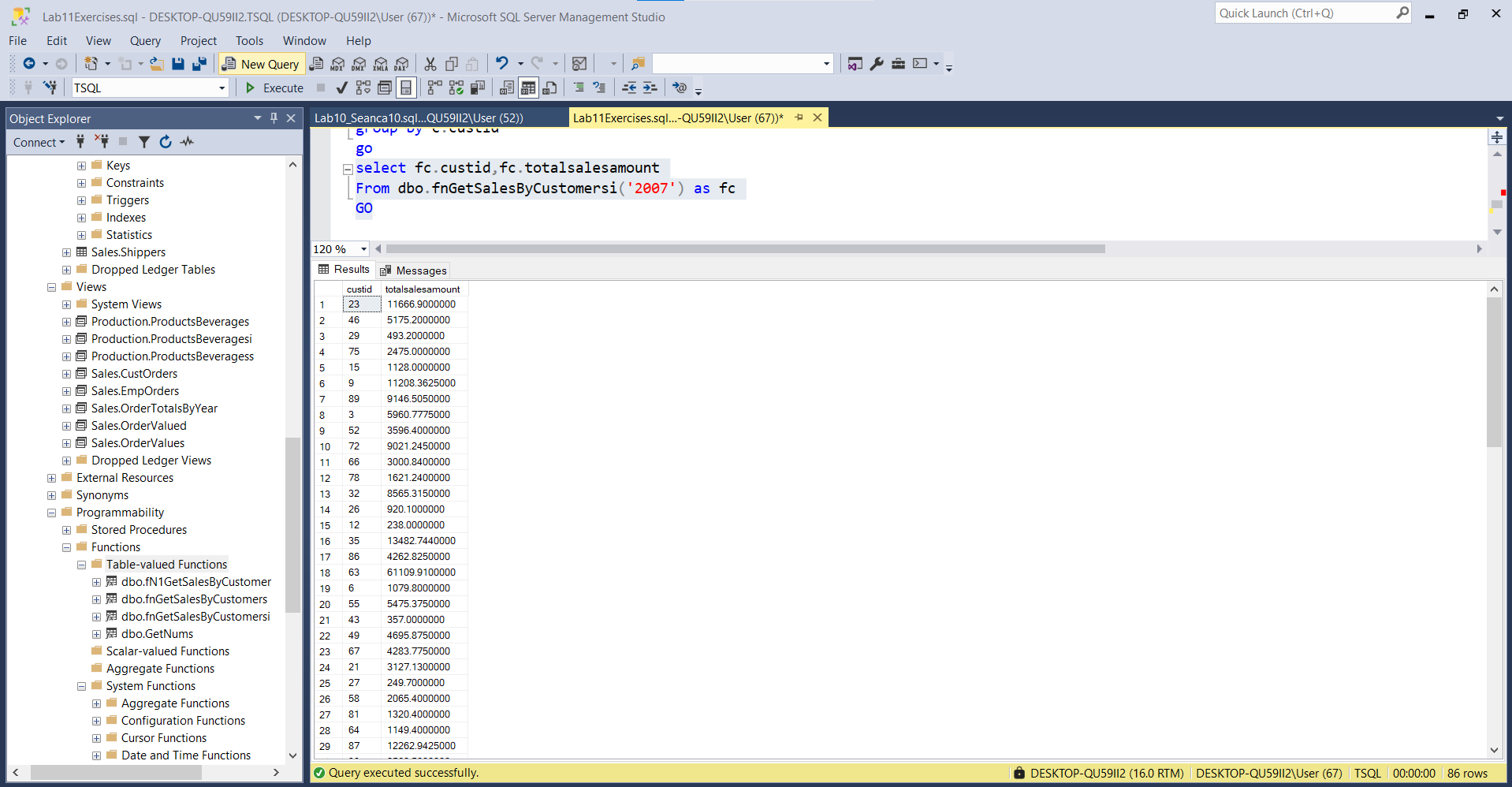
From dbo.fnGetSalesByCustomersi('2007') as fc

GO

Krijimi me sukses i Funksionit:



Ekzekutimi me sukses i Funksionit :



---------------------------------------------------------------------

-- Task 2

--

-- Write a SELECT statement to retrieve the custid and totalsalesamount columns from

-- the dbo.fnGetSalesByCustomer inline table-valued function. Use the value 2007 for the needed parameter.

-- Execute the written statement and compare the results that you got with the recommended result

-- shown in the file 83 - Lab Exercise 4 - Task 2 Result.txt.

---------------------------------------------------------------------

CREATE FUNCTION dbo.fnGetSalesByCustomer(@yearOrder as INT)

Returns Table

AS

Return

Select sc.custid,SUM(od.unitprice\*od.qty) as totalsalesamount,year(o.orderdate) as yeari

From Sales.Customers sc inner join Sales.Orders as o on sc.custid = o.custid

inner join Sales.OrderDetails as od ON o.orderid = od.orderid

Where year(o.orderdate) = @yearOrder

Group By sc.custid,year(o.orderdate)

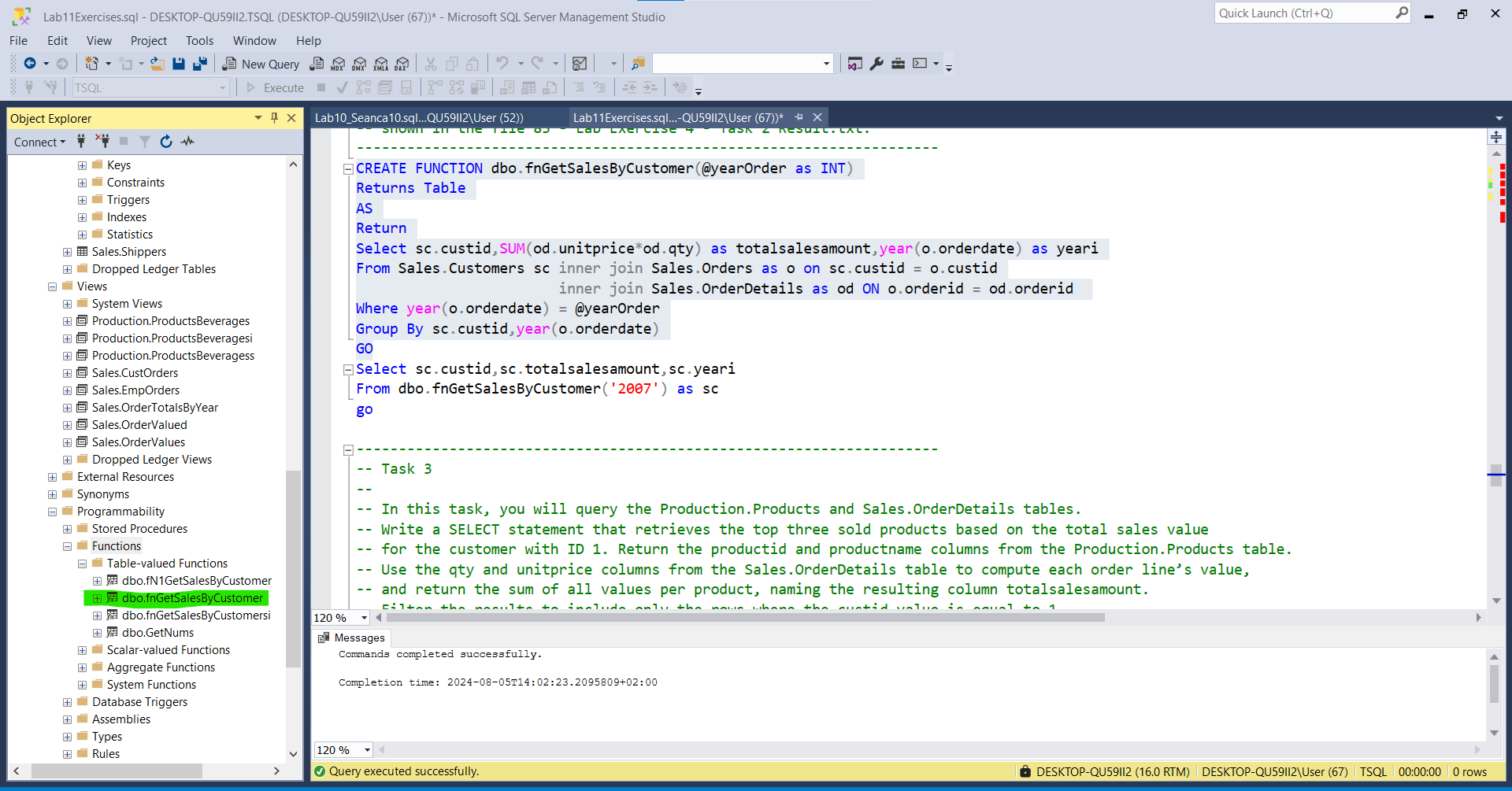
GO

Select sc.custid,sc.totalsalesamount,sc.yeari

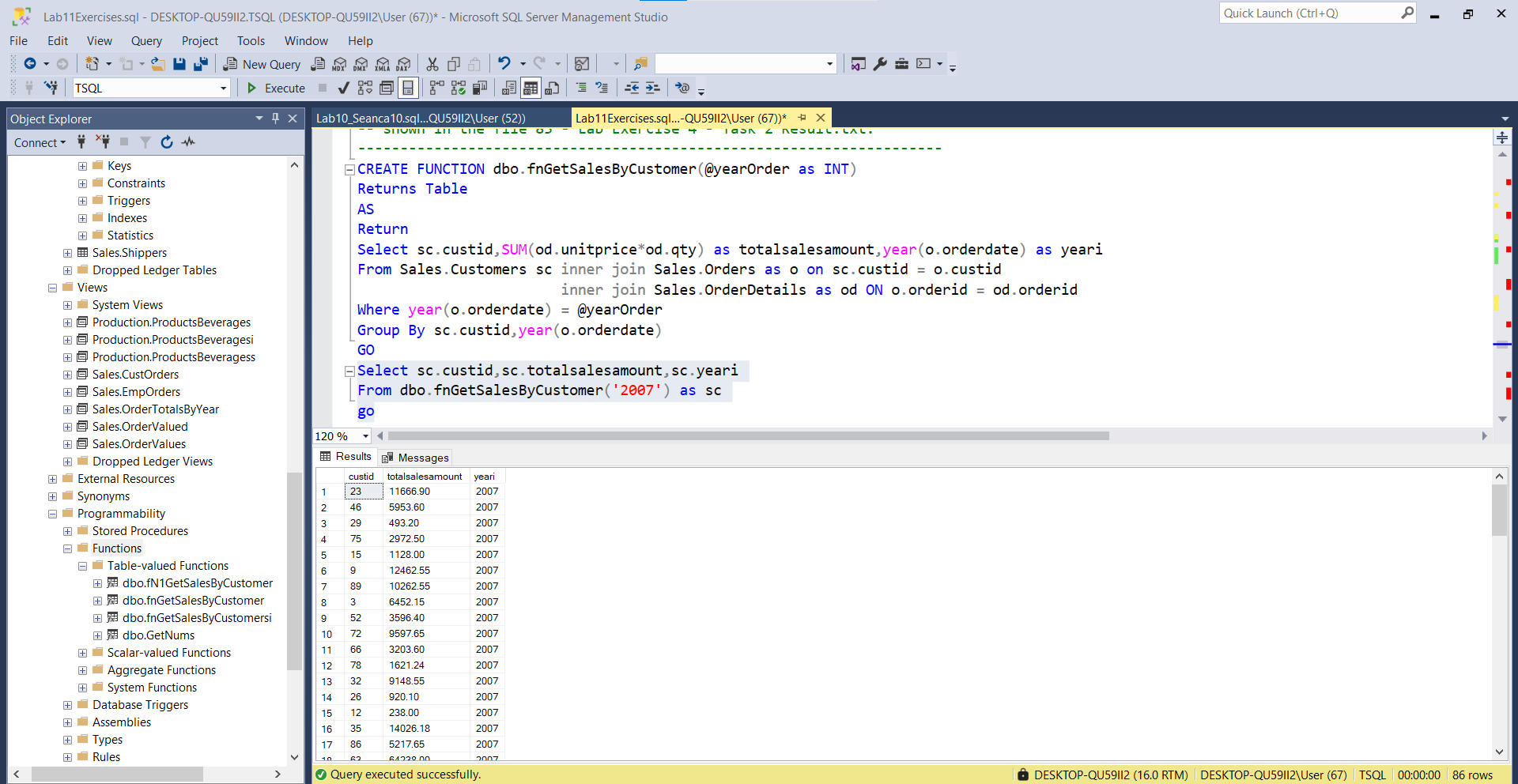
From dbo.fnGetSalesByCustomer('2007') as sc

go

Krijimi me sukses I funksionit:



Ekzekutimi me sukses I Funksionit:



---------------------------------------------------------------------

-- Task 3

--

-- In this task, you will query the Production.Products and Sales.OrderDetails tables.

-- Write a SELECT statement that retrieves the top three sold products based on the total sales value

-- for the customer with ID 1. Return the productid and productname columns from the Production.Products table.

-- Use the qty and unitprice columns from the Sales.OrderDetails table to compute each order line’s value,

-- and return the sum of all values per product, naming the resulting column totalsalesamount.

-- Filter the results to include only the rows where the custid value is equal to 1.

-- Execute the T-SQL code and compare the results that you got with the recommended result

-- shown in the file 84 - Lab Exercise 4 - Task 3\_1 Result.txt.

-- Create an inline table-valued function based on the following function header,using the previous SELECT statement.

-- Replace the constant custid value 1 in the query with the function’s input parameter @custid:

-- Highlight the complete code and execute it. This will create an inline table-valued function named

-- dbo.fnGetTop3ProductsForCustomer that excepts a parameter for the customer id.

-- Test the created inline table-valued function by writing a SELECT statement against it

-- and use the value 1 for the customer id parameter. Retrieve the productid, productname, and totalsalesamount

-- columns, and use the alias p for the inline table-valued function.

-- Execute the T-SQL code and compare the results that you got with the recommended result

-- shown in the file 85 - Lab Exercise 4 - Task 3\_2 Result.txt.

---------------------------------------------------------------------

-- initial SQL statement-- write here the SQL statement against the created function--

CREATE FUNCTION dbo.fnGetTop3ProductsForCustomer

(@custid AS INT) RETURNS TABLE

AS

RETURN

Select TOP(3) pp.productid , pp.productname , Sum((od.qty\*od.unitprice)\*(1-od.discount)) as totalSalesValue

FROM Production.Products pp INNER JOIN Sales.OrderDetails od ON pp.productid = od.productid

Where Exists(Select \*

From Sales.Orders o

Where od.orderid = o.orderid and EXISTS

(SELECT c.custid

From Sales.Customers c

Where c.custid = o.custid and c.custid = @custid)

)

Group By pp.productid,pp.productname

Order By Sum((od.qty\*od.unitprice)\*(1-od.discount)) DESC

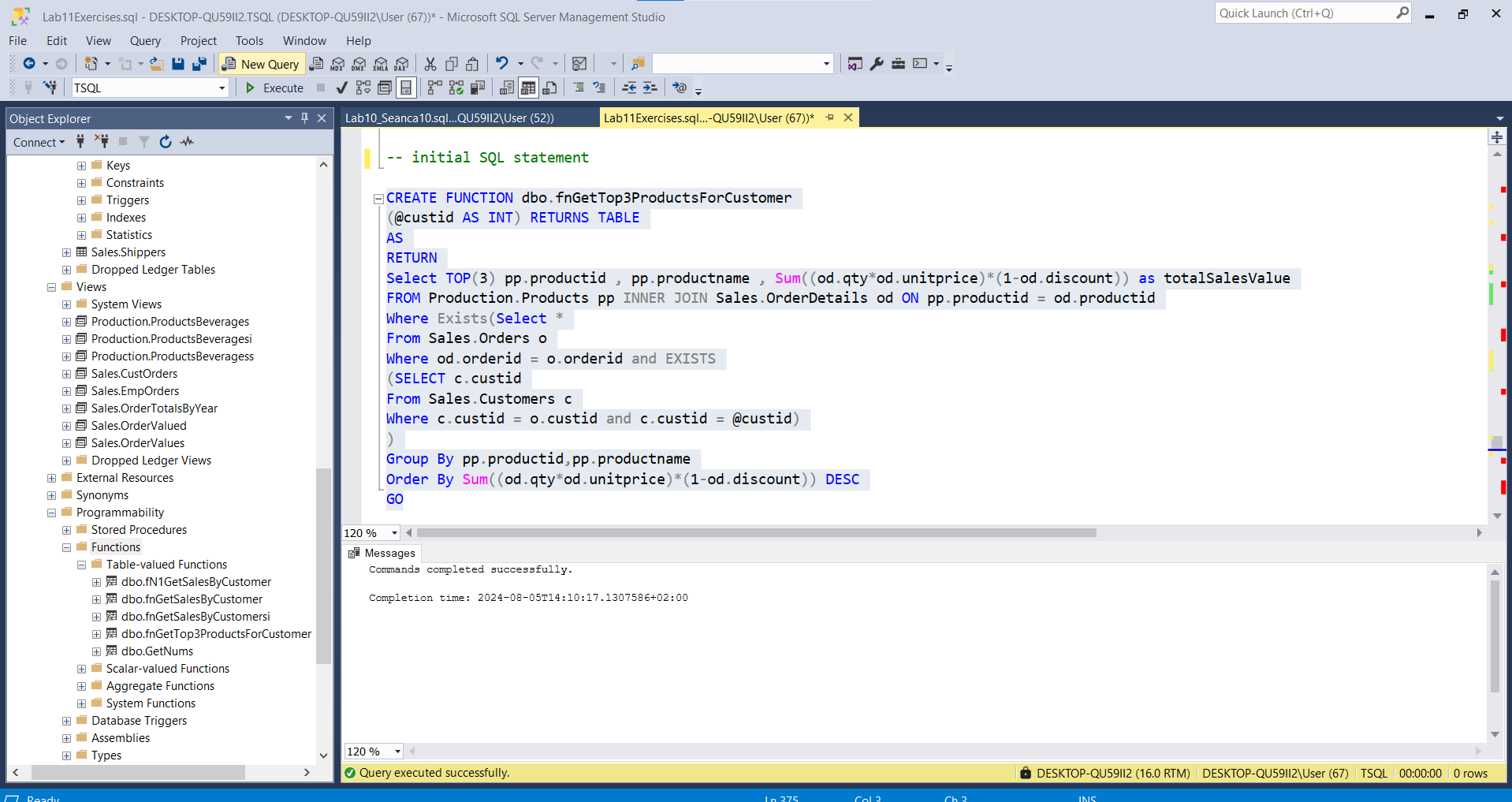
GO

Select f3c.productid,f3c.productname , f3c.totalSalesValue

From dbo.fnGetTop3ProductsForCustomer('1') f3c

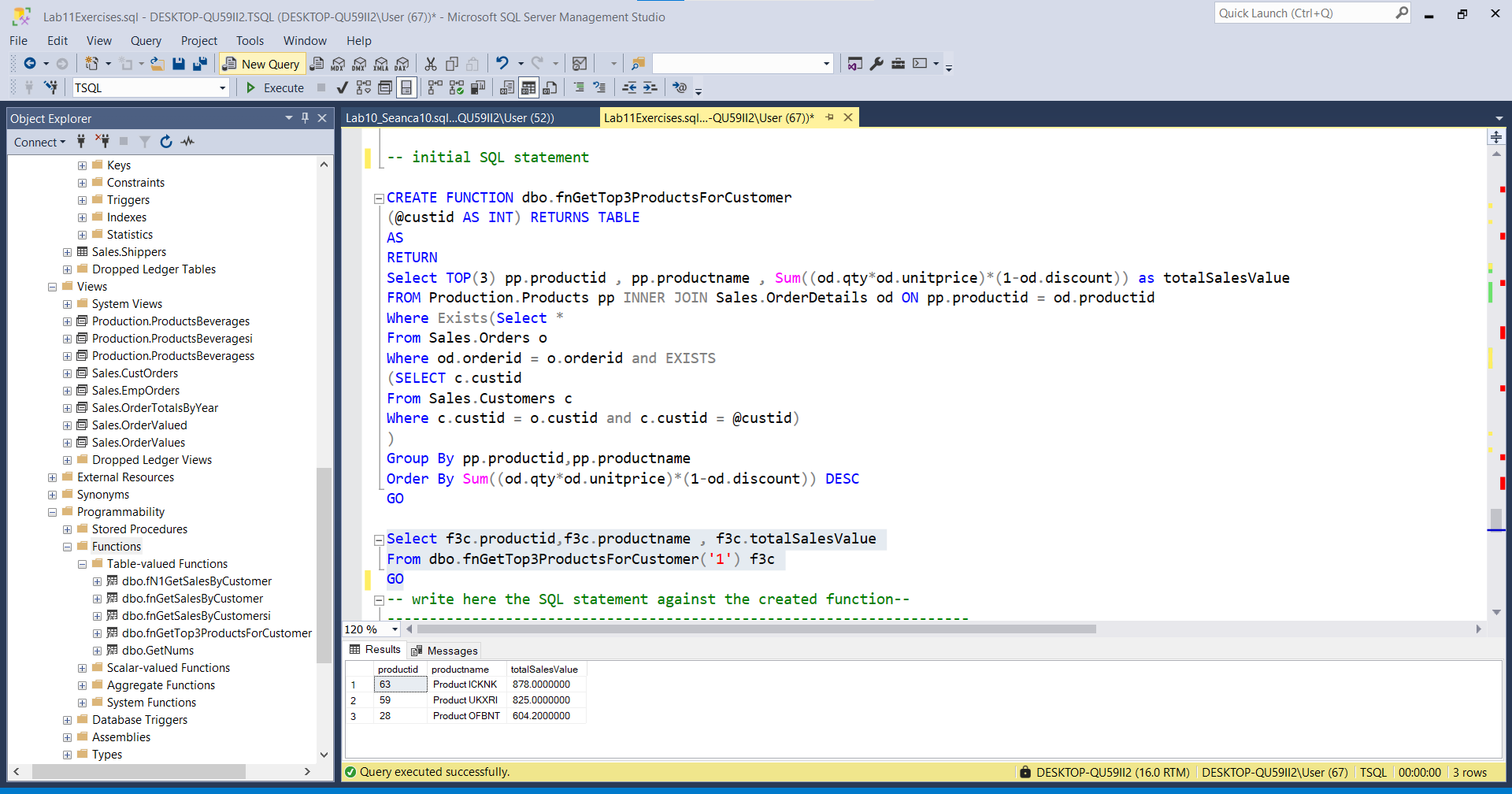
GO

Krijimi I funksionit:



---------------------------------------------------------------------

Ekzekutimi me sukses I funksionit:



-- Task 4

--

-- Write a SELECT statement to retrieve the same result as in exercise 3, task 3, but use

-- the created inline table-valued function in task 2 (dbo.fN1GetSalesByCustomer).

-- Execute the written statement and compare the results that you got with the recommended result

-- shown in the file 86 - Lab Exercise 4 - Task 4 Result.txt.

---------------------------------------------------------------------

create function dbo.fN1GetSalesByCustomer(@custid int)

RETURNS TABLE

AS

RETURN

Select TOP(3) pp.productid , pp.productname , Sum((od.qty\*od.unitprice)\*(1-od.discount)) as totalSalesValue

FROM Production.Products pp INNER JOIN Sales.OrderDetails od ON pp.productid = od.productid

Where Exists(Select \*

From Sales.Orders o

Where od.orderid = o.orderid and EXISTS

(SELECT c.custid

From Sales.Customers c

Where c.custid = o.custid and c.custid = @custid)

)

Group By pp.productid,pp.productname

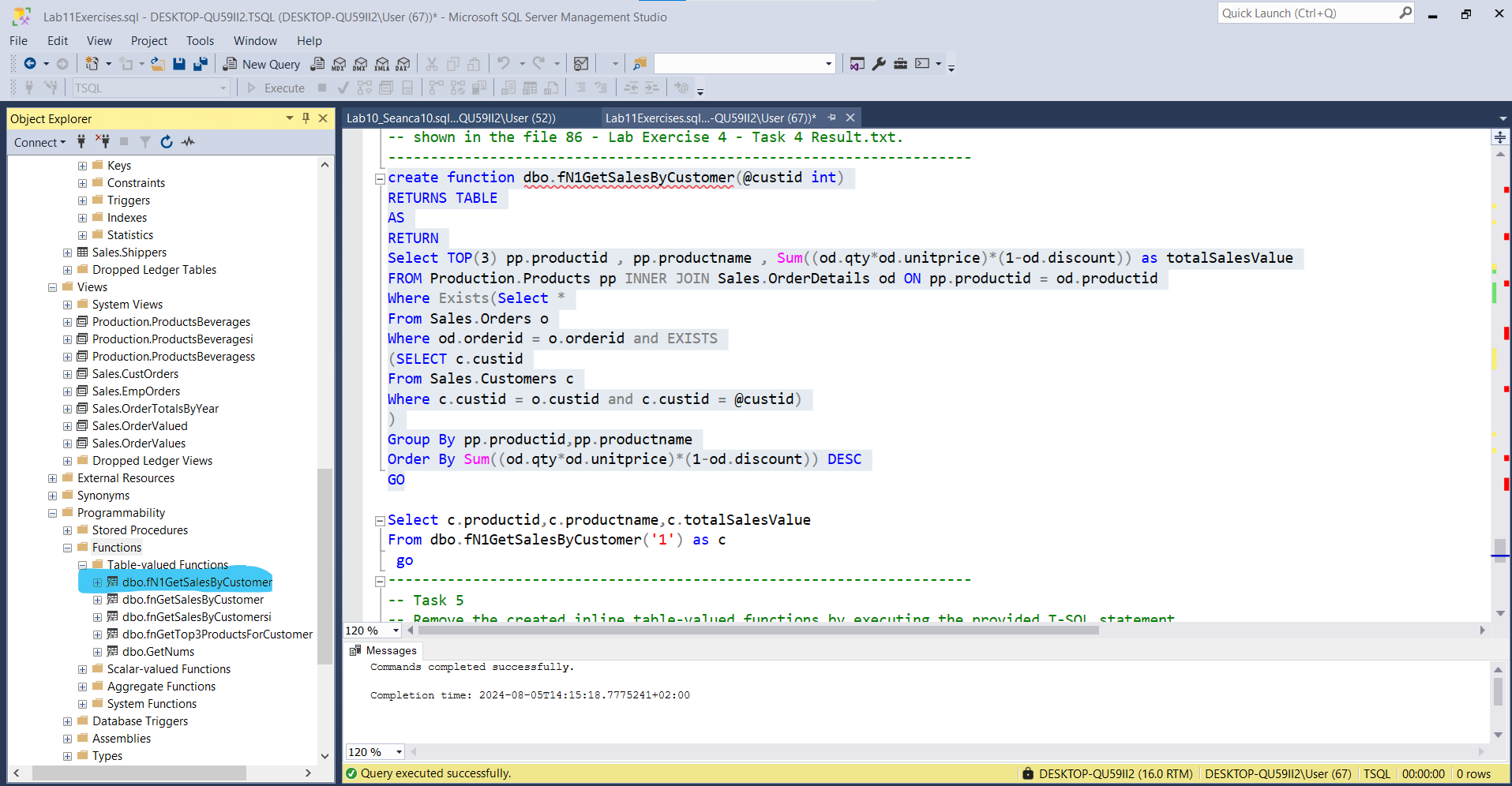
Order By Sum((od.qty\*od.unitprice)\*(1-od.discount)) DESC

GO

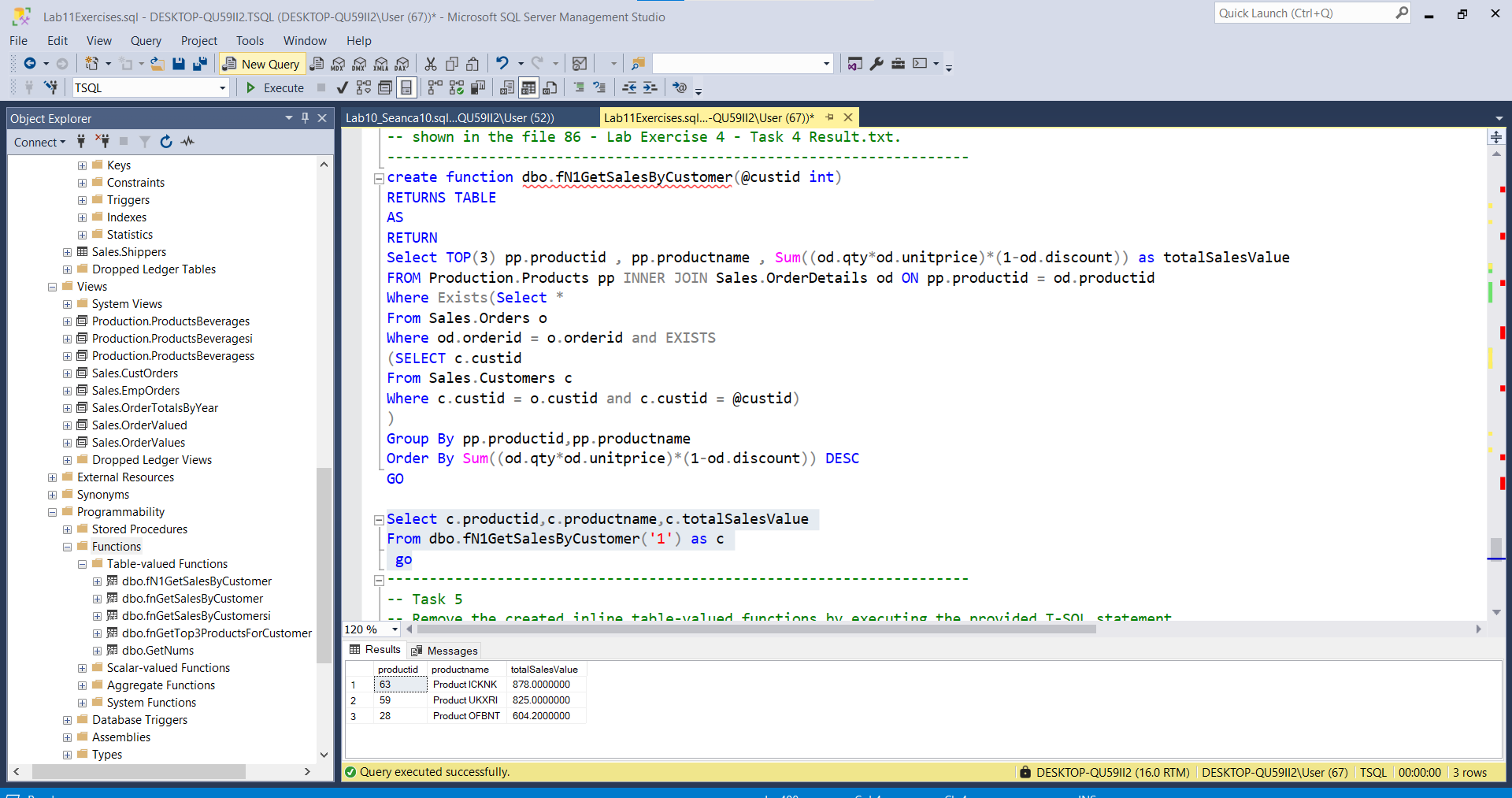
Select c.productid,c.productname,c.totalSalesValue

From dbo.fN1GetSalesByCustomer('1') as c

Go--Krijimi me sukses i funksionit:



Ekzekutimi me sukses I funksionit:



---------------------------------------------------------------------

-- Task 5

-- Remove the created inline table-valued functions by executing the provided T-SQL statement.

-- Execute this code exactly as written inside a query window.

---------------------------------------------------------------------

IF OBJECT\_ID('dbo.fnGetSalesByCustomer') IS NOT NULL

DROP FUNCTION dbo.fnGetSalesByCustomer;

-------------------------------------------------------

Commands completed successfully.

Completion time: 2024-08-04T22:58:47.6017213+02:00

----------------------------------------------------------------

IF OBJECT\_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL

DROP FUNCTION dbo.fnGetTop3ProductsForCustomer;

GO

----------------------------------------------------------

Commands completed successfully.

Completion time: 2024-08-04T22:59:19.6771092+02:00

---------------------------------------------------