JOBSHEET
PRAKTIKUM BASIS DATA LANJUT

Jurusan Teknologi Informasi

POLITEKNIK NEGERI MALANG

Week 6

SQL SERVER – TABLE EXPRESSION

Information Technology Department, Malang State Polytechnic

Jobsheet 6: Table Expression

Supervisor: Advanced Database Teaching Team

September 2024

Topics

1. Table Expressions

SAFRIZAL RAHMAN_19_SIB_2G

Objective

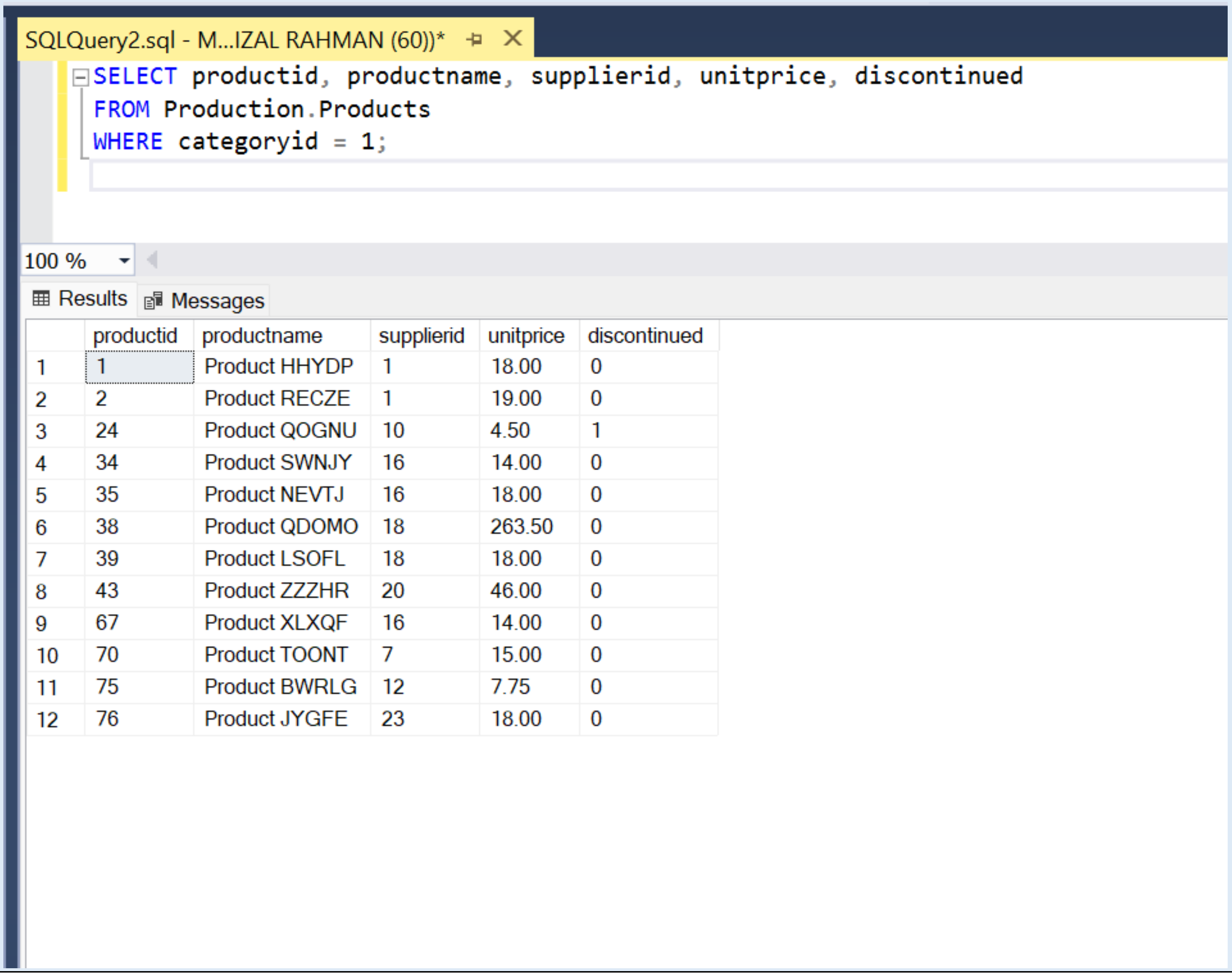
1. Students understand how to use VIEWS
2. Students understand how to use derived tables
3. Students understand how to use common table-expression (CTE)
4. Students understand how to use inline table-valued functions (TVF)

General Instructions

1. Follow the steps in the practical sections in the order given.
2. You can use SQL Server 2012 Standard Edition to try the practicum on this jobsheet. Adjust it to your computer's condition.
3. Answer all questions marked [Question-X] that are found in certain steps in each part of the practicum.
4. In each step of the practicum, there is an explanation that will help you answer the questions in instruction number 3, so read and do all the practicum parts in this jobsheet.
5. Write the answers to the questions in the instructions number 3 in a report that is done using a word processing application (Word, OpenOffice, or other similar). Export as a **PDF file** with the following name format:
   - **BDL_Task 6 _Class_2X_AbsenteeNumberDigit_YourFullName** .pdf - Example:
     - **BDL_Assignment 6 _SIB2Q_99_DonaldDuck** .pdf -

   Pay close attention to the naming format.
   - Collect the PDF files as a practical report to the supervising lecturer.
   - In addition to the file name, also include your identity on the first page of the report.

Lab – Part 1: View - Write a SELECT query to get all products in a particular category

Step	Information
1	Make sure your database is connected to 'TSQL'
2	[Question- 1] Write a SELECT query to display the *productid* , *productname* , *supplierid* , *unitprice* and *discontinued columns* from the **Productions.Product table** . Then filter the results to only display products in the Beverages category (`categoryid = 1`) 

SQLQuery2.sql - M...IZAL RAHMAN (60))*

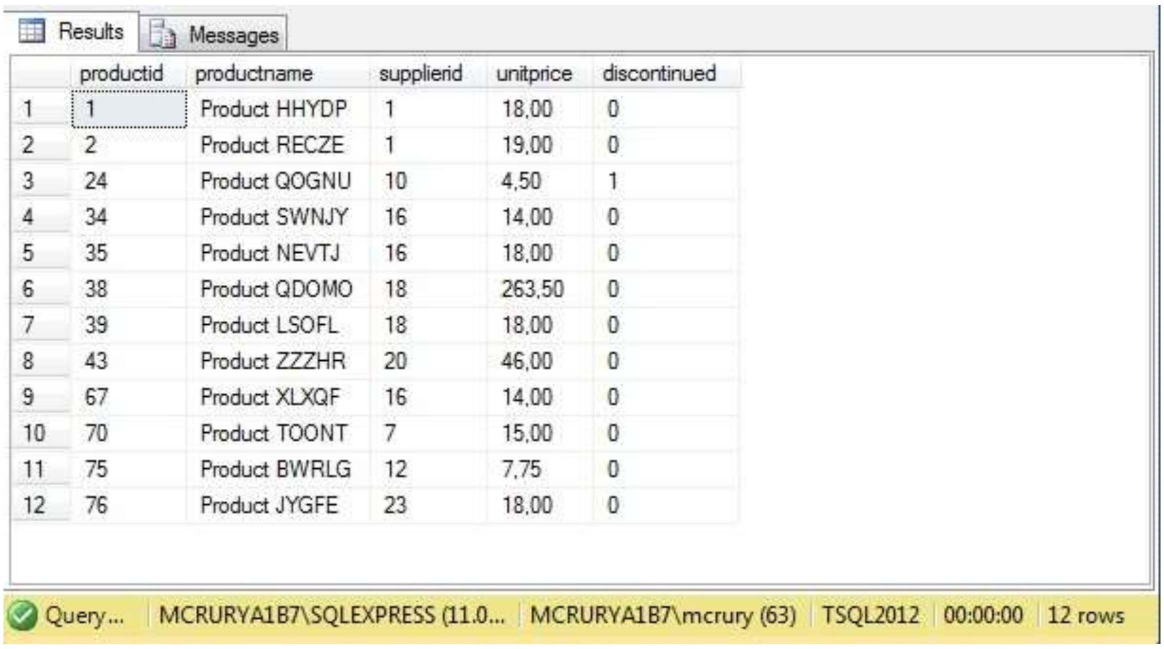
```sql
SELECT productid, productname, supplierid, unitprice, discontinued
FROM Production.Products
WHERE categoryid = 1;
```

100 %

Results Messages

	productid	productname	supplierid	unitprice	discontinued
1	1	Product HHYDP	1	18.00	0
2	2	Product RECZE	1	19.00	0
3	24	Product QOGNU	10	4.50	1
4	34	Product SWNJY	16	14.00	0
5	35	Product NEVTJ	16	18.00	0
6	38	Product QDOMO	18	263.50	0
7	39	Product LSOFL	18	18.00	0
8	43	Product ZZZHR	20	46.00	0
9	67	Product XLXQF	16	14.00	0
10	70	Product TOONT	7	15.00	0
11	75	Product BWRLG	12	7.75	0
12	76	Product JYGFE	23	18.00	0

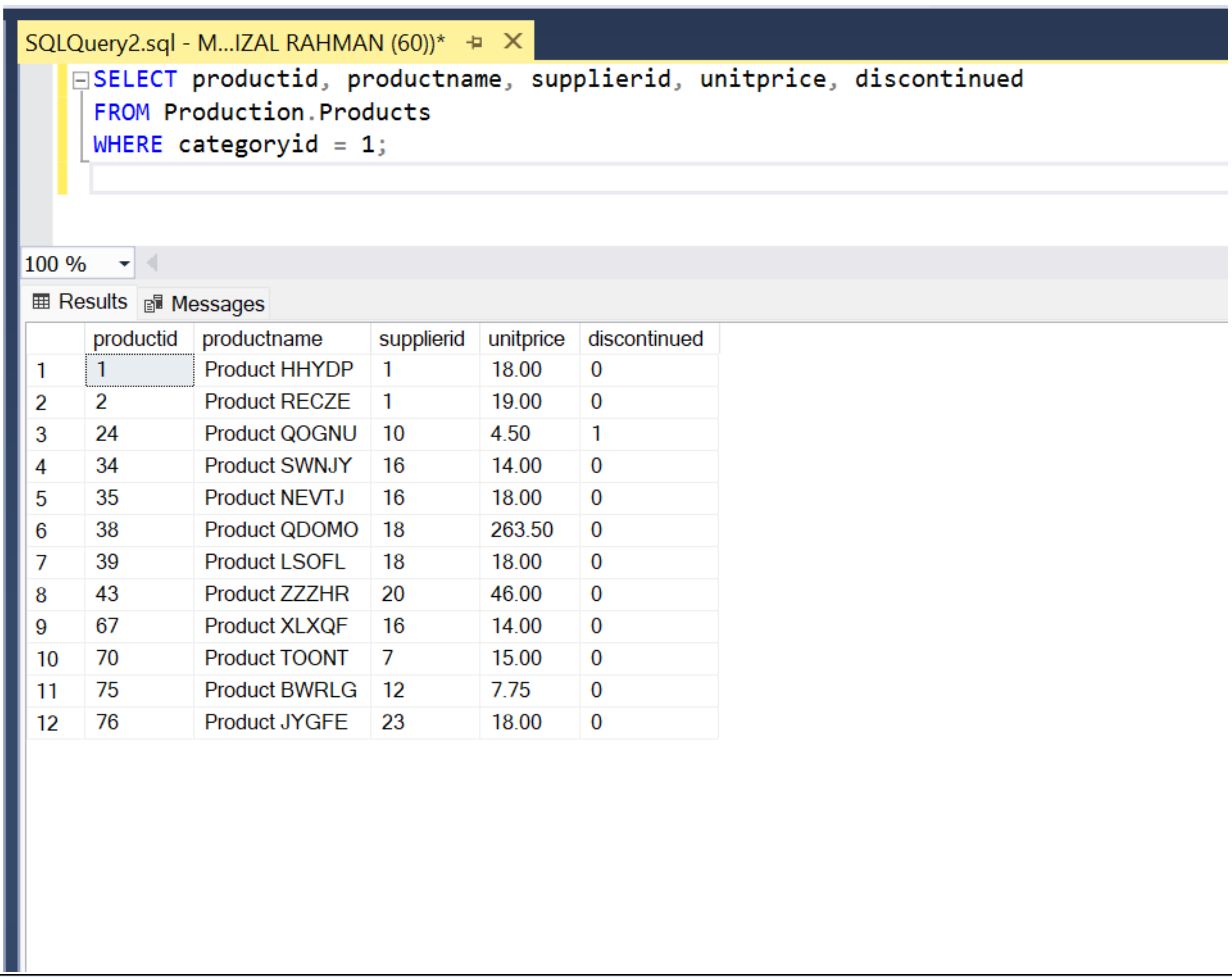
3	Execute the query in step 2 above and compare it with the results shown in the following display:



Results | Messages

	productid	productname	supplierid	unitprice	discontinued
1	1	Product HHYDP	1	18,00	0
2	2	Product RECZE	1	19,00	0
3	24	Product QOGNU	10	4,50	1
4	34	Product SWNJY	16	14,00	0
5	35	Product NEVTJ	16	18,00	0
6	38	Product QDOMO	18	263,50	0
7	39	Product LSOFL	18	18,00	0
8	43	Product ZZZHR	20	46,00	0
9	67	Product XLXQF	16	14,00	0
10	70	Product TOONT	7	15,00	0
11	75	Product BWRLG	12	7,75	0
12	76	Product JYGFE	23	18,00	0

Query... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (63) | TSQL2012 | 00:00:00 | 12 rows

	
4	[Question- 2] Modify the T-SQL code from no. 2 above by adding the following T-SQL code (place it before the SELECT query) `CREATE VIEW Production . ProductsBeverages AS`

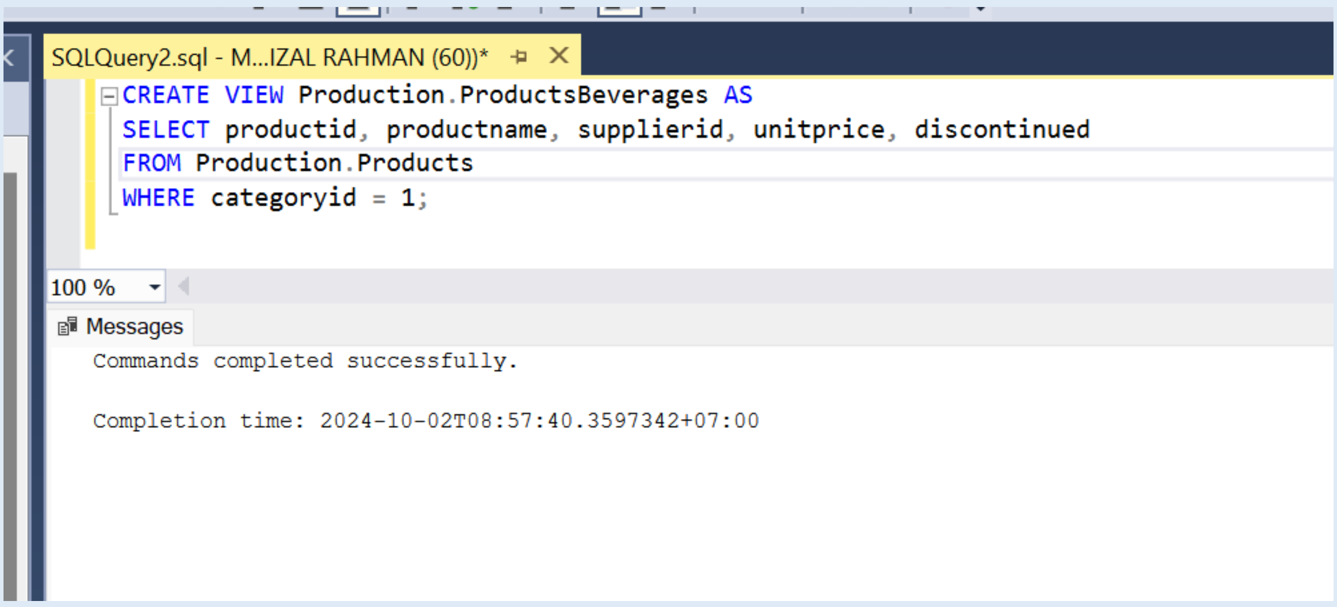
SQLQuery2.sql - M...IZAL RAHMAN (60))*

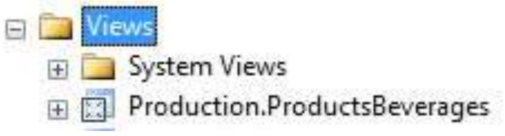
```
SELECT productid, productname, supplierid, unitprice, discontinued
FROM Production.Products
WHERE categoryid = 1;
```

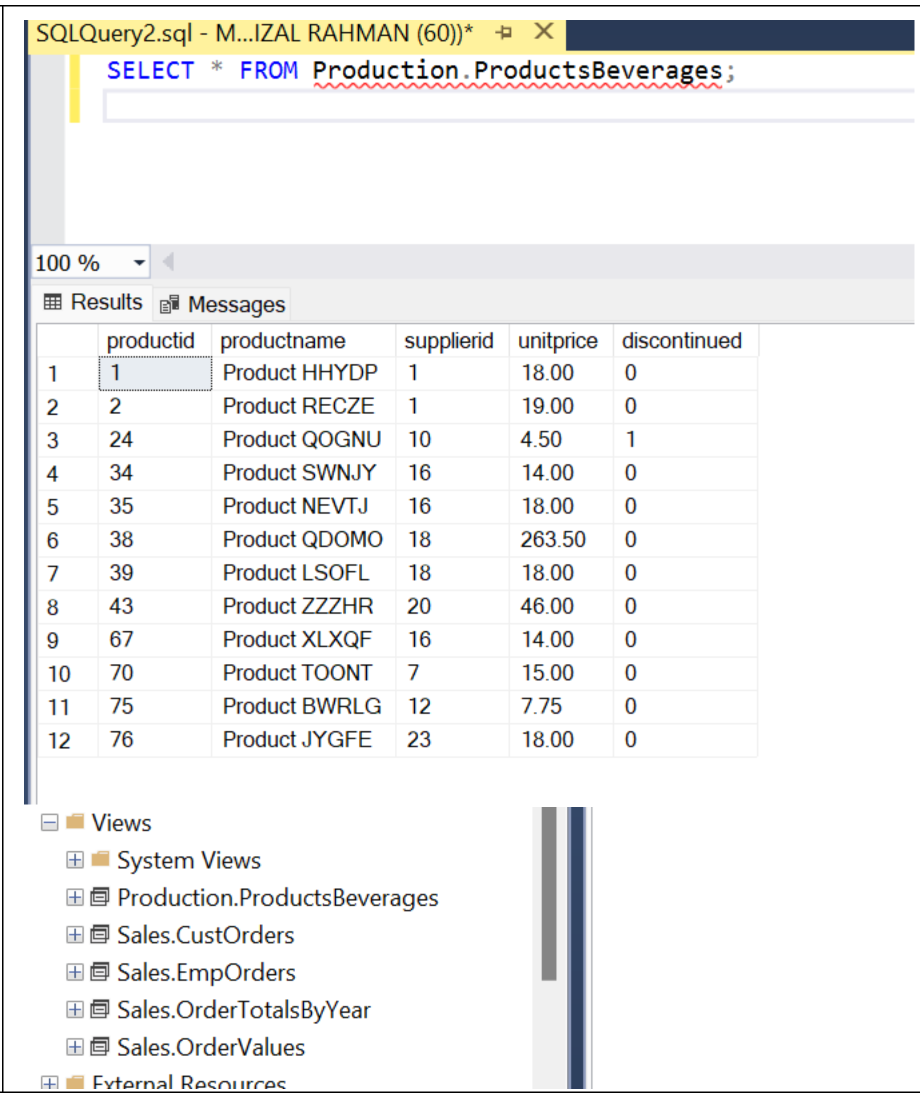
100 %

Results | Messages

	productid	productname	supplierid	unitprice	discontinued
1	1	Product HHYDP	1	18.00	0
2	2	Product RECZE	1	19.00	0
3	24	Product QOGNU	10	4.50	1
4	34	Product SWNJY	16	14.00	0
5	35	Product NEVTJ	16	18.00	0
6	38	Product QDOMO	18	263.50	0
7	39	Product LSOFL	18	18.00	0
8	43	Product ZZZHR	20	46.00	0
9	67	Product XLXQF	16	14.00	0
10	70	Product TOONT	7	15.00	0
11	75	Product BWRLG	12	7.75	0
12	76	Product JYGFE	23	18.00	0

	 SQLQuery2.sql - M...IZAL RAHMAN (60))*
	 `CREATE VIEW Production.ProductsBeverages AS`
	 `SELECT productid, productname, supplierid, unitprice, discontinued`
	 `FROM Production.Products`
	 `WHERE categoryid = 1;`
	 100 %
	 Messages
	 `Commands completed successfully.`
	 `Completion time: 2024-10-02T08:57:40.3597342+07:00`

5	Execute the query in step 4 above, which will produce a *VIEW object* named **ProductsBeverages** in the **Production schema.**  Views System Views Production.ProductsBeverages



SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
SELECT * FROM Production.ProductsBeverages;
```

100 %

Results | Messages

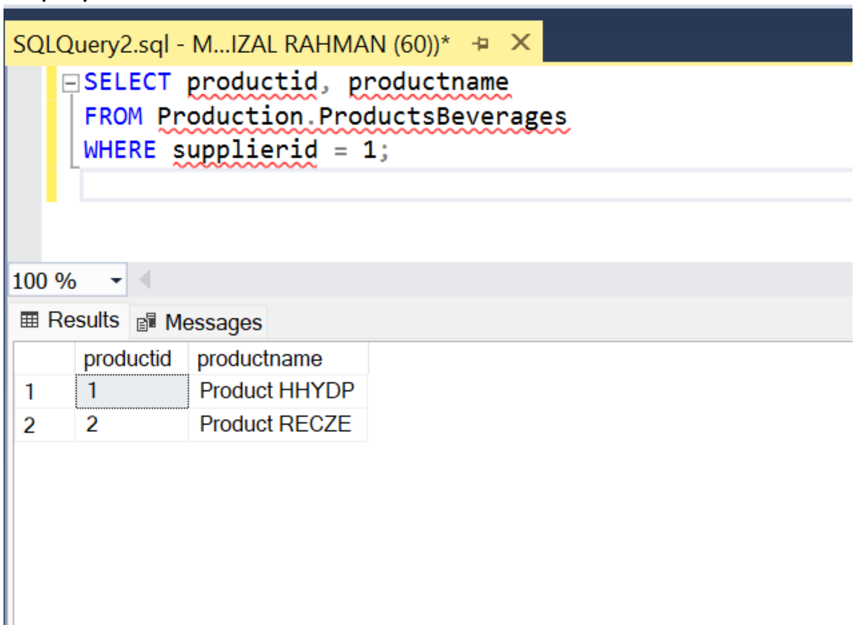
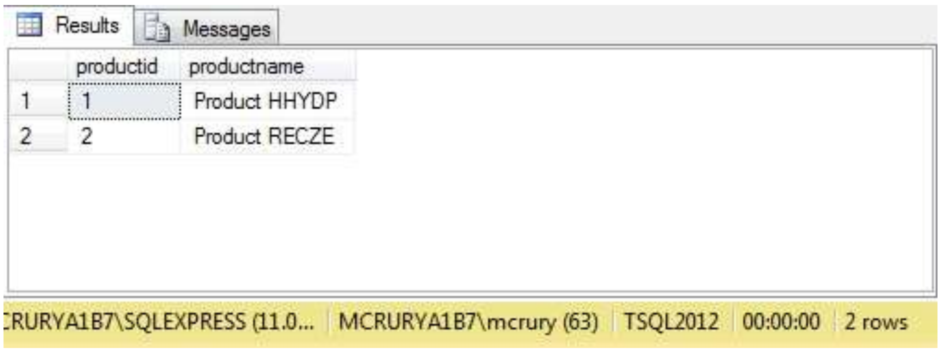
	productid	productname	supplierid	unitprice	discontinued
1	1	Product HHYDP	1	18.00	0
2	2	Product RECZE	1	19.00	0
3	24	Product QOGNU	10	4.50	1
4	34	Product SWNJY	16	14.00	0
5	35	Product NEVTJ	16	18.00	0
6	38	Product QDOMO	18	263.50	0
7	39	Product LSOFL	18	18.00	0
8	43	Product ZZZHR	20	46.00	0
9	67	Product XLXQF	16	14.00	0
10	70	Product TOONT	7	15.00	0
11	75	Product BWRLG	12	7.75	0
12	76	Product JYGFE	23	18.00	0

Views
- System Views
- Production.ProductsBeverages
- Sales.CustOrders
- Sales.EmpOrders
- Sales.OrderTotalsByYear
- Sales.OrderValues

External Resources

Practical – Part 2: View - Writing a SELECT query against the VIEW that has been created

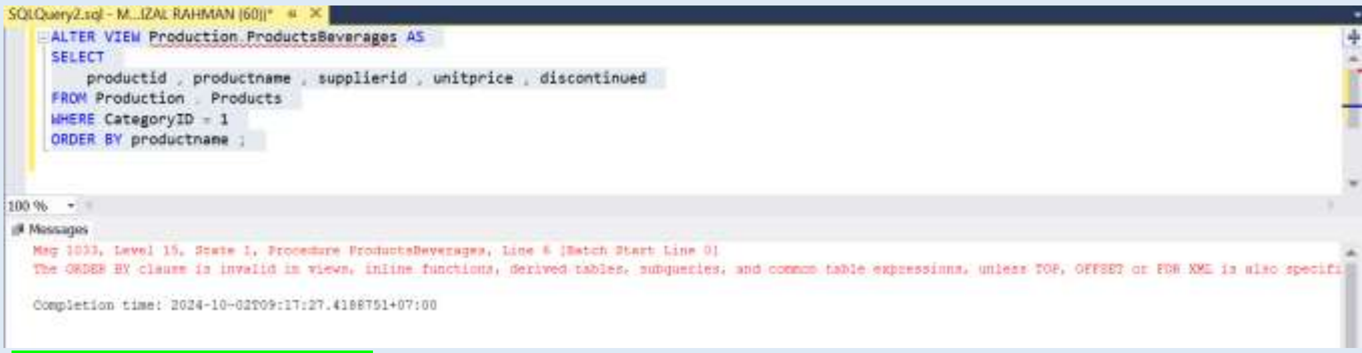
Step	Information
1	[Question- 3] Create a SELECT query consisting of the *productid* and *productname columns* from *VIEW* **Production.ProductsBeverages** . Then filter the results to only display products with `supplierid = 1` .

2	Execute the query in step 1 above and compare it with the results shown in the following display:  SQLQuery2.sql - M...IZAL RAHMAN (60))* `SELECT productid, productname FROM Production.ProductsBeverages WHERE supplierid = 1;` 100 % Results Messages productid | productname 1 | 1 | Product HHYDP 2 | 2 | Product RECZE  Results Messages productid | productname 1 | 1 | Product HHYDP 2 | 2 | Product RECZE CRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (63) | TSQL2012 | 00:00:00 | 2 rows

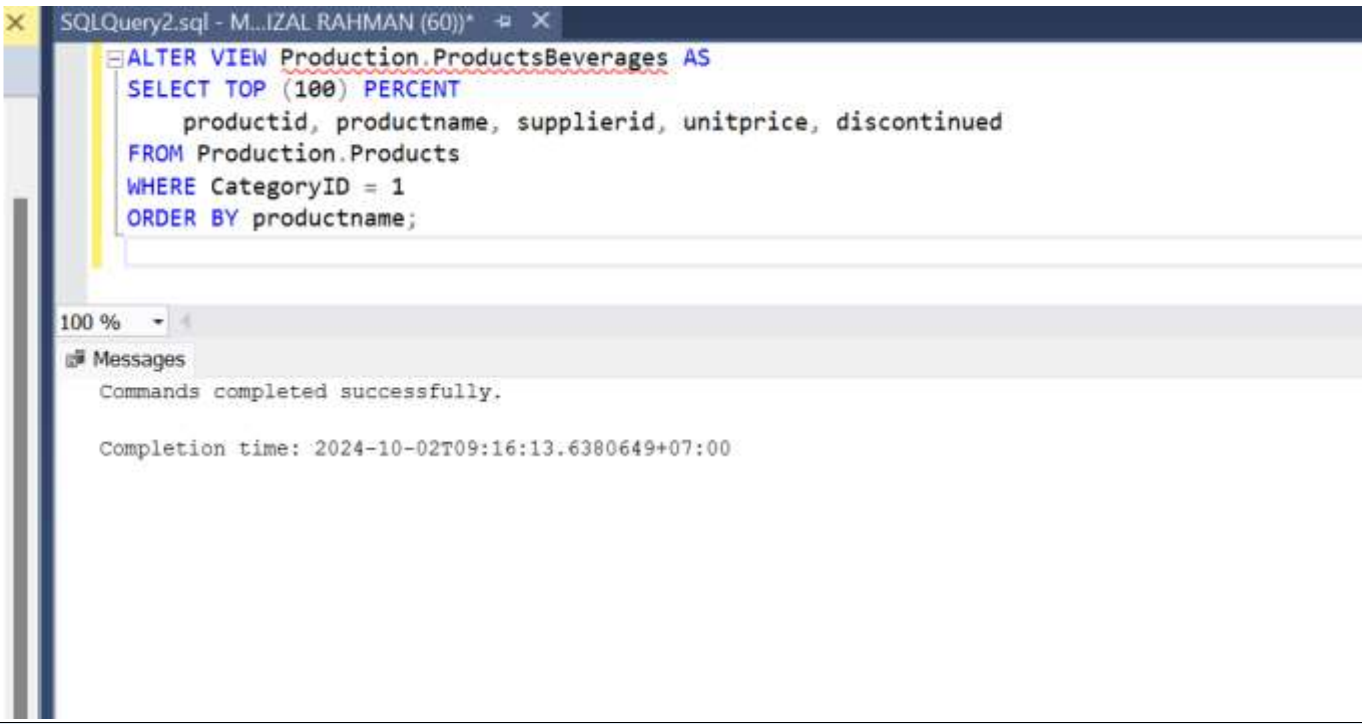
Lab – Part 3: View - Adding an ORDER BY clause to *a VIEW*

Step	Information

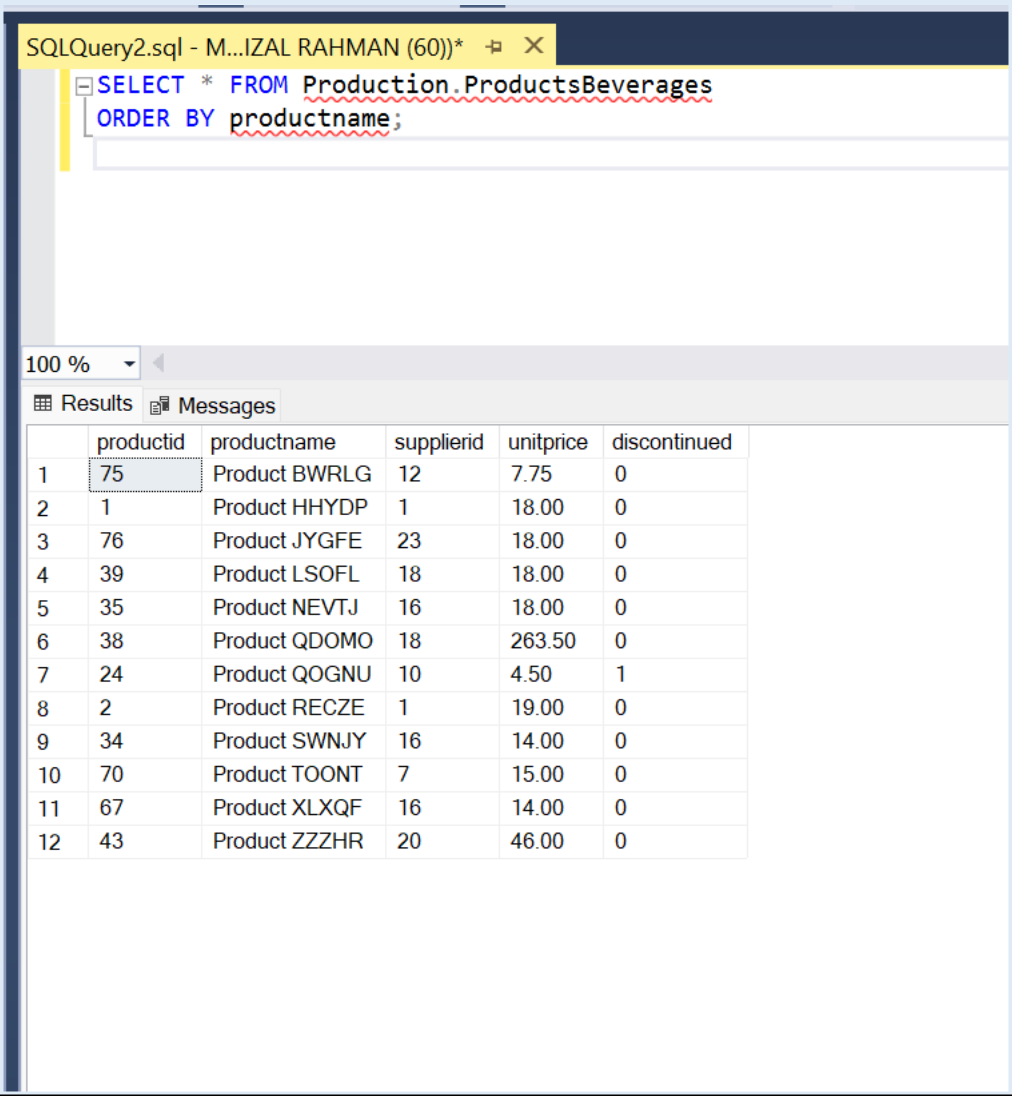
1	Consider the following T-SQL script:
	
	`ALTER VIEW Production . ProductsBeverages AS`
	`SELECT`
	`      productid , product name , supplierid , unit price , discontinued`
	`FROM Production . Products`
	`WHERE Category ID = 1`
	`ORDER BY product name ;`

2	[Question- 4] After executing the T-SQL above, what happens? Write down the error message and explain the cause of the error!
	
	`SQLQuery2.sql - M...IZAL RAHMAN (60))*`
	`ALTER VIEW Production.ProductsBeverages AS`
	`SELECT`
	`   productid , productname , supplierid , unitprice , discontinued`
	`FROM Production . Products`
	`WHERE CategoryID = 1`
	`ORDER BY productname ;`
	`100 %`
	Messages
	`Msg 1033, Level 15, State 1, Procedure ProductsBeverages, Line 6 [Batch Start Line 0]`
	`The ORDER BY clause is invalid in views, inline functions, derived tables, subqueries, and common table expressions, unless TOP, OFFSET or FOR XML is also specifi`
	`Completion time: 2024-10-02T09:17:27.4188751+07:00`
	
	Invalid Usage Without TOP:
	• When defining a view, an ORDER BY clause is not allowed unless it is combined with TOP, OFFSET, or FOR XML. This is why your initial attempt resulted in an error.
	Workaround with TOP (100) PERCENT:
	• Adding TOP (100) PERCENT is a common workaround to include an ORDER BY in a view. While it allows the view definition to succeed, it doesn't ensure that the data will be sorted when queried.
	Unordered Result Sets:
	• Even if you include ORDER BY in the view definition, SQL Server treats views as unordered sets of data. Thus, the rows returned by the view may not reflect the defined order unless explicitly specified in the query that selects from the view.

3	Modify the T-SQL in step 1 above by adding `TOP(100) PERCENT` so that now the query becomes:
	
	`ALTER VIEW Production . ProductsBeverages AS`
	`SELECT TOP ( 100 ) PERCENT`
	`      productid , product name , supplierid , unit price , discontinued`
	`FROM Production . Products`
	`WHERE Category ID = 1`
	`ORDER BY product name ;`

4	Execute the T-SQL in step 3 above and notice that the query has successfully changed the VIEW **Production.ProductsBeverages** even though there is still an ORDER BY clause in the query.
	
	
	
	`ALTER VIEW Production.ProductsBeverages AS`
	`SELECT TOP (100) PERCENT`
	`    productid, productname, supplierid, unitprice, discontinued`
	`FROM Production.Products`
	`WHERE CategoryID = 1`
	`ORDER BY productname;`
	
	`Commands completed successfully.`
	
	`Completion time: 2024-10-02T09:16:13.6380649+07:00`

5	[Question- 5] If a query is run against a modified VIEW **Production.ProductsBeverages , will the rows generated from the VIEW always be sorted by** *productname* ? Explain! **View Definition vs. Query Execution**: • When you define a view in SQL Server, the ORDER BY clause is primarily used for sorting the result set returned by the view's definition. However, this ordering is not guaranteed when the view is queried. SQL Server treats views as sets of data, which are inherently unordered. **ORDER BY in Views**: • The ORDER BY clause in the view definition (even with TOP (100) PERCENT) is more of a syntactic allowance than a directive that enforces sorting. When the view is queried without an explicit ORDER BY, SQL Server may return the results in any order it deems efficient. The database engine does not promise to maintain the order specified in the view.



SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
SELECT * FROM Production.ProductsBeverages
ORDER BY productname;
```

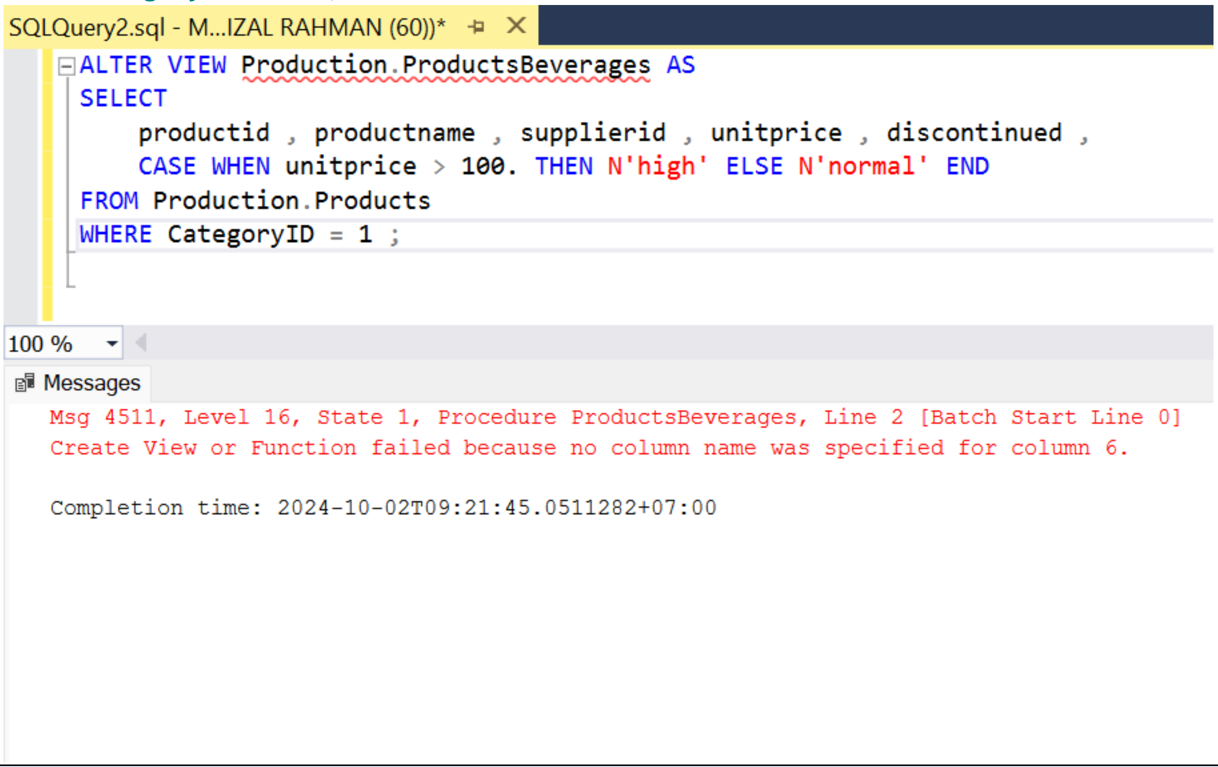
100 %

Results | Messages

	productid	productname	supplierid	unitprice	discontinued
1	75	Product BWRLG	12	7.75	0
2	1	Product HHYDP	1	18.00	0
3	76	Product JYGFE	23	18.00	0
4	39	Product LSOFL	18	18.00	0
5	35	Product NEVTJ	16	18.00	0
6	38	Product QDOMO	18	263.50	0
7	24	Product QOGNU	10	4.50	1
8	2	Product RECZE	1	19.00	0
9	34	Product SWNJY	16	14.00	0
10	70	Product TOONT	7	15.00	0
11	67	Product XLXQF	16	14.00	0
12	43	Product ZZZHR	20	46.00	0

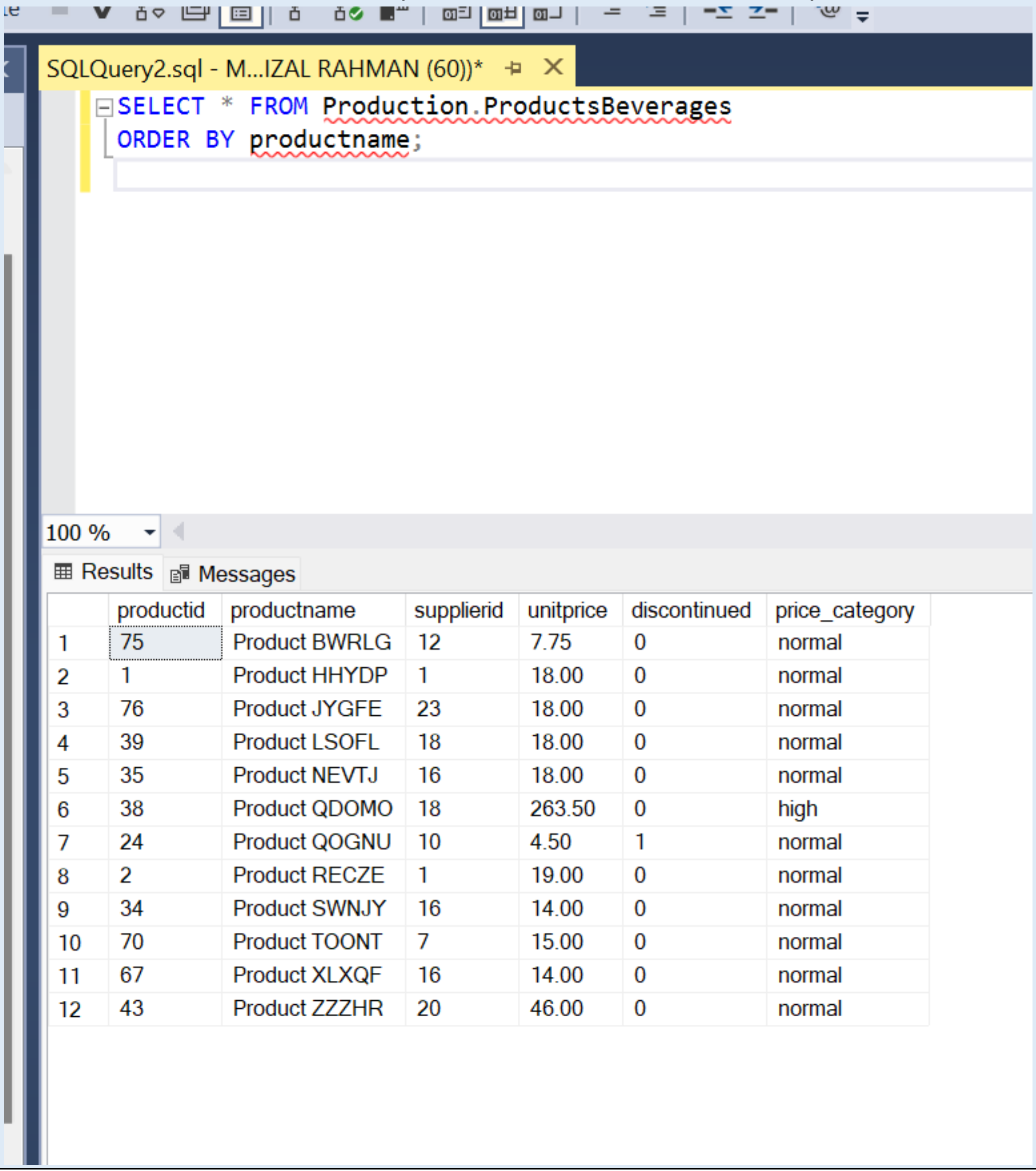
Lab – Part 4: View - Adding columns to *a VIEW*

Step	Information

1	Consider the following T-SQL statement that adds an additional column to the VIEW **Production.ProductsBeverages** that was created in the Practical - Part 1 with the ALTER VIEW command.
	
	`ALTER VIEW Production . ProductsBeverages AS`
	`SELECT`
	`      productid , product name , supplierid , unit price , discontinued ,`
	`      CASE WHEN unit price > 100. THEN N'high' ELSE N'normal' END`
	`FROM Production . Products`
	`WHERE Category ID = 1 ;`
	
	SQLQuery2.sql - M...IZAL RAHMAN (60))*
	`ALTER VIEW Production.ProductsBeverages AS`
	`SELECT`
	`    productid , productname , supplierid , unitprice , discontinued ,`
	`    CASE WHEN unitprice > 100. THEN N'high' ELSE N'normal' END`
	`FROM Production.Products`
	`WHERE CategoryID = 1 ;`
	100 %
	Messages
	`Msg 4511, Level 16, State 1, Procedure ProductsBeverages, Line 2 [Batch Start Line 0]`
	`Create View or Function failed because no column name was specified for column 6.`
	
	`Completion time: 2024-10-02T09:21:45.0511282+07:00`

2	[Question- 6] After executing the T-SQL above, what happens? Write down the error message and explain the cause of the error! **Invalid Column Names**: The error arises because the column names in the SELECT statement contain spaces. SQL Server does not recognize product name, unit price, and Category ID as valid column identifiers due to the presence of spaces. Column names must either not contain spaces or must be enclosed in square brackets. **Syntax Issues**: When column names contain special characters or spaces, they need to be explicitly specified to avoid ambiguity.

3	[Question- 7] Fix the T-SQL script above so that it runs correctly.



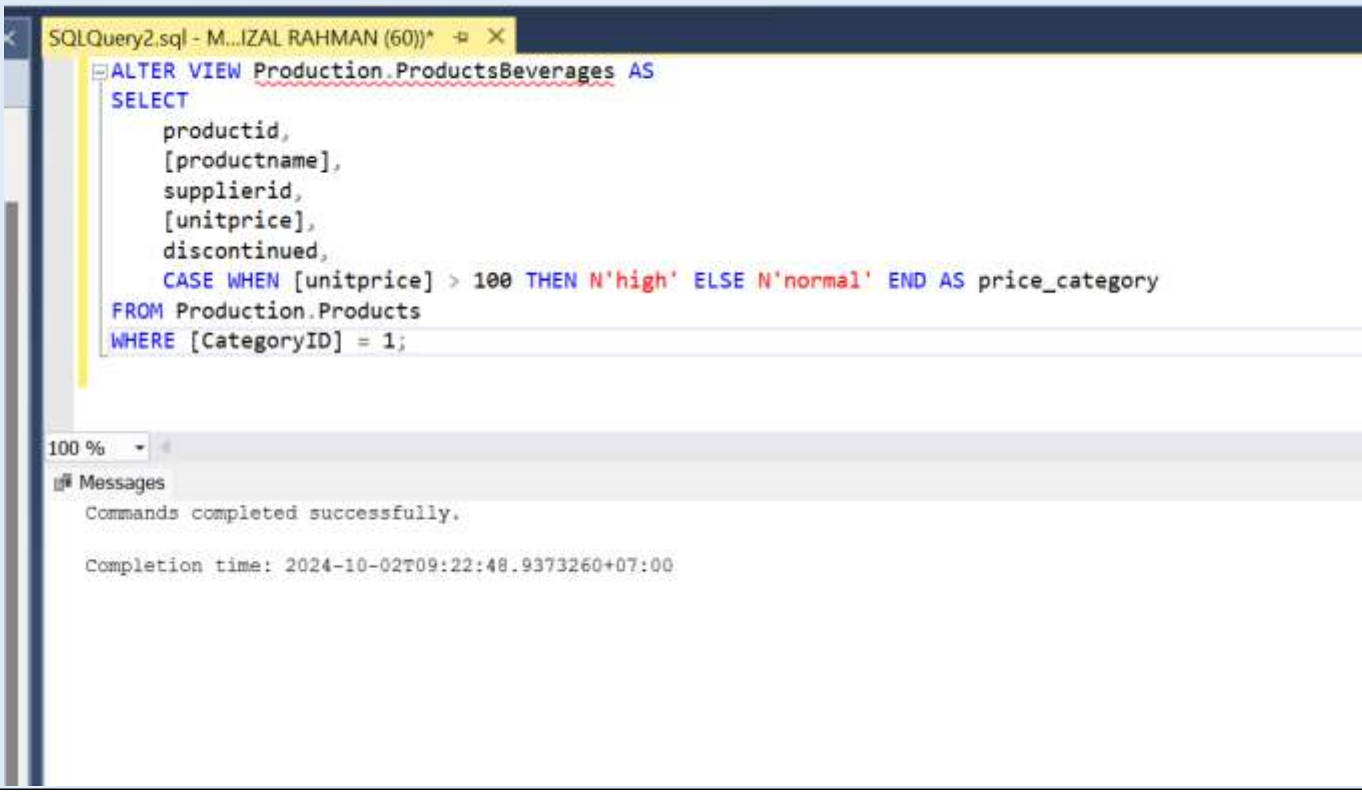
SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
SELECT * FROM Production.ProductsBeverages
ORDER BY productname;
```

100 %

Results | Messages

	productid	productname	supplierid	unitprice	discontinued	price_category
1	75	Product BWRLG	12	7.75	0	normal
2	1	Product HHYDP	1	18.00	0	normal
3	76	Product JYGFE	23	18.00	0	normal
4	39	Product LSOFL	18	18.00	0	normal
5	35	Product NEVTJ	16	18.00	0	normal
6	38	Product QDOMO	18	263.50	0	high
7	24	Product QOGNU	10	4.50	1	normal
8	2	Product RECZE	1	19.00	0	normal
9	34	Product SWNJY	16	14.00	0	normal
10	70	Product TOONT	7	15.00	0	normal
11	67	Product XLXQF	16	14.00	0	normal
12	43	Product ZZZHR	20	46.00	0	normal

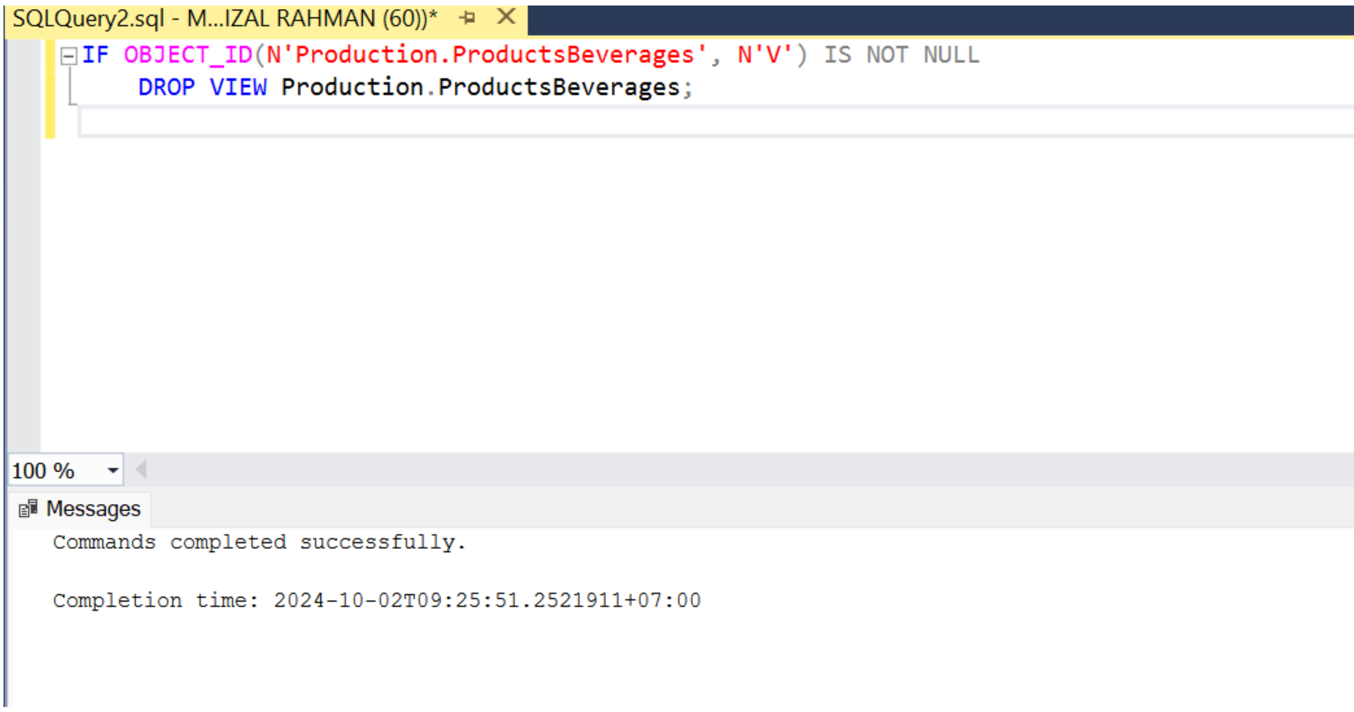
SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
ALTER VIEW Production.ProductsBeverages AS
SELECT
    productid,
    [productname],
    supplierid,
    [unitprice],
    discontinued,
    CASE WHEN [unitprice] > 100 THEN N'high' ELSE N'normal' END AS price_category
FROM Production.Products
WHERE [CategoryID] = 1;
```

```
Commands completed successfully.

Completion time: 2024-10-02T09:22:48.9373260+07:00
```

Lab – Part 5: View - Deleting a VIEW

Step	Information
1	To delete the VIEW **Production.ProductsBeverages** , execute the following T-SQL command: **OBJECT_ID Function**: • The OBJECT_ID function is used to retrieve the object ID for a specified object. In this case, it checks for th named Production.ProductsBeverages. • The first parameter is the name of the object (the view in this case), and the second parameter specifies t the object. Here, N'V' indicates that the object is a view. **IF Condition**: • The IF statement checks if the result of OBJECT_ID is not NULL. If the view exists, OBJECT_ID returns the o if it does not exist, it returns NULL. **DROP VIEW Command**: • If the view exists, the DROP VIEW command is executed to delete the view from the database. `IF OBJECT_ID ( N'Production.ProductsBeverages' , N'V' ) IS NOT NULL        DROP VIEW Production . ProductsBeverages ;`

SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
IF OBJECT_ID(N'Production.ProductsBeverages', N'V') IS NOT NULL
    DROP VIEW Production.ProductsBeverages;
```

Messages

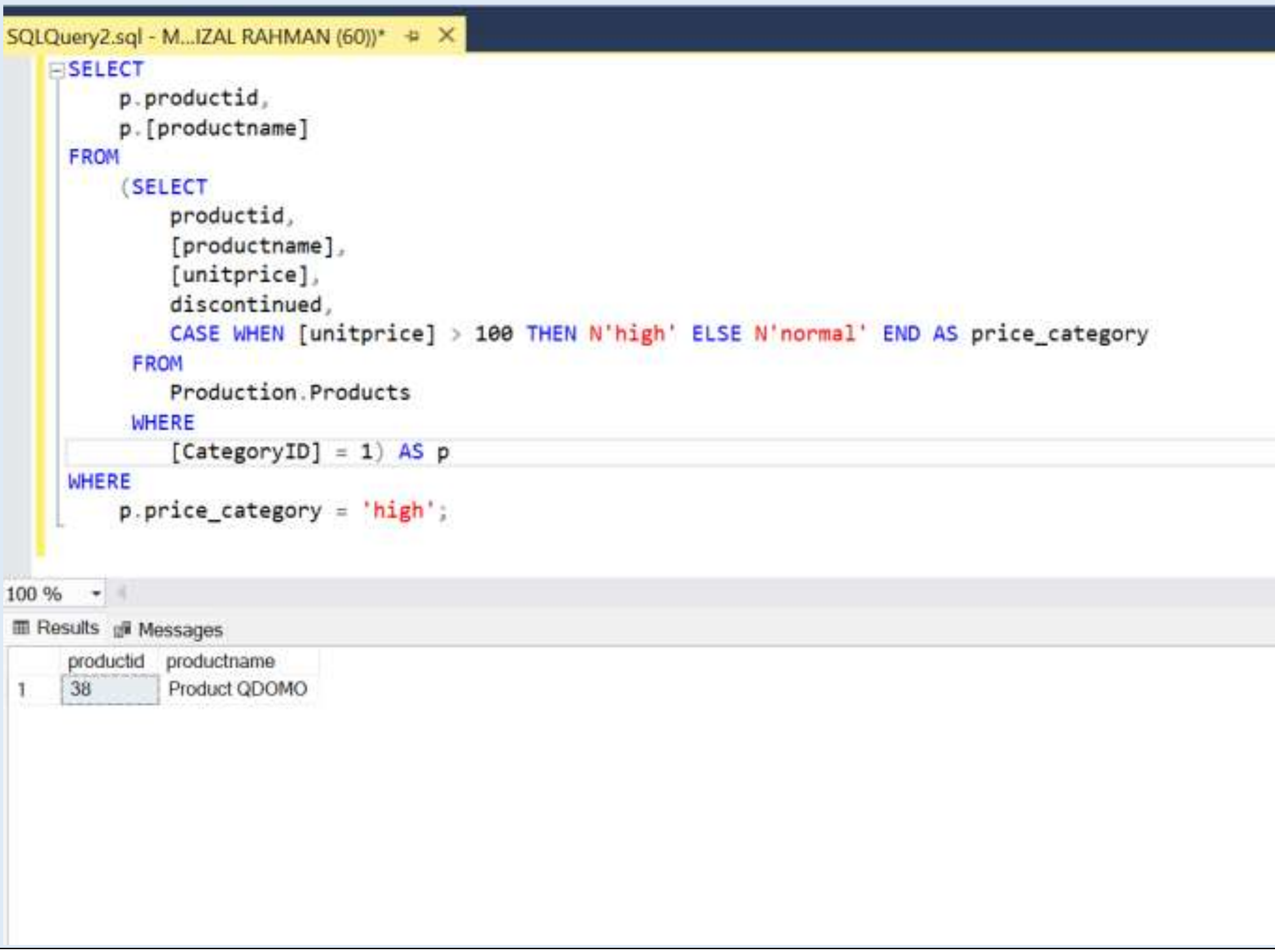
```
Commands completed successfully.

Completion time: 2024-10-02T09:25:51.2521911+07:00
```

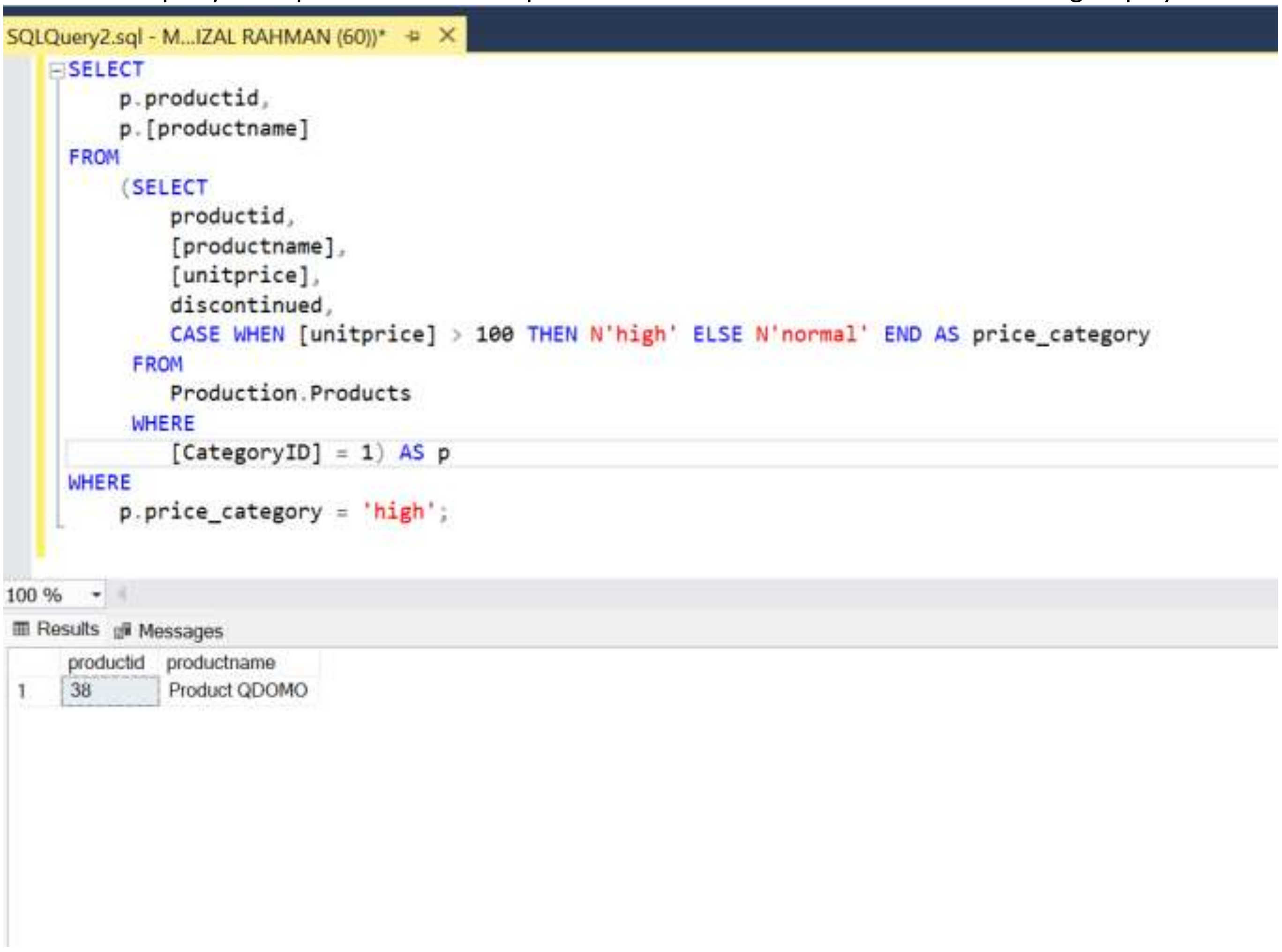
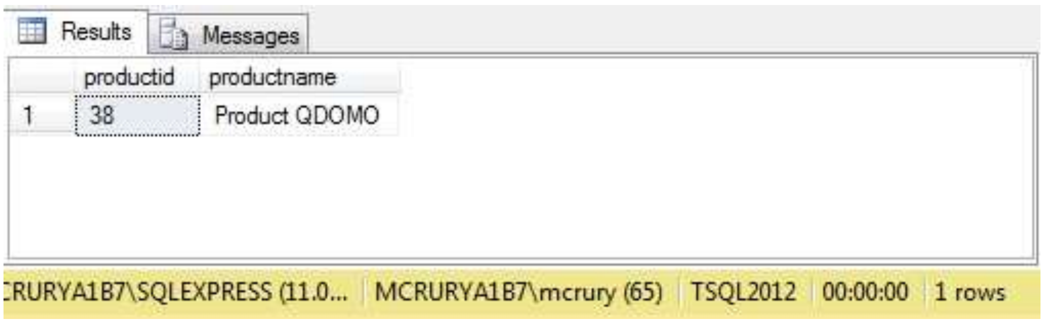
Practical – Part 6: Derived Table - Creating a SELECT query in a derived table

Step	Information

1	[Question-8] Using the TSQL database, create a SELECT query against *the derived table* containing the *productid* and *productname columns* , with a filter to only display data whose *'pricetype'* is 'high'. Use the SELECT query in the Practical - Part 4 - Step 1 as *the derived table* . Give the alias name *p* to the *derived table* .
	SELECT
	   p.productid,
	   p.[productname]
	FROM
	   (SELECT
	      productid,
	      [productname],
	      [unitprice],
	      discontinued,
	      CASE WHEN [unitprice] > 100 THEN N'high' ELSE N'normal' END AS price_category
	    FROM
	      Production.Products

	`WHERE`
	`[CategoryID] = 1) AS p`
	`WHERE`
	`p.price_category = 'high';`

	 SQLQuery2.sql - M...IZAL RAHMAN (60))*
	`SELECT`
	`p.productid,`
	`p.[productname]`
	`FROM`
	`(SELECT`
	`productid,`
	`[productname],`
	`[unitprice],`
	`discontinued,`
	`CASE WHEN [unitprice] > 100 THEN N'high' ELSE N'normal' END AS price_category`
	`FROM`
	`Production.Products`
	`WHERE`
	`[CategoryID] = 1) AS p`
	`WHERE`
	`p.price_category = 'high';`
	100 %
	Results Messages
	`   productid  productname`
	`1  38         Product QDOMO`

2	Execute the query in step 1 above and compare it with the results shown in the following display:  

```
SELECT
    p.productid,
    p.[productname]
FROM
    (SELECT
        productid,
        [productname],
        [unitprice],
        discontinued,
        CASE WHEN [unitprice] > 100 THEN N'high' ELSE N'normal' END AS price_category
     FROM
        Production.Products
     WHERE
        [CategoryID] = 1) AS p
WHERE
    p.price_category = 'high';
```

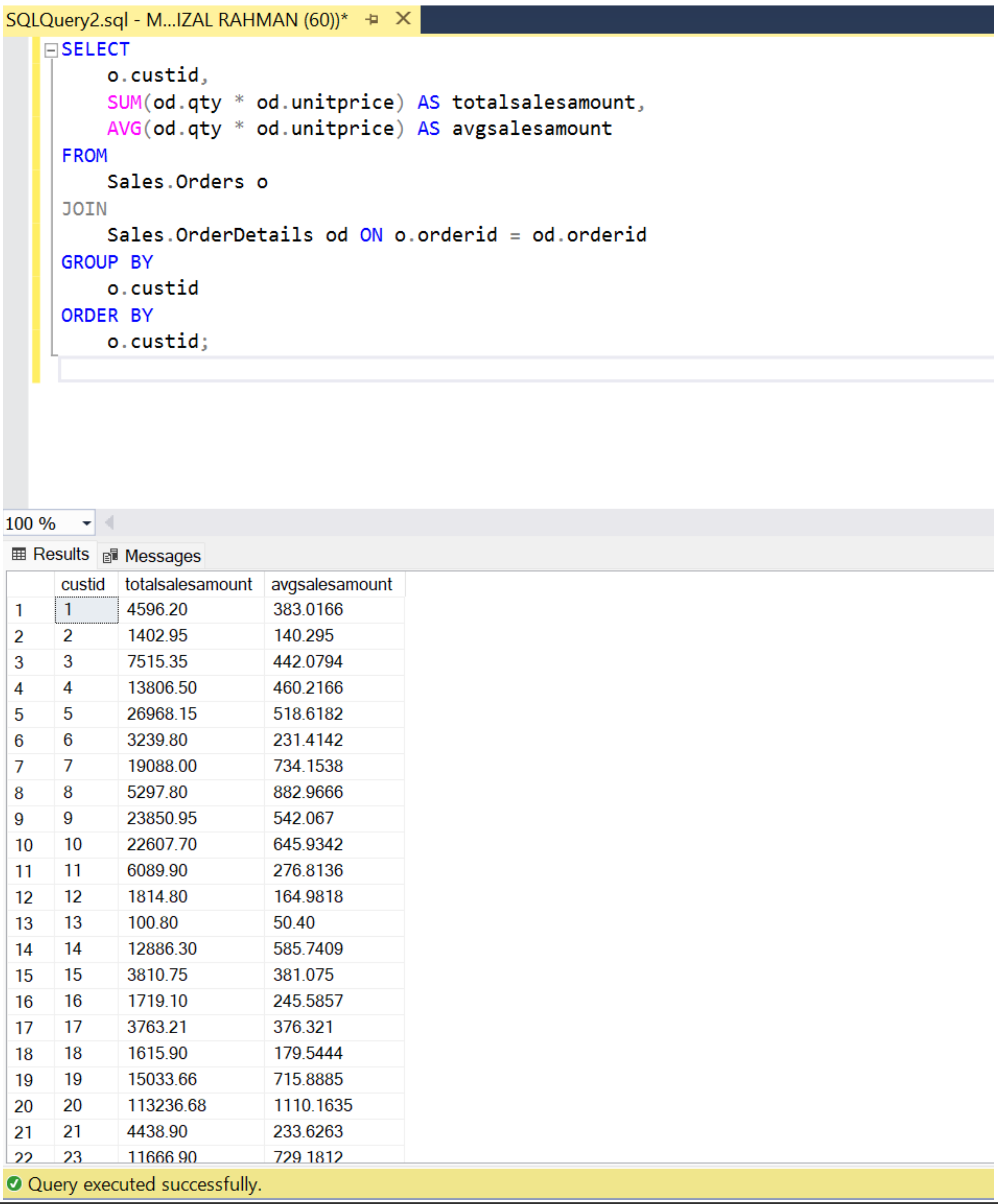
```
   productid  productname
1  38         Product QDOMO
```

```
   productid  productname
1  38         Product QDOMO
```

```
CRURYA1B7\SQLEXPRESS (11.0...  MCRURYA1B7\mcrury (65)  TSQL2012  00:00:00  1 rows
```

Practical – Part 7: Derived Table - Create a SELECT query to find out the total and average number of orders (nominal)

Step	**Information**
1	[Question- 9] Create a SELECT query to get the *custid column* and 2 (two) calculation columns, namely *totalsalesamount* (total nominal amount of orders per customer) and *avgsalesamount* (average nominal amount of orders per customer). To find out the average nominal order per customer, you must first find the total nominal amount per order. The way to do this is by creating a *derived table* that contains a JOIN query between the **Sales.Orders** and **Sales.OrderDetails tables. After that, you can use the** *custid* and *orderid* columns from the **Sales.Orders table** , as well as the *qty* and *unitprice columns* from the **Sales.OrderDetails table** . 1. **SELECT Statement**: o o.custid: Selecting the customer ID. o SUM(od.qty * od.unitprice) AS totalsalesamount: Calculating the total sales amount for each customer by summing the product of quantity (qty) and unit price (unitprice) from the Sales.OrderDetails. o AVG(od.qty * od.unitprice) AS avgsalesamount: Calculating the average sales amount per customer. 2. **FROM Clause**: o Specifies the Sales.Orders table as the primary table (o is an alias for easier reference). 3. **JOIN Clause**: o Joins Sales.OrderDetails (od) on the common orderid to combine the orders with their respective details. 4. **GROUP BY Clause**: o Groups the results by custid to aggregate the total and average sales amounts for each customer. 5. **ORDER BY Clause**: o Sorts the result set by custid for better readability. **Execution and Comparison** You would run the above query on your SQL Server database, and it should provide you with the total and average sales amounts for each customer. You can then compare the results with the values you've provided in your display output.
2	Execute the query in step 1 above and compare it with the results shown in the following display:



SQLQuery2.sql - M...IZAL RAHMAN (60))*

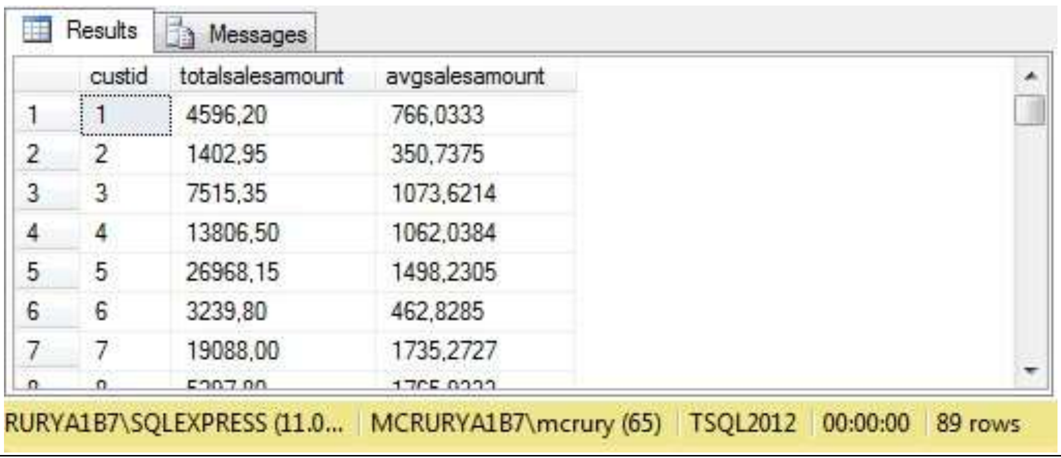
```
SELECT
    o.custid,
    SUM(od.qty * od.unitprice) AS totalsalesamount,
    AVG(od.qty * od.unitprice) AS avgsalesamount
FROM
    Sales.Orders o
JOIN
    Sales.OrderDetails od ON o.orderid = od.orderid
GROUP BY
    o.custid
ORDER BY
    o.custid;
```

100 %

Results | Messages

	custid	totalsalesamount	avgsalesamount
1	1	4596.20	383.0166
2	2	1402.95	140.295
3	3	7515.35	442.0794
4	4	13806.50	460.2166
5	5	26968.15	518.6182
6	6	3239.80	231.4142
7	7	19088.00	734.1538
8	8	5297.80	882.9666
9	9	23850.95	542.067
10	10	22607.70	645.9342
11	11	6089.90	276.8136
12	12	1814.80	164.9818
13	13	100.80	50.40
14	14	12886.30	585.7409
15	15	3810.75	381.075
16	16	1719.10	245.5857
17	17	3763.21	376.321
18	18	1615.90	179.5444
19	19	15033.66	715.8885
20	20	113236.68	1110.1635
21	21	4438.90	233.6263
22	23	11666.90	729.1812

Query executed successfully.

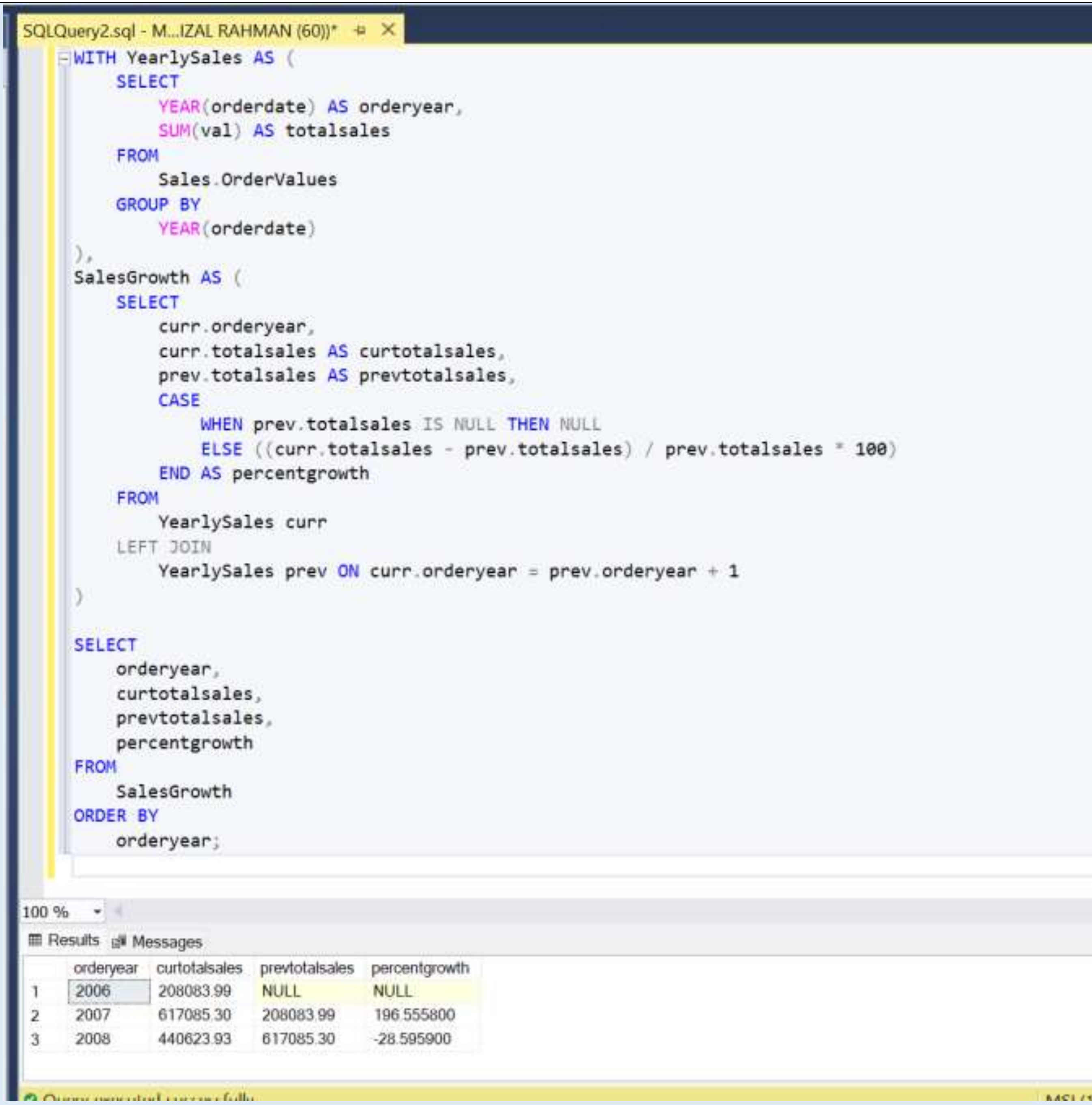
Results | Messages

```
   custid  totalsalesamount  avgsalesamount
1  1       4596,20           766,0333
2  2       1402,95           350,7375
3  3       7515,35           1073,6214
4  4       13806,50          1062,0384
5  5       26968,15          1498,2305
6  6       3239,80           462,8285
7  7       19088,00          1735,2727
```

RURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (65) | TSQL2012 | 00:00:00 | 89 rows

<u>Practical – Part 8: Derived Table - Create a SELECT query to get the sales growth percentage</u>

Step	Information
1	[Question- 10] Write a SELECT query that contains the following columns: - *orderyear* : year from order date - *curtotalsales* : total amount of sales in the year - *prevtotalsales* : total sales amount in the previous year - *percentgrowth* : percentage of sales growth from the current year compared to the previous year

SQLQuery2.sql - M...IZAL RAHMAN (60))*

```sql
WITH YearlySales AS (
    SELECT
        YEAR(orderdate) AS orderyear,
        SUM(val) AS totalsales
    FROM
        Sales.OrderValues
    GROUP BY
        YEAR(orderdate)
),
SalesGrowth AS (
    SELECT
        curr.orderyear,
        curr.totalsales AS curtotalsales,
        prev.totalsales AS prevtotalsales,
        CASE
            WHEN prev.totalsales IS NULL THEN NULL
            ELSE ((curr.totalsales - prev.totalsales) / prev.totalsales * 100)
        END AS percentgrowth
    FROM
        YearlySales curr
    LEFT JOIN
        YearlySales prev ON curr.orderyear = prev.orderyear + 1
)

SELECT
    orderyear,
    curtotalsales,
    prevtotalsales,
    percentgrowth
FROM
    SalesGrowth
ORDER BY
    orderyear;
```

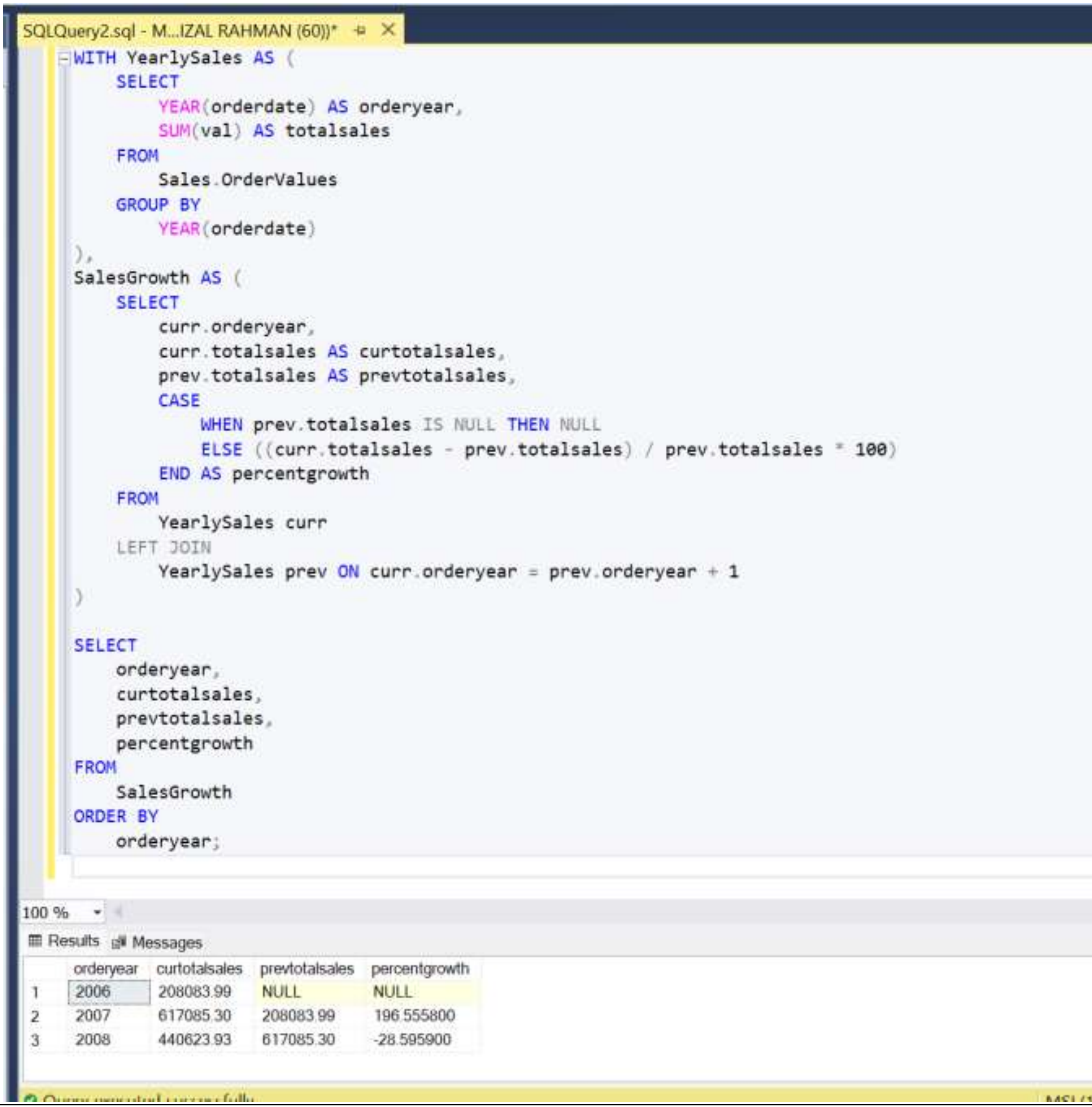
100 %

Results   Messages

```
   orderyear  curtotalsales  prevtotalsales  percentgrowth
1  2006       208083.99      NULL            NULL
2  2007       617085.30      208083.99       196.555800
3  2008       440623.93      617085.30       -28.595900
```

2	You need to create a T-SQL query using 2 (two) *derived tables* . To get the year and total sales for each SELECT query, you can use the existing VIEW named **Sales.OrderValues** . In that view, the *val column* represents the sales amount.

3	It should be noted that in the TSQL database, 2006 is the earliest order year (there are no previous years), but the query can still be executed. ⍰ **Common Table Expressions (CTEs)**: • **YearlySales**: This CTE calculates the total sales amount for each order year by grouping the sales values from the Sales.OrderValues view. • **SalesGrowth**: This CTE calculates the current year's total sales (curtotalsales) and the previous year's tota sales (prevtotalsales). The growth percentage is calculated only when there is a previous year's sales data ⍰ **LEFT JOIN**: The LEFT JOIN between the current and previous year's totals allows you to include years with no previous data (like 2006). ⍰ **Final SELECT**: The final SELECT statement retrieves the required columns and orders the results by orderyear.
4	Execute the query in step 1 above and compare it with the results shown in the following display:



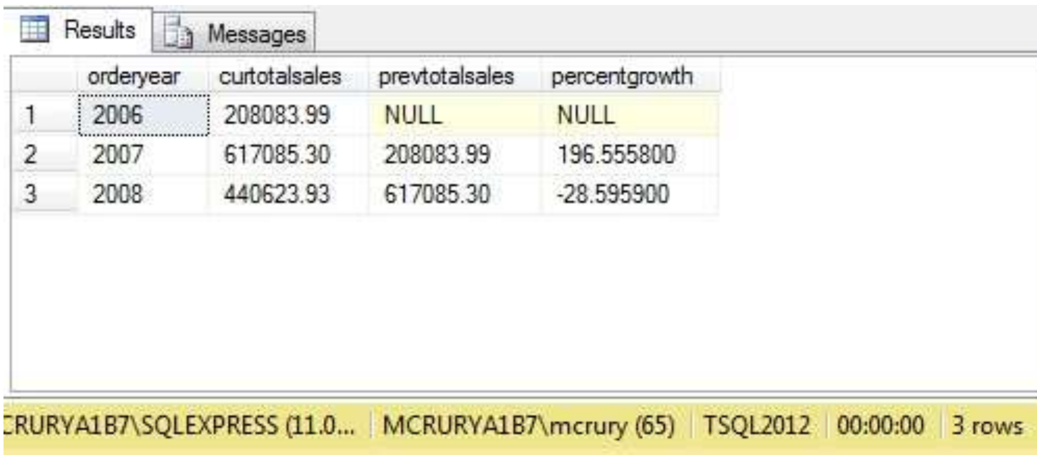
SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
WITH YearlySales AS (
    SELECT
        YEAR(orderdate) AS orderyear,
        SUM(val) AS totalsales
    FROM
        Sales.OrderValues
    GROUP BY
        YEAR(orderdate)
),
SalesGrowth AS (
    SELECT
        curr.orderyear,
        curr.totalsales AS curtotalsales,
        prev.totalsales AS prevtotalsales,
        CASE
            WHEN prev.totalsales IS NULL THEN NULL
            ELSE ((curr.totalsales - prev.totalsales) / prev.totalsales * 100)
        END AS percentgrowth
    FROM
        YearlySales curr
    LEFT JOIN
        YearlySales prev ON curr.orderyear = prev.orderyear + 1
)

SELECT
    orderyear,
    curtotalsales,
    prevtotalsales,
    percentgrowth
FROM
    SalesGrowth
ORDER BY
    orderyear;
```

100 %

Results | Messages

	orderyear	curtotalsales	prevtotalsales	percentgrowth
1	2006	208083.99	NULL	NULL
2	2007	617085.30	208083.99	196.555800
3	2008	440623.93	617085.30	-28.595900

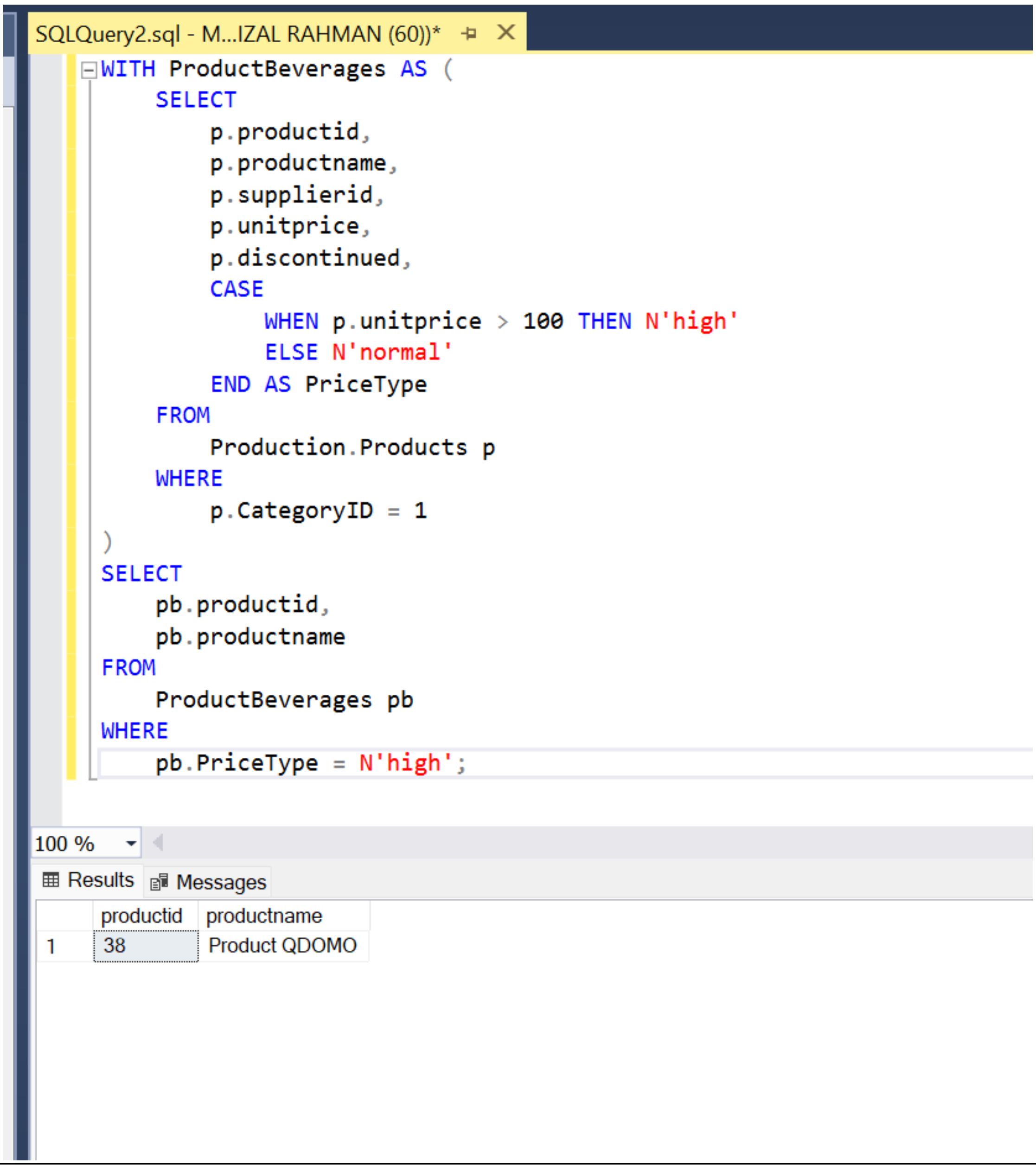
Results | Messages

```
   orderyear  curtotalsales  prevtotalsales  percentgrowth
1  2006       208083.99      NULL            NULL
2  2007       617085.30      208083.99       196.555800
3  2008       440623.93      617085.30       -28.595900
```

```
CRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (65) | TSQL2012 | 00:00:00 | 3 rows
```

Practical – Part 9 : CTE - Creating a SELECT query using CTE

Step	Information
1	[Question-11] While still using the TSQL database, create a SELECT query like in the Practical - Part 6 , but using Common Table Expressions (CTE). Name the CTE query alias as **ProductBeverages** . **CTE Definition**: The ProductBeverages CTE retrieves product IDs and names for products categorized as "Beverages." Adjust the filtering condition based on your requirements. **Final SELECT**: The final SELECT retrieves product IDs and names from the ProductBeverages CTE.
2	Execute the query in step 1 and compare it with the results shown in the following display:

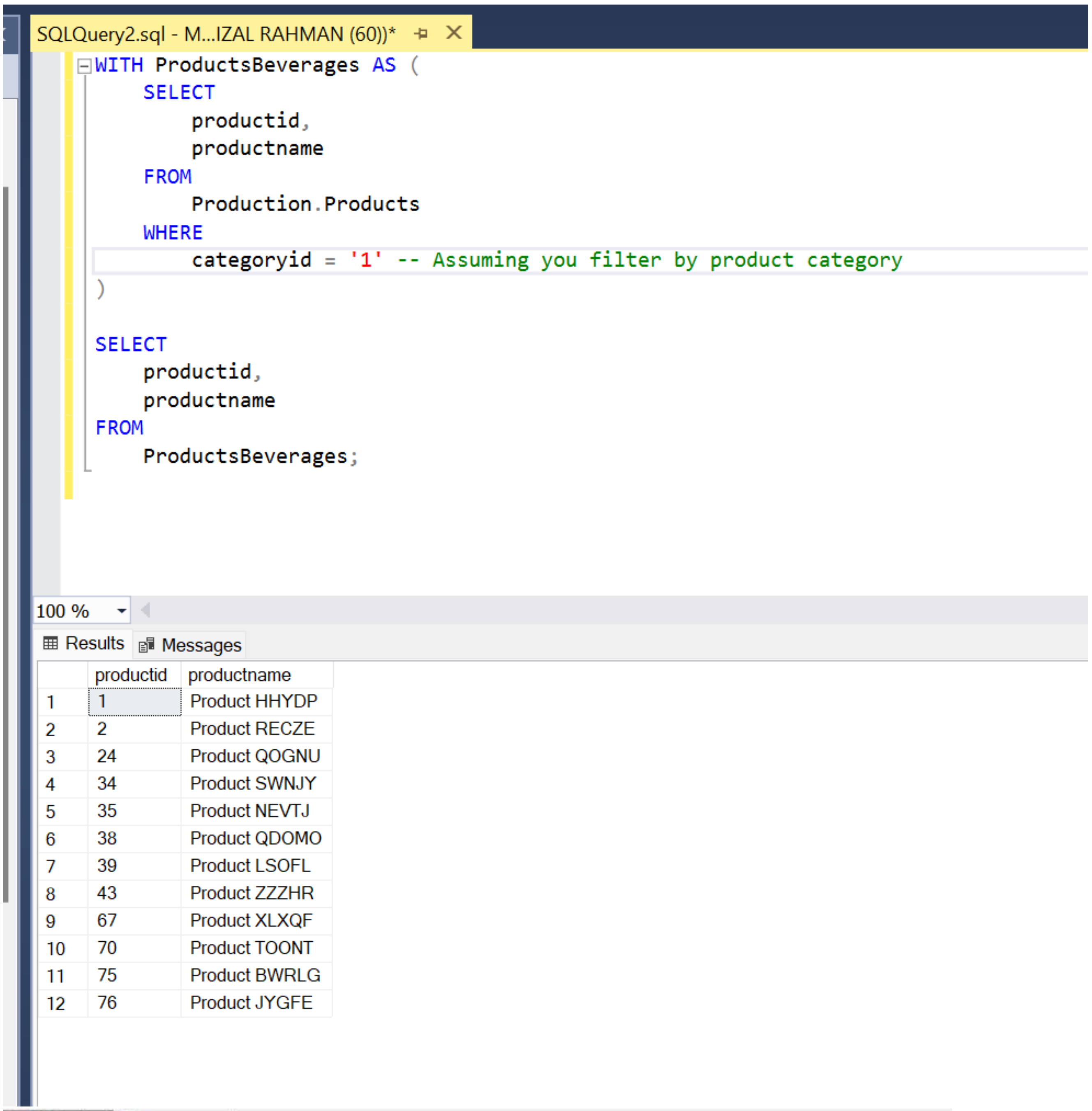
SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
WITH ProductBeverages AS (
    SELECT
        p.productid,
        p.productname,
        p.supplierid,
        p.unitprice,
        p.discontinued,
        CASE
            WHEN p.unitprice > 100 THEN N'high'
            ELSE N'normal'
        END AS PriceType
    FROM
        Production.Products p
    WHERE
        p.CategoryID = 1
)
SELECT
    pb.productid,
    pb.productname
FROM
    ProductBeverages pb
WHERE
    pb.PriceType = N'high';
```

100 %

Results | Messages

	productid	productname
1	38	Product QDOMO

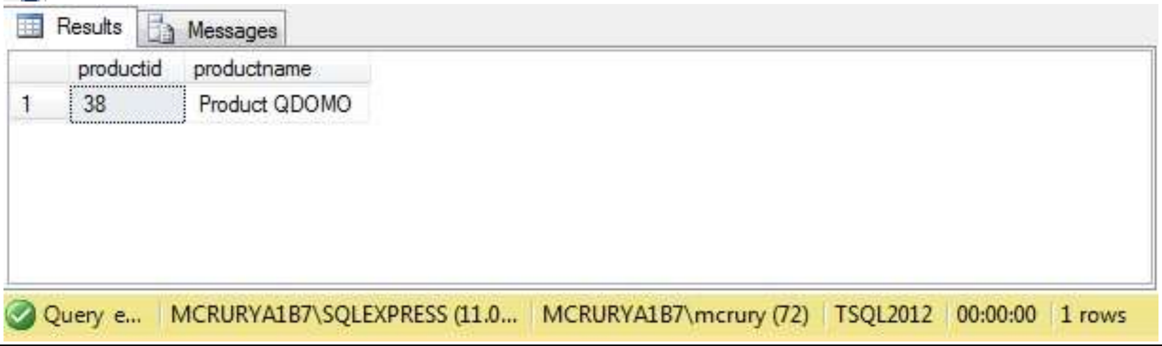

SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
WITH ProductsBeverages AS (
    SELECT
        productid,
        productname
    FROM
        Production.Products
    WHERE
        categoryid = '1' -- Assuming you filter by product category
)

SELECT
    productid,
    productname
FROM
    ProductsBeverages;
```

100 %

Results | Messages

	productid	productname
1	1	Product HHYDP
2	2	Product RECZE
3	24	Product QOGNU
4	34	Product SWNJY
5	35	Product NEVTJ
6	38	Product QDOMO
7	39	Product LSOFL
8	43	Product ZZZHR
9	67	Product XLXQF
10	70	Product TOONT
11	75	Product BWRLG
12	76	Product JYGFE

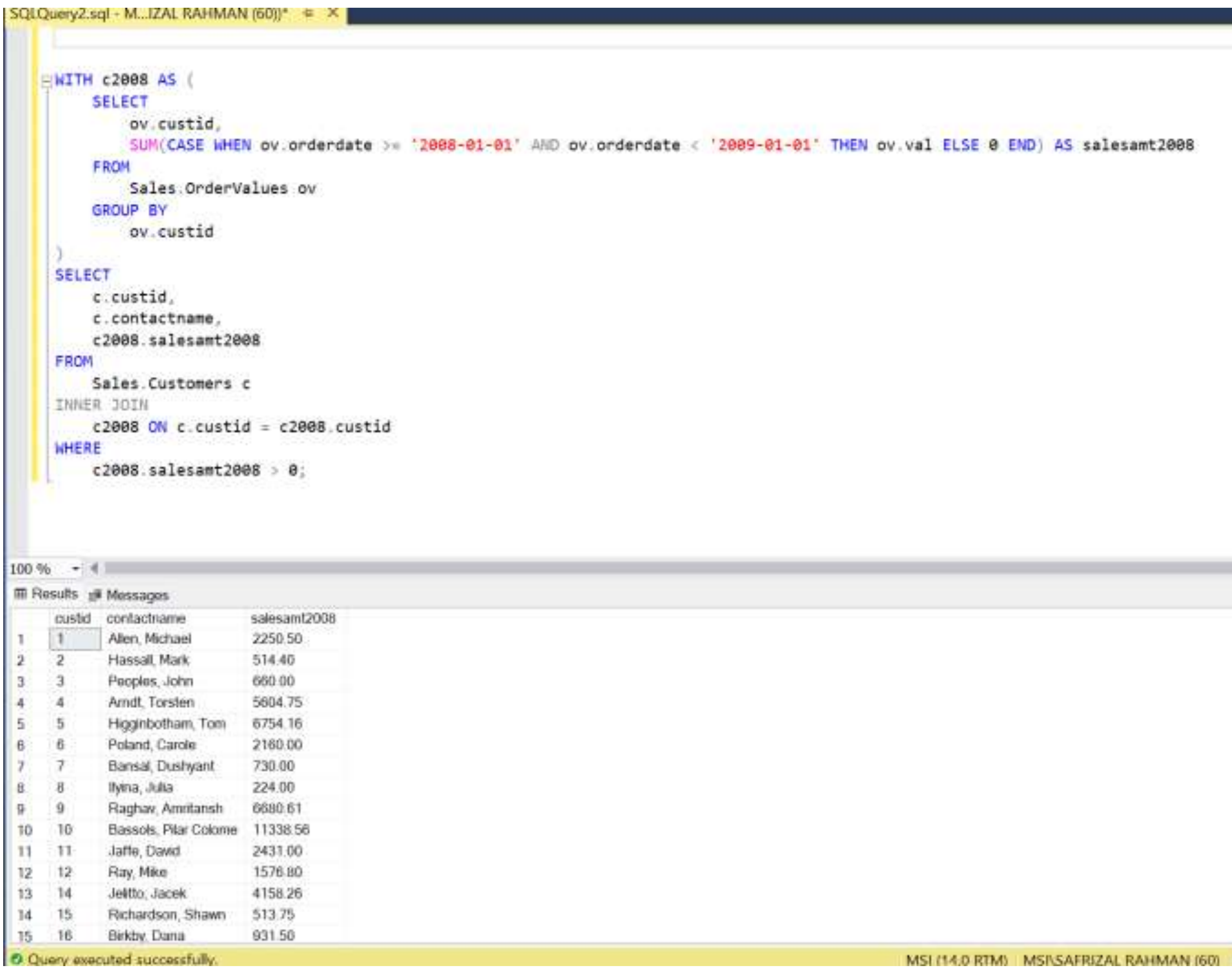

Results | Messages

	productid	productname
1	38	Product QDOMO

Query e... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (72) | TSQL2012 | 00:00:00 | 1 rows

Practical – Part 10 : CTE - Create a SELECT query to get the total sales amount (nominal) for each customer.

Step	Information
1	[Question-12] Create a SELECT query against the **Sales.OrderValues view** to get the customer ID and total sales amount in 2008. Name this CTE as **c2008** , which consists of the *custid* and *salesamt2008 columns.*
	Then, perform a JOIN operation between the **Sales.Customers table** and the CTE c2008, resulting in the *custid* and *contactname columns* from the **Sales.Customer** table and *the salesamt2008 column* from the CTE **c2008** .
2	Execute the query in step 1 above and compare it with the results shown in the following display:



SQLQuery2.sql - M...IZAL RAHMAN (60))*

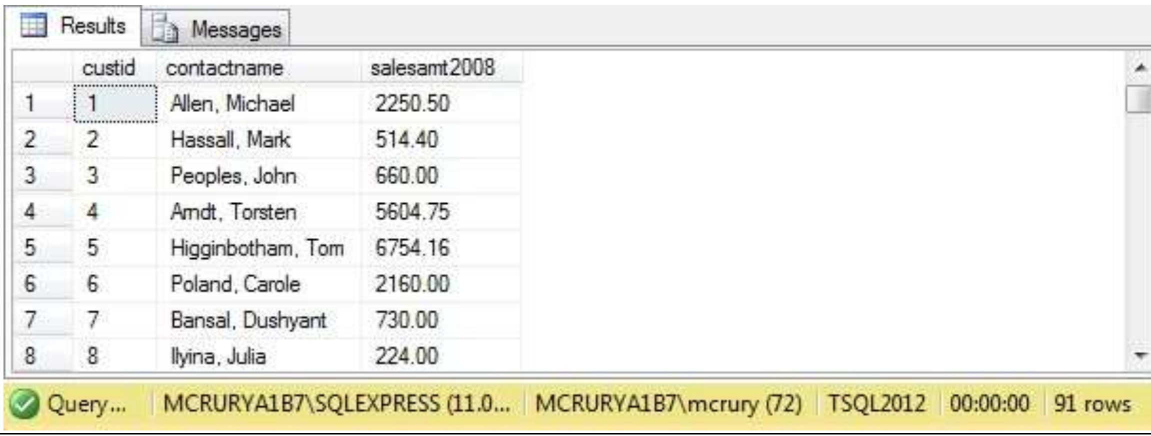
```sql
WITH c2008 AS (
    SELECT
        ov.custid,
        SUM(CASE WHEN ov.orderdate >= '2008-01-01' AND ov.orderdate < '2009-01-01' THEN ov.val ELSE 0 END) AS salesamt2008
    FROM
        Sales.OrderValues ov
    GROUP BY
        ov.custid
)
SELECT
    c.custid,
    c.contactname,
    c2008.salesamt2008
FROM
    Sales.Customers c
INNER JOIN
    c2008 ON c.custid = c2008.custid
WHERE
    c2008.salesamt2008 > 0;
```

100 %

Results Messages

```
     custid  contactname          salesamt2008
1    1       Allen, Michael       2250.50
2    2       Hassall, Mark        514.40
3    3       Peoples, John        660.00
4    4       Arndt, Torsten       5604.75
5    5       Higginbotham, Tom    6754.16
6    6       Poland, Carole       2160.00
7    7       Bansal, Dushyant     730.00
8    8       Ilyina, Julia        224.00
9    9       Raghav, Amritansh    6680.61
10   10      Bassols, Pilar Colome 11338.56
11   11      Jaffe, David         2431.00
12   12      Ray, Mike            1576.80
13   14      Jelitto, Jacek       4158.26
14   15      Richardson, Shawn    513.75
15   16      Birkby, Dana         931.50
```

Query executed successfully. MSI (14.0 RTM) MSI\SAFRIZAL RAHMAN (60)



Results Messages

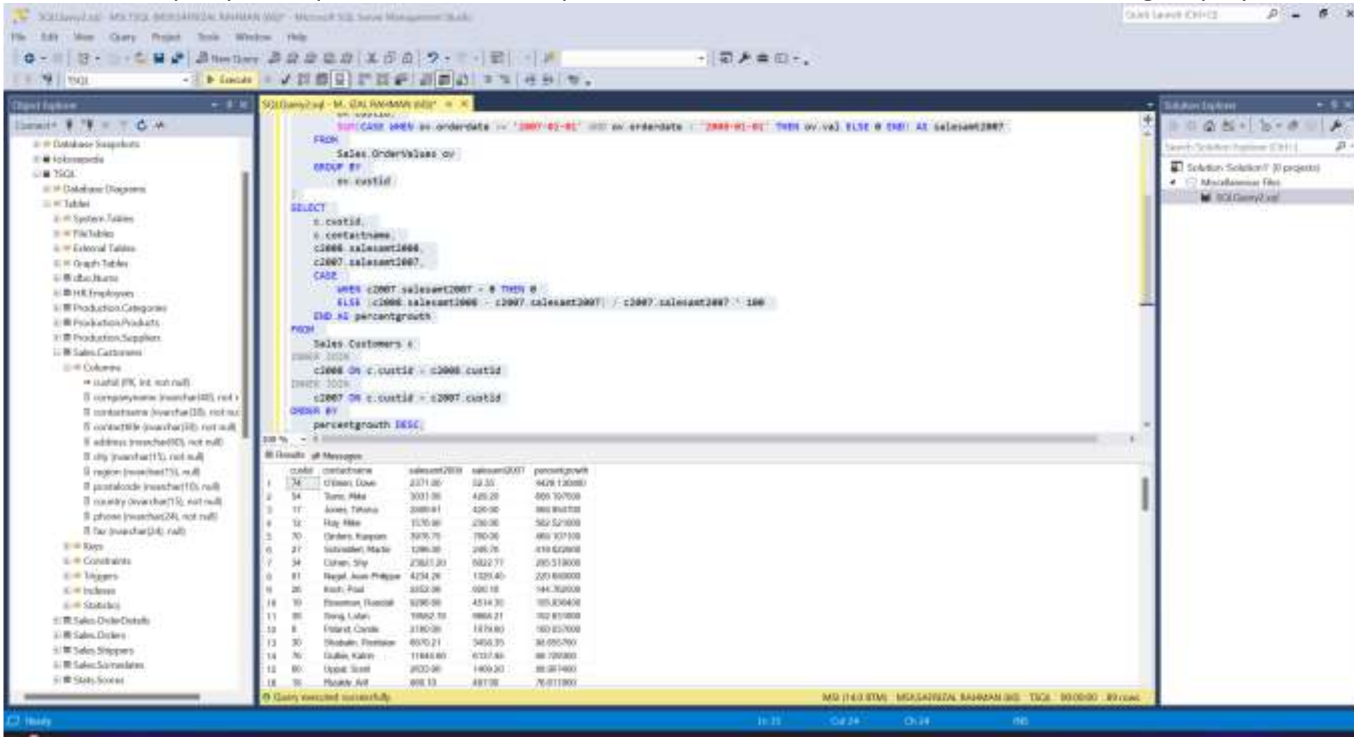
```
     custid  contactname          salesamt2008
1    1       Allen, Michael       2250.50
2    2       Hassall, Mark        514.40
3    3       Peoples, John        660.00
4    4       Arndt, Torsten       5604.75
5    5       Higginbotham, Tom    6754.16
6    6       Poland, Carole       2160.00
7    7       Bansal, Dushyant     730.00
8    8       Ilyina, Julia        224.00
```

Query... MCRURYA1B7\SQLEXPRESS (11.0... MCRURYA1B7\mcrury (72) TSQL2012 00:00:00 91 rows

<u>Practical – Part 11 : CTE - Create a SELECT query to compare the total sales amount for each customer with the previous year.</u>

Step	Information
1	[Question- 13] Create a SELECT query containing the *custid* and *contactname columns* against the **Sales.Customers table** . Also, get the values for the following columns: - *salesamt2008* : total sales amount in 2008 - *salesamt2007* : total sales amount in 2007 - *percentgrowth* : percentage growth in sales between 2007 and 2008 If *percentgrowth* returns NULL, display it as 0. You can use the CTE from Lab Part 10 and create another CTE for the year 2007. Then, perform a JOIN operation between the two CTEs with the **Sales.Customers table** . Sort the results by the *percentgrowth column.*

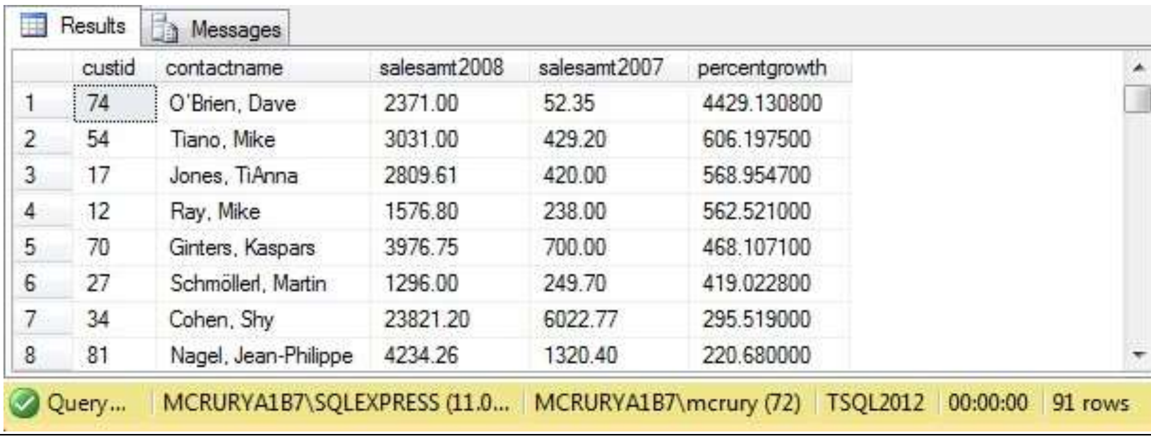
```
WITH c2008 AS (
    SELECT
        ov.custid,
        SUM(CASE WHEN ov.orderdate >= '2008-01-01' AND ov.orderdate < '2009-01-01' THEN ov.val ELSE 0 END) AS
salesamt2008
    FROM
        Sales.OrderValues ov
    GROUP BY
        ov.custid
),
c2007 AS (
    SELECT
        ov.custid,
        SUM(CASE WHEN ov.orderdate >= '2007-01-01' AND ov.orderdate < '2008-01-01' THEN ov.val ELSE 0 END) AS
salesamt2007
    FROM
        Sales.OrderValues ov
    GROUP BY
        ov.custid
)
SELECT
    c.custid,
    c.contactname,
    c2008.salesamt2008,
    c2007.salesamt2007,
    CASE
        WHEN c2007.salesamt2007 = 0 THEN 0
        ELSE (c2008.salesamt2008 - c2007.salesamt2007) / c2007.salesamt2007 * 100
    END AS percentgrowth
FROM
```

	Sales.Customers c INNER JOIN c2008 ON c.custid = c2008.custid INNER JOIN c2007 ON c.custid = c2007.custid ORDER BY percentgrowth DESC;
2	Execute the query in step 1 above and compare it with the results shown in the following display: 

```
        SUM(CASE WHEN ov.orderdate >= '2007-01-01' AND ov.orderdate < '2008-01-01' THEN ov.val ELSE 0 END) AS salesamt2007
    FROM
        Sales.OrderValues ov
    GROUP BY
        ov.custid
)
SELECT
    c.custid,
    c.contactname,
    c2008.salesamt2008,
    c2007.salesamt2007,
    CASE
        WHEN c2007.salesamt2007 = 0 THEN 0
        ELSE (c2008.salesamt2008 - c2007.salesamt2007) / c2007.salesamt2007 * 100
    END AS percentgrowth
FROM
    Sales.Customers c
INNER JOIN
    c2008 ON c.custid = c2008.custid
INNER JOIN
    c2007 ON c.custid = c2007.custid
ORDER BY
    percentgrowth DESC;
```

```
     custid  contactname            salesamt2008  salesamt2007  percentgrowth
1    74      O'Brien, Dave          2371.00       32.35         9428.130000
2    54      Tiano, Mike            3031.00       425.20        605.107000
3    17      Jones, TiAnna          2000.61       420.00        380.854000
4    12      Ray, Mike              1576.80       236.00        562.521000
5    70      Ginters, Kaspars       3976.75       780.00        409.107100
6    27      [illegible]            1286.30       248.78        418.622800
7    34      Cohen, Shy             23821.20      5022.77       285.519000
8    81      Nagel, Jean-Philippe   4234.26       1320.40       220.680000
9    26      Koch, Paul             3352.00       [illegible]   144.762000
10   10      [illegible]            9290.50       4514.35       105.836400
11   35      Song, Lolan            19982.10      9864.21       102.611000
12   8       [illegible]            3190.00       1579.80       100.857000
13   30      Shabalin, Rostislav    6970.21       3458.35       98.095700
14   76      Gulbis, Katrin         11644.60      6337.86       88.720000
15   60      Uppal, Sunil           2622.00       1409.20       88.387400
16   55      [illegible]            608.13        487.00        76.011000
```

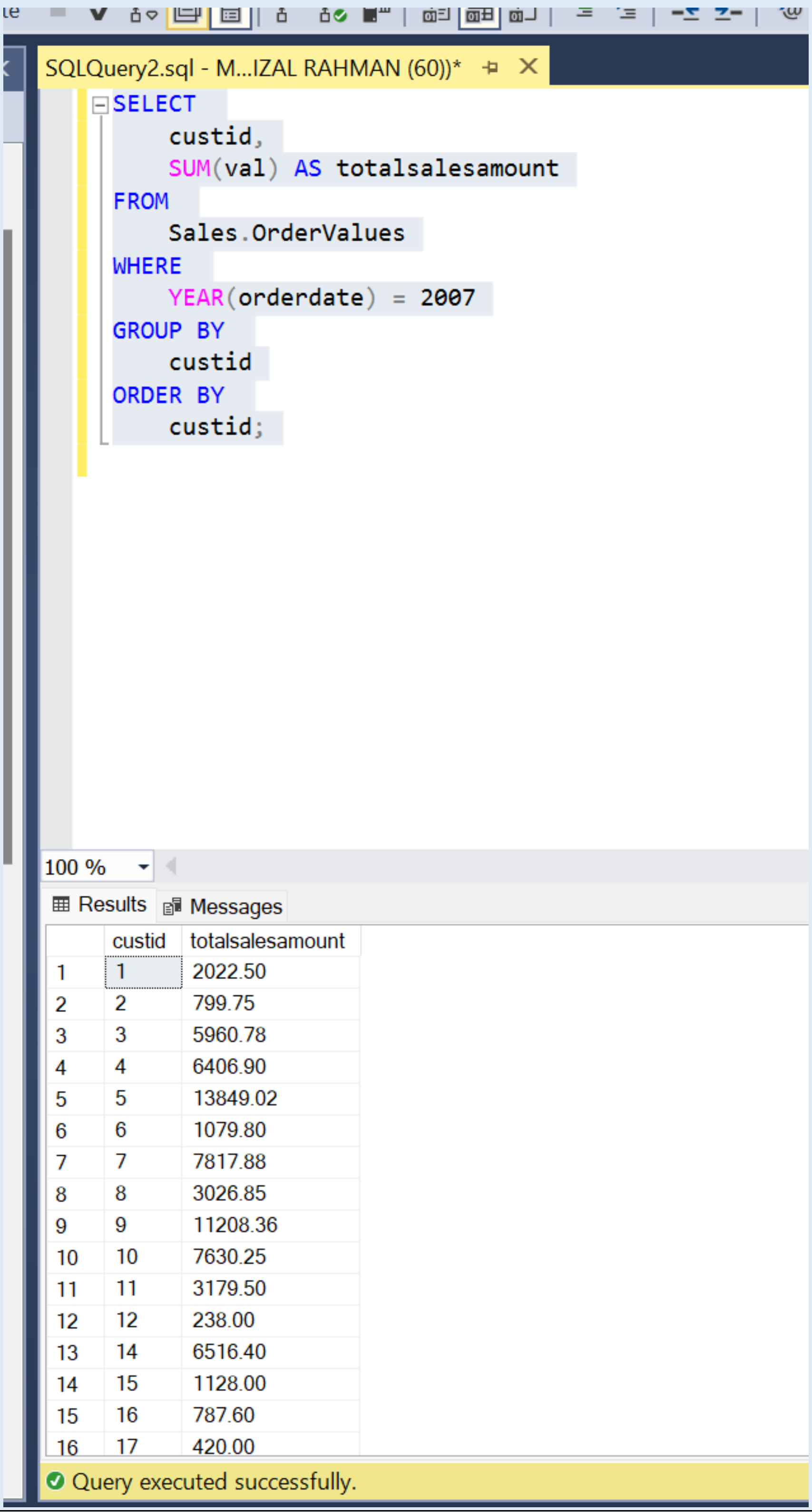
Query executed successfully.

```
   custid  contactname            salesamt2008  salesamt2007  percentgrowth
1  74      O'Brien, Dave          2371.00       52.35         4429.130800
2  54      Tiano, Mike            3031.00       429.20        606.197500
3  17      Jones, TiAnna          2809.61       420.00        568.954700
4  12      Ray, Mike              1576.80       238.00        562.521000
5  70      Ginters, Kaspars       3976.75       700.00        468.107100
6  27      Schmöllerl, Martin     1296.00       249.70        419.022800
7  34      Cohen, Shy             23821.20      6022.77       295.519000
8  81      Nagel, Jean-Philippe   4234.26       1320.40       220.680000

Query...  MCRURYA1B7\SQLEXPRESS (11.0...  MCRURYA1B7\mcrury (72)  TSQL2012  00:00:00  91 rows
```

Practical – Part 12: Inline TVF - Create a SELECT query to get the total sales amount (nominal) for each customer.

Step	Information
1	[Question- 14] Using a TSQL database, create a SELECT query against the **Sales.OrderValues view** that c *totalsalesamount column* (the total of the *val column*). Filter the results to only display orders in 2007.

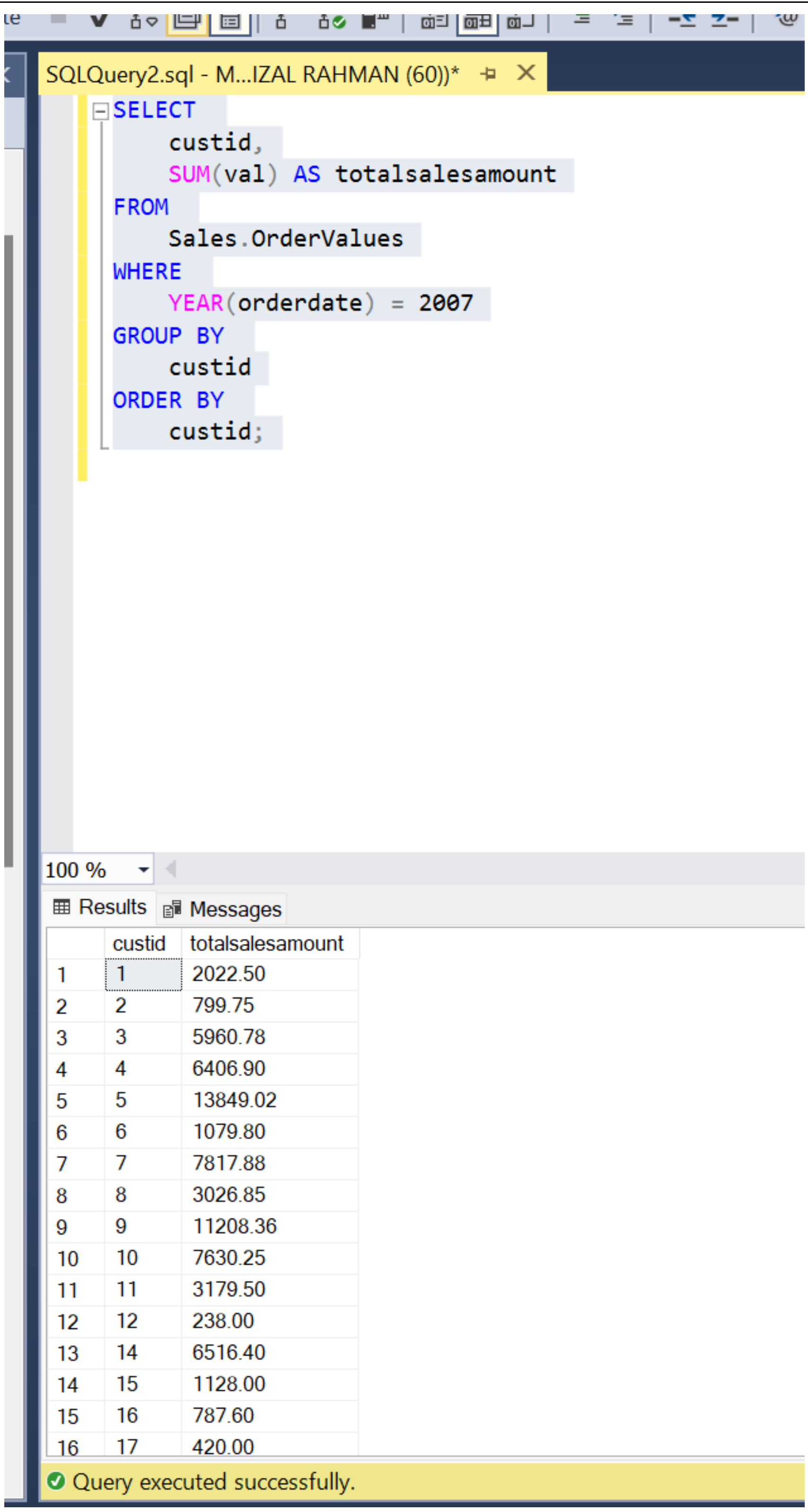


```
SELECT
    custid,
    SUM(val) AS totalsalesamount
FROM
    Sales.OrderValues
WHERE
    YEAR(orderdate) = 2007
GROUP BY
    custid
ORDER BY
    custid;
```

```
    custid  totalsalesamount
1   1       2022.50
2   2       799.75
3   3       5960.78
4   4       6406.90
5   5       13849.02
6   6       1079.80
7   7       7817.88
8   8       3026.85
9   9       11208.36
10  10      7630.25
11  11      3179.50
12  12      238.00
13  14      6516.40
14  15      1128.00
15  16      787.60
16  17      420.00
```

Query executed successfully.

2	Execute the query in step 1 above and compare it with the results shown in the following display:


SQLQuery2.sql - M...IZAL RAHMAN (60))*

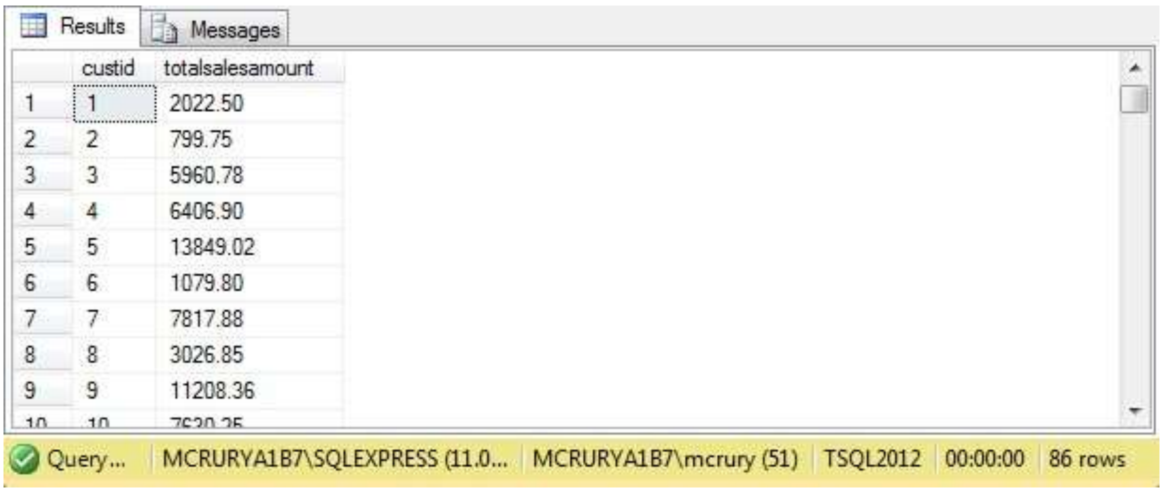
```
SELECT
    custid,
    SUM(val) AS totalsalesamount
FROM
    Sales.OrderValues
WHERE
    YEAR(orderdate) = 2007
GROUP BY
    custid
ORDER BY
    custid;
```

100 %

Results | Messages

	custid	totalsalesamount
1	1	2022.50
2	2	799.75
3	3	5960.78
4	4	6406.90
5	5	13849.02
6	6	1079.80
7	7	7817.88
8	8	3026.85
9	9	11208.36
10	10	7630.25
11	11	3179.50
12	12	238.00
13	14	6516.40
14	15	1128.00
15	16	787.60
16	17	420.00

Query executed successfully.

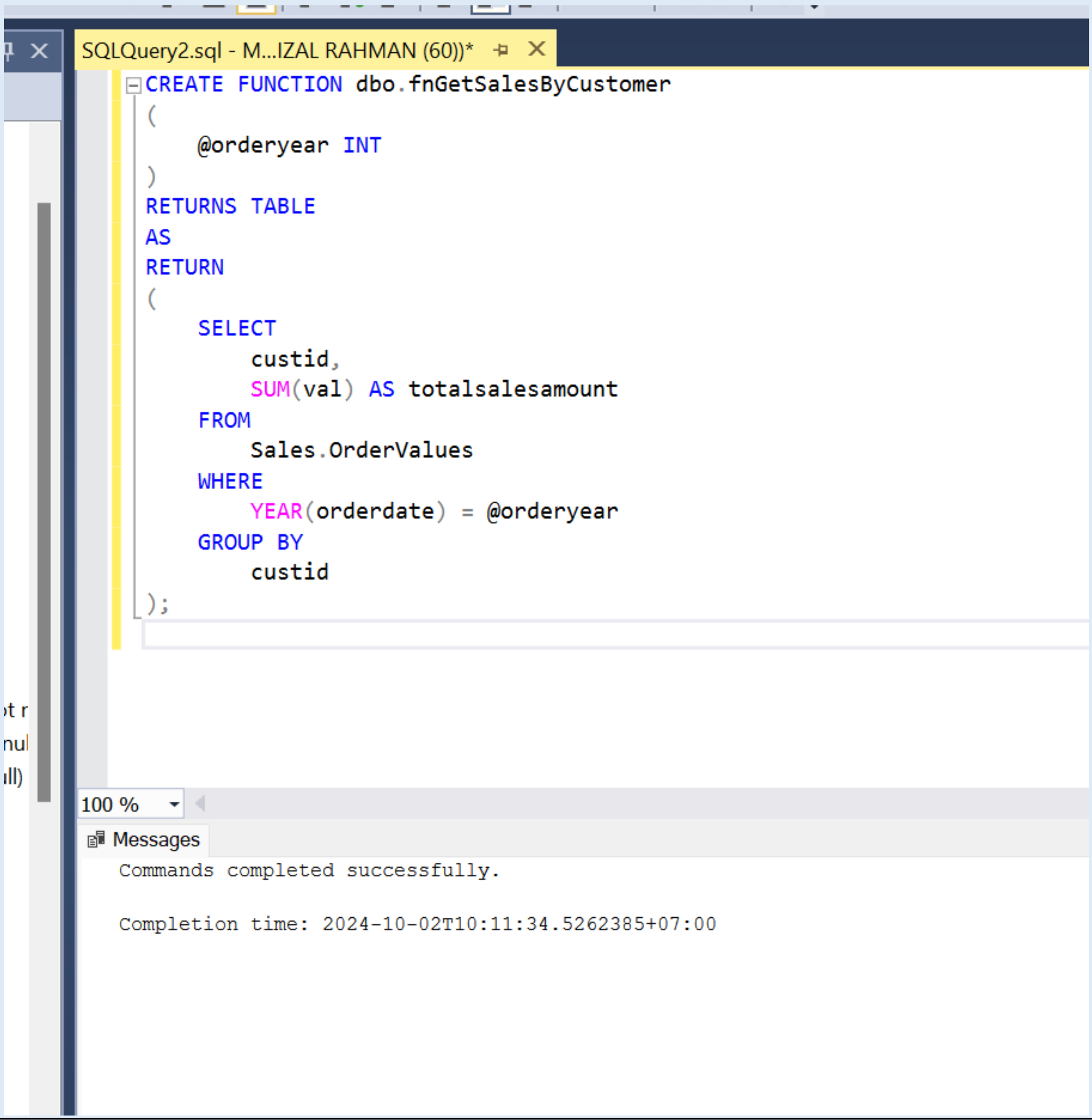
	

3	[Question- 15] Create an inline TVF/Table-Valued Function by adding the following line and placing it be
	
	`CREATE FUNCTION dbo . fnGetSalesByCustomer`
	`( @orderyear US INT ) RETURNS TABLE AS`

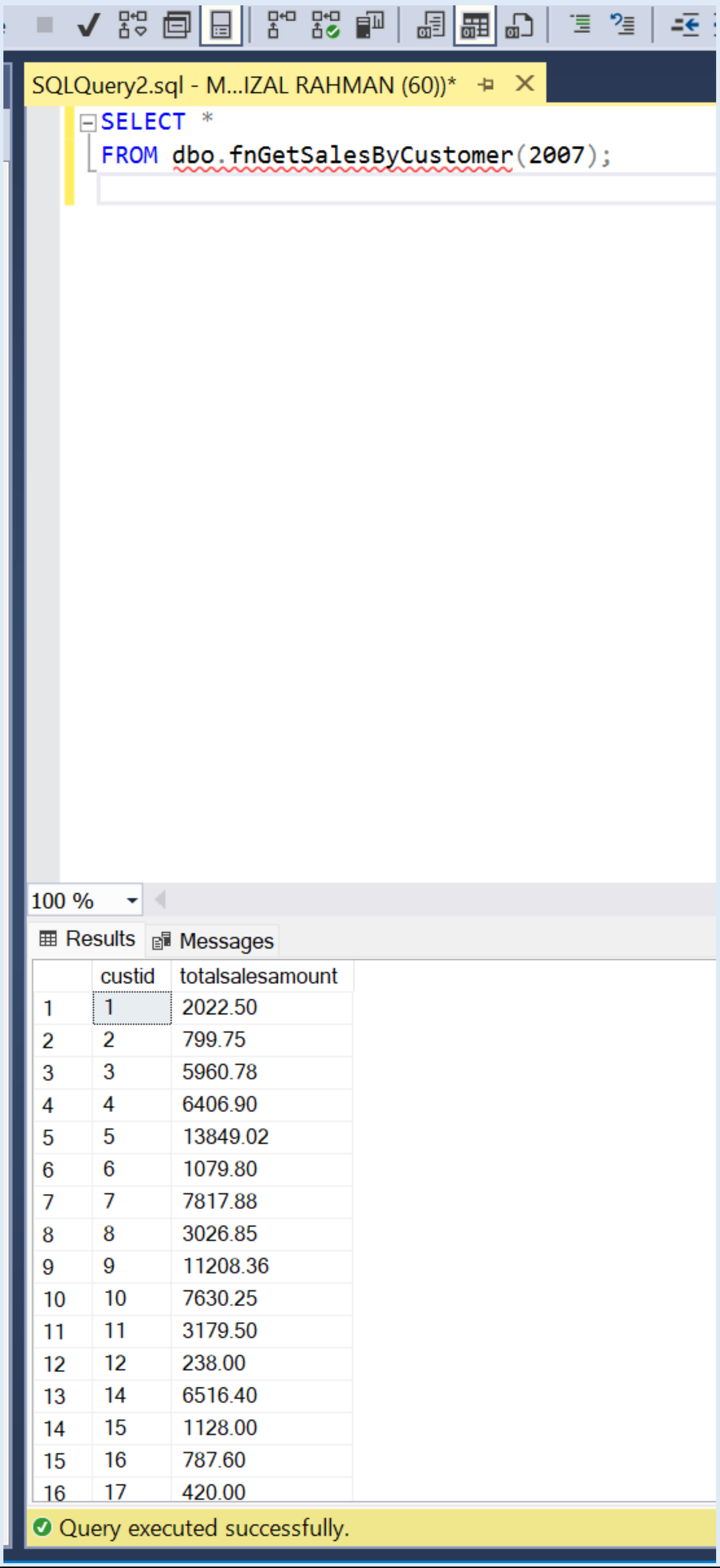
Results | Messages

```
   custid  totalsalesamount
1  1       2022.50
2  2       799.75
3  3       5960.78
4  4       6406.90
5  5       13849.02
6  6       1079.80
7  7       7817.88
8  8       3026.85
9  9       11208.36
```

Query... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (51) | TSQL2012 | 00:00:00 | 86 rows

	 SQLQuery2.sql - M...IZAL RAHMAN (60))*
	`CREATE FUNCTION dbo.fnGetSalesByCustomer (     @orderyear INT ) RETURNS TABLE AS RETURN (     SELECT         custid,         SUM(val) AS totalsalesamount     FROM         Sales.OrderValues     WHERE         YEAR(orderdate) = @orderyear     GROUP BY         custid );`
	100 %
	Messages
	`Commands completed successfully.`
	
	`Completion time: 2024-10-02T10:11:34.5262385+07:00`

	RETURN



SQLQuery2.sql - M...IZAL RAHMAN (60))*

```sql
SELECT *
FROM dbo.fnGetSalesByCustomer(2007);
```

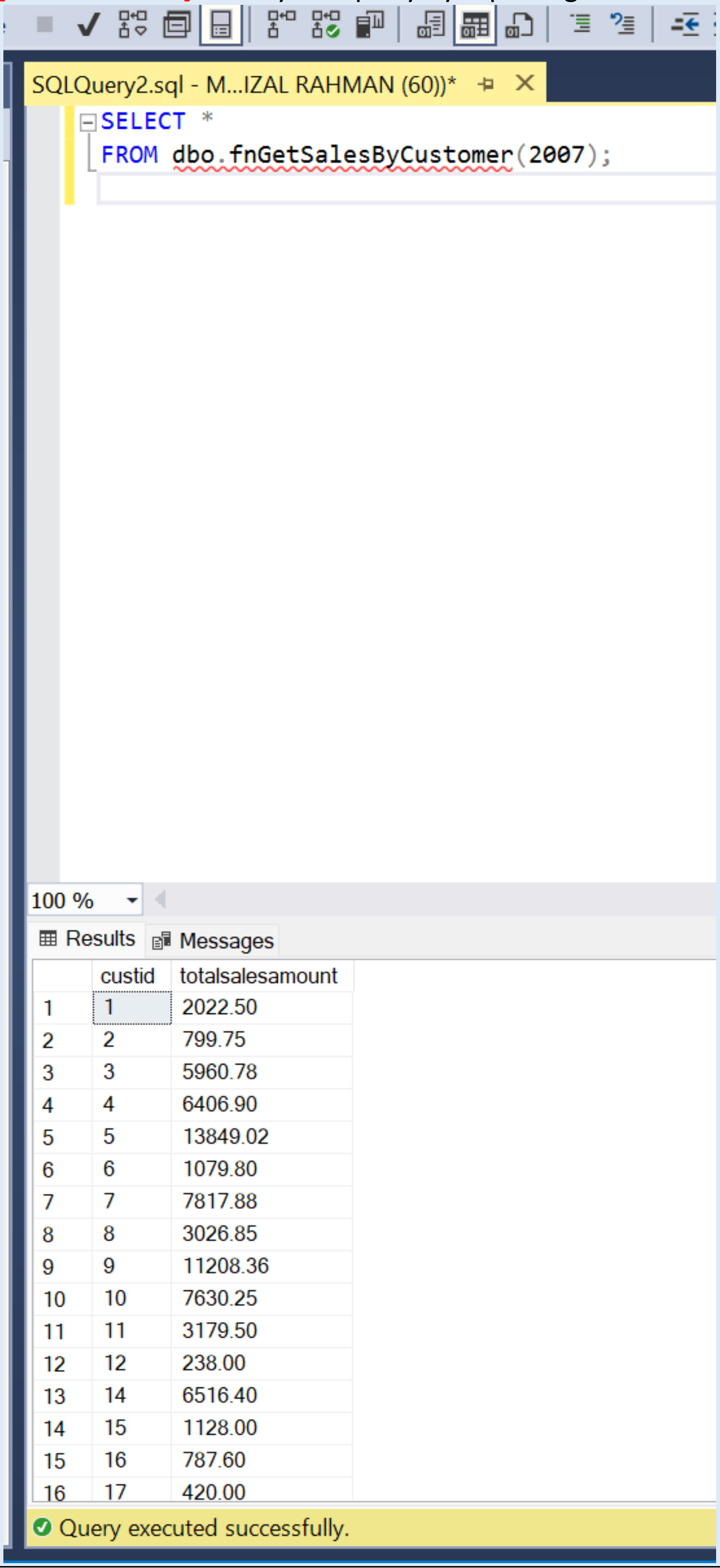
100 %

Results | Messages

	custid	totalsalesamount
1	1	2022.50
2	2	799.75
3	3	5960.78
4	4	6406.90
5	5	13849.02
6	6	1079.80
7	7	7817.88
8	8	3026.85
9	9	11208.36
10	10	7630.25
11	11	3179.50
12	12	238.00
13	14	6516.40
14	15	1128.00
15	16	787.60
16	17	420.00

Query executed successfully.

4	[Question- 16] Modify the query by replacing the constant value of 2007 in the WHERE clause, with the



SQLQuery2.sql - M...IZAL RAHMAN (60))*

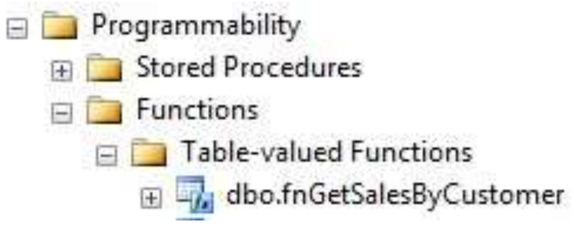
```
SELECT *
FROM dbo.fnGetSalesByCustomer(2007);
```

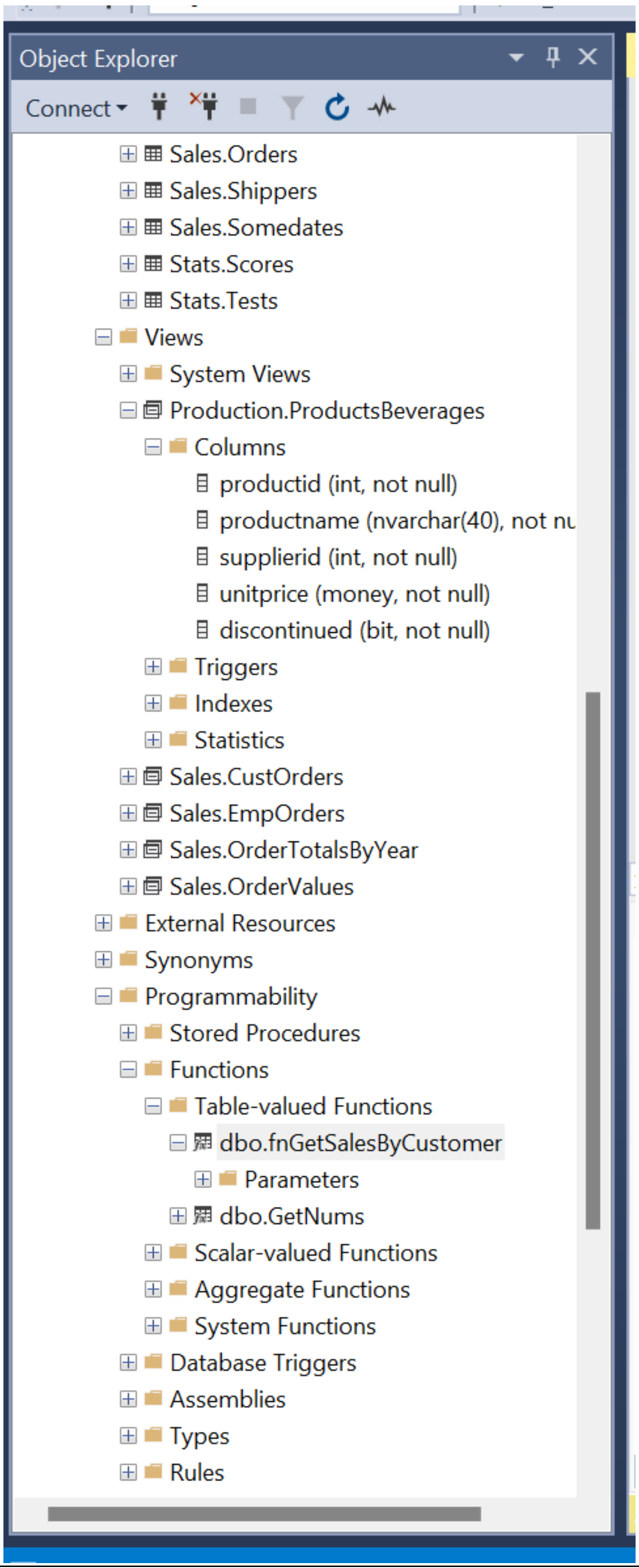
100 %

Results Messages

	custid	totalsalesamount
1	1	2022.50
2	2	799.75
3	3	5960.78
4	4	6406.90
5	5	13849.02
6	6	1079.80
7	7	7817.88
8	8	3026.85
9	9	11208.36
10	10	7630.25
11	11	3179.50
12	12	238.00
13	14	6516.40
14	15	1128.00
15	16	787.60
16	17	420.00

Query executed successfully.

5	Run the script in step 4 above so that an inline TVF named **dbo.fnGetSalesByCustomer will be created.**  Programmability / Stored Procedures / Functions / Table-valued Functions / dbo.fnGetSalesByCustomer

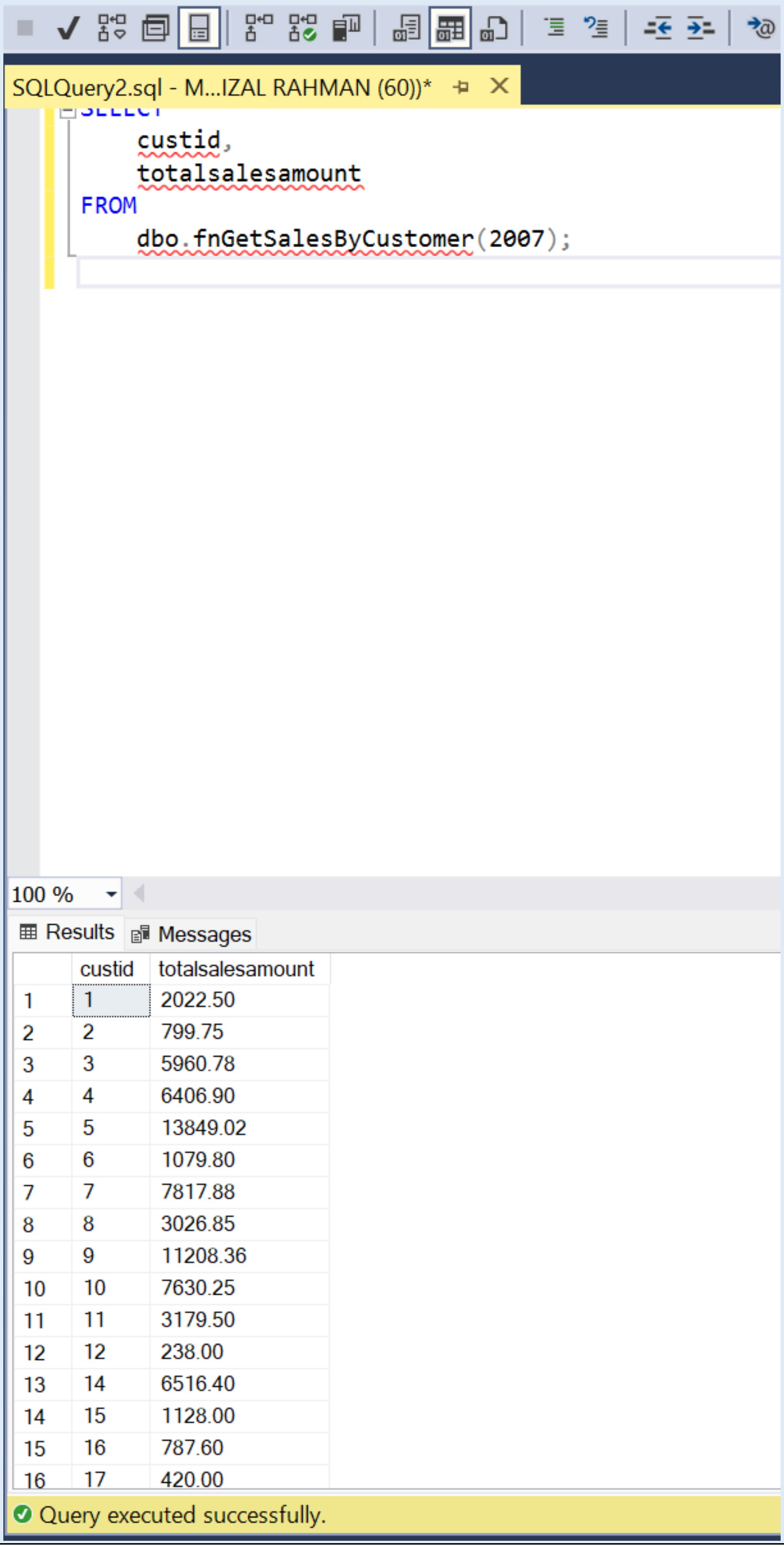
Object Explorer

Connect

```
        ⊞ Sales.Orders
        ⊞ Sales.Shippers
        ⊞ Sales.Somedates
        ⊞ Stats.Scores
        ⊞ Stats.Tests
      ⊟ Views
        ⊞ System Views
        ⊟ Production.ProductsBeverages
          ⊟ Columns
              productid (int, not null)
              productname (nvarchar(40), not nu
              supplierid (int, not null)
              unitprice (money, not null)
              discontinued (bit, not null)
          ⊞ Triggers
          ⊞ Indexes
          ⊞ Statistics
        ⊞ Sales.CustOrders
        ⊞ Sales.EmpOrders
        ⊞ Sales.OrderTotalsByYear
        ⊞ Sales.OrderValues
      ⊞ External Resources
      ⊞ Synonyms
      ⊟ Programmability
        ⊞ Stored Procedures
        ⊟ Functions
          ⊟ Table-valued Functions
            ⊟ dbo.fnGetSalesByCustomer
              ⊞ Parameters
            ⊞ dbo.GetNums
          ⊞ Scalar-valued Functions
          ⊞ Aggregate Functions
          ⊞ System Functions
        ⊞ Database Triggers
        ⊞ Assemblies
        ⊞ Types
        ⊞ Rules
```

Practical – Part 12 : Inline ITF - Creating a SELECT query that operates on an inline table-valued function

Step	Information
1	[Question- 17] Create a SELECT query containing the *custid* and *totalsalesamount columns* against the inline TVF **dbo.fnGetSalesByCustomer** . Enter the value 2007 as the parameter.


SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
SELECT
    custid,
    totalsalesamount
FROM
    dbo.fnGetSalesByCustomer(2007);
```

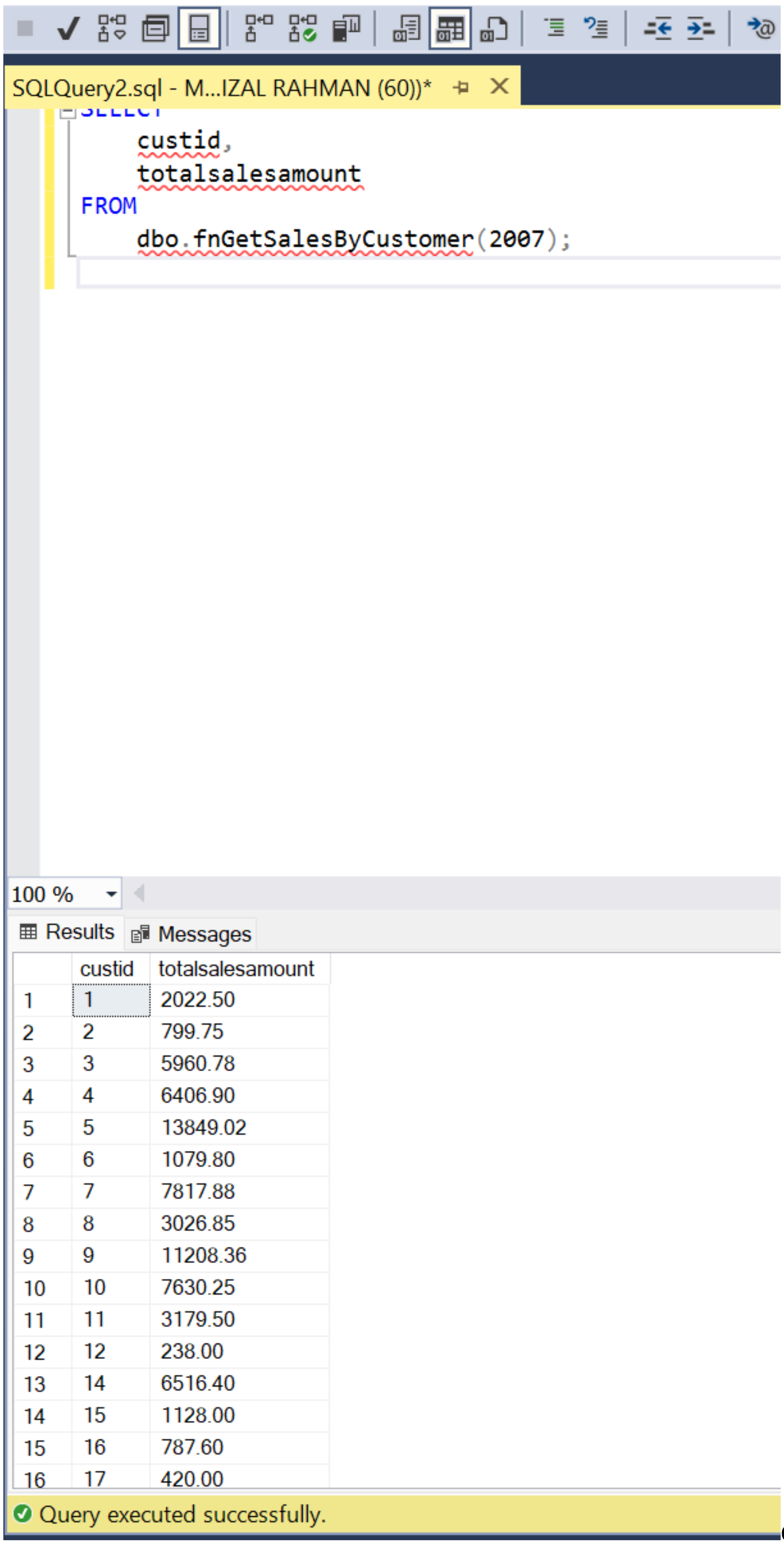
100 %

Results | Messages

	custid	totalsalesamount
1	1	2022.50
2	2	799.75
3	3	5960.78
4	4	6406.90
5	5	13849.02
6	6	1079.80
7	7	7817.88
8	8	3026.85
9	9	11208.36
10	10	7630.25
11	11	3179.50
12	12	238.00
13	14	6516.40
14	15	1128.00
15	16	787.60
16	17	420.00

Query executed successfully.

2	Execute the query in step 1 above and compare it with the results shown in the following



SQLQuery2.sql - M...IZAL RAHMAN (60))*

```
    custid,
    totalsalesamount
FROM
    dbo.fnGetSalesByCustomer(2007);
```

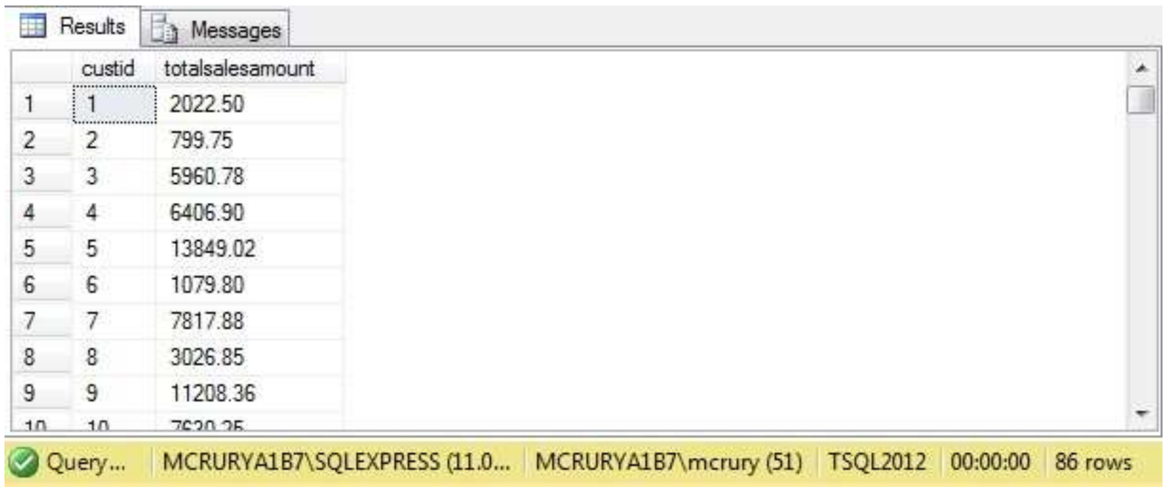
100 %

Results | Messages

	custid	totalsalesamount
1	1	2022.50
2	2	799.75
3	3	5960.78
4	4	6406.90
5	5	13849.02
6	6	1079.80
7	7	7817.88
8	8	3026.85
9	9	11208.36
10	10	7630.25
11	11	3179.50
12	12	238.00
13	14	6516.40
14	15	1128.00
15	16	787.60
16	17	420.00

Query executed successfully.

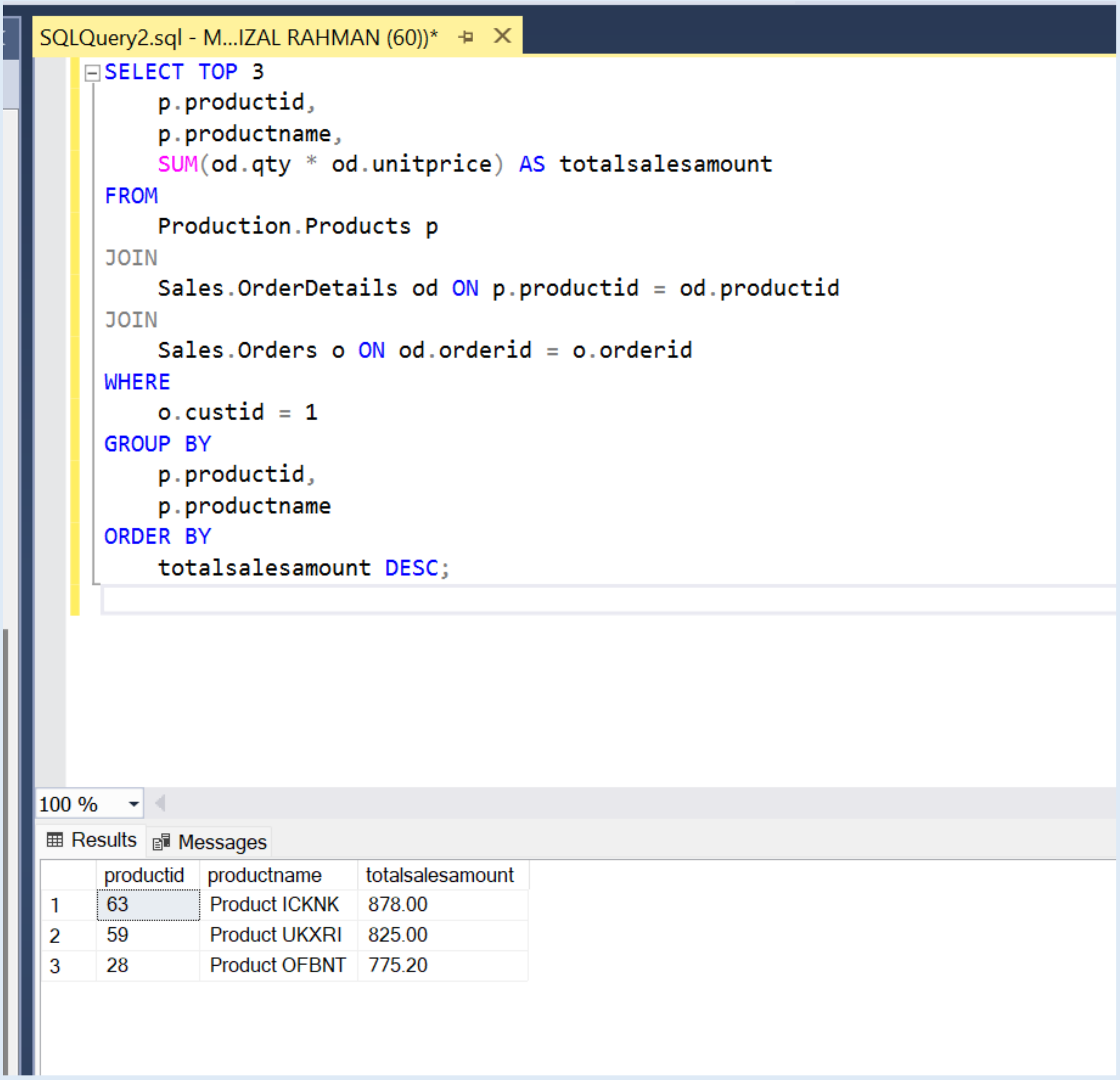
display:

```
Results   Messages
     custid  totalsalesamount
1    1       2022.50
2    2       799.75
3    3       5960.78
4    4       6406.90
5    5       13849.02
6    6       1079.80
7    7       7817.88
8    8       3026.85
9    9       11208.36
10   10      7630.25
Query...  MCRURYA1B7\SQLEXPRESS (11.0...  MCRURYA1B7\mcrury (51)  TSQL2012  00:00:00  86 rows
```

Practical – Part 13 : Inline ITF - Creating a SELECT query to get the 3 best-selling products for a particular customer

Step	**Information**
1	[Question-1 8] Create a SELECT query that displays the top 3 best-selling products for a customer with ID = 1. Get the *productid* and *productname columns* from the **Production.Products table** . Use the *qty* and *unitprice columns from the Sales.OrderDetails* table to calculate the nominal value for each order row, which is then added up for each product to produce the *totalsalesamount column* . Filter the results to only display data with a custid value = 1.

SQLQuery2.sql - M...IZAL RAHMAN (60))*

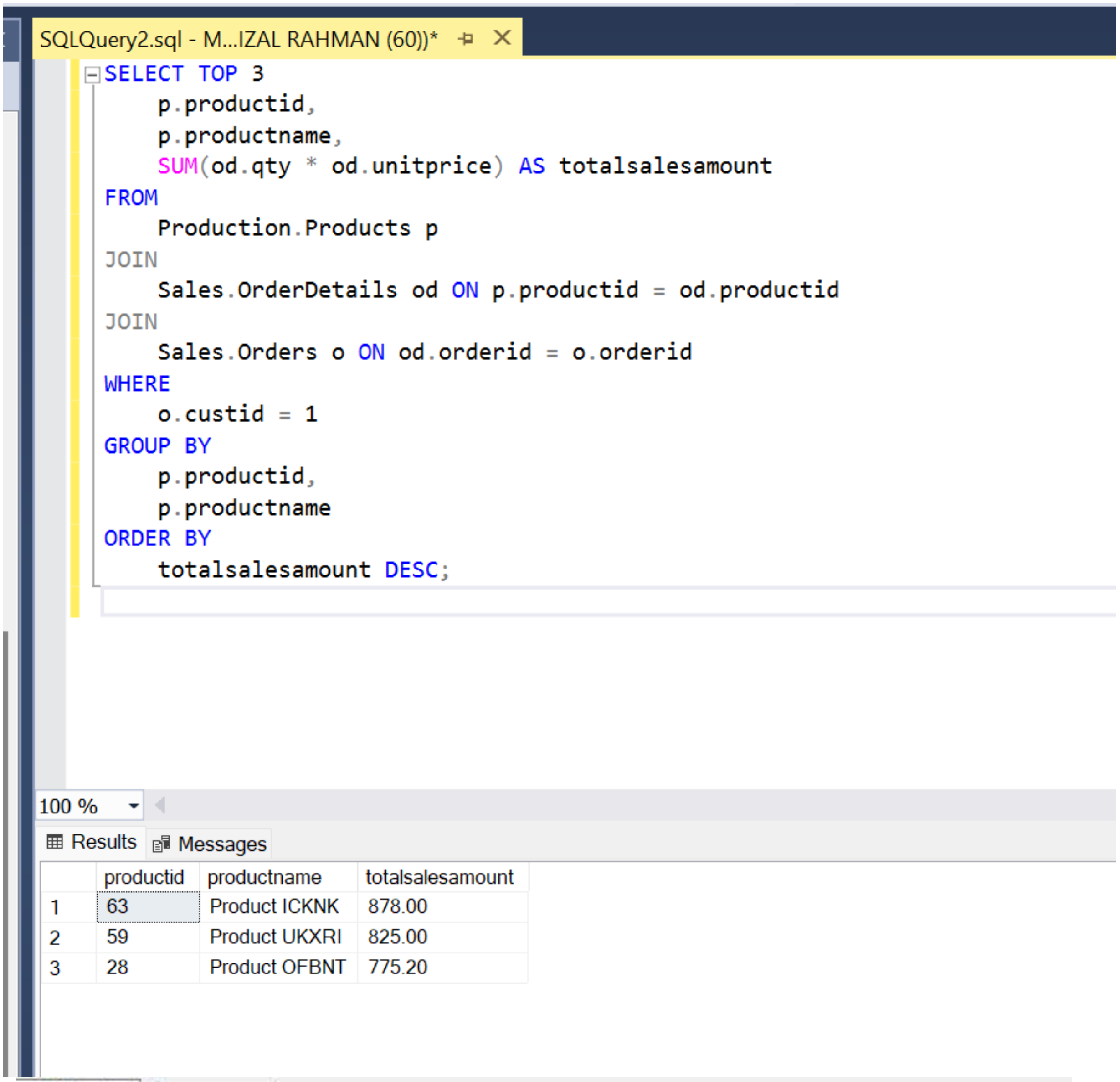
```
SELECT TOP 3
     p.productid,
     p.productname,
     SUM(od.qty * od.unitprice) AS totalsalesamount
FROM
     Production.Products p
JOIN
     Sales.OrderDetails od ON p.productid = od.productid
JOIN
     Sales.Orders o ON od.orderid = o.orderid
WHERE
     o.custid = 1
GROUP BY
     p.productid,
     p.productname
ORDER BY
     totalsalesamount DESC;
```

100 %

Results | Messages

	productid	productname	totalsalesamount
1	63	Product ICKNK	878.00
2	59	Product UKXRI	825.00
3	28	Product OFBNT	775.20

2	Execute the query in step 1 above and compare it with the results shown in the following display:



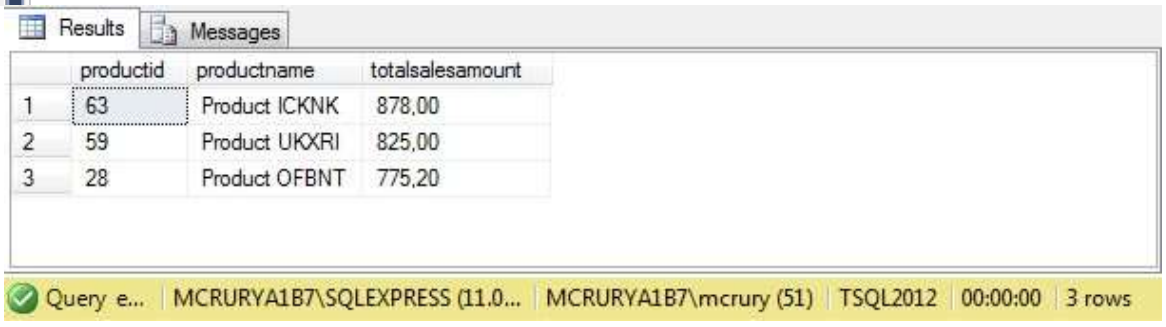
SQLQuery2.sql - M...IZAL RAHMAN (60))*

```sql
SELECT TOP 3
     p.productid,
     p.productname,
     SUM(od.qty * od.unitprice) AS totalsalesamount
FROM
     Production.Products p
JOIN
     Sales.OrderDetails od ON p.productid = od.productid
JOIN
     Sales.Orders o ON od.orderid = o.orderid
WHERE
     o.custid = 1
GROUP BY
     p.productid,
     p.productname
ORDER BY
     totalsalesamount DESC;
```

100 %

Results | Messages

```
   productid  productname     totalsalesamount
1  63         Product ICKNK   878.00
2  59         Product UKXRI   825.00
3  28         Product OFBNT   775.20
```



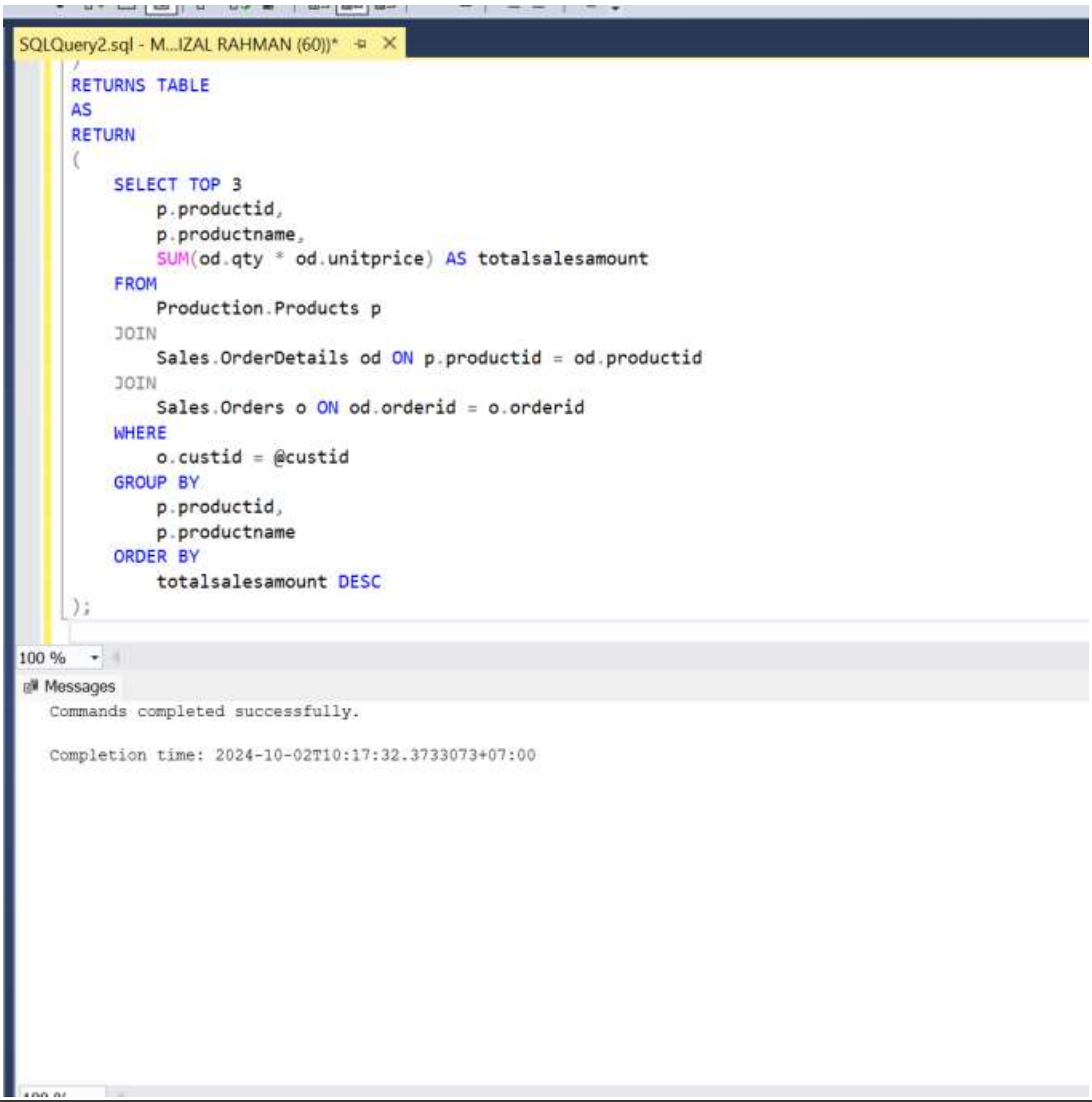
Results | Messages

```
   productid  productname     totalsalesamount
1  63         Product ICKNK   878,00
2  59         Product UKXRI   825,00
3  28         Product OFBNT   775,20
```

Query e... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (51) | TSQL2012 | 00:00:00 | 3 rows

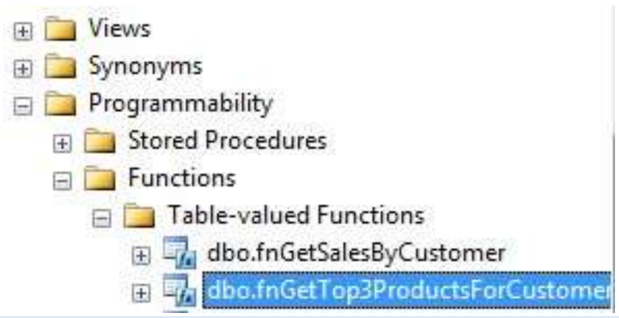
3	[Question-1 9] Using the SELECT query in step 1 above, create an inline TVF by adding a few lines of function before the SELECT query and set the value of *the custid constant* in the query with the **@custid parameter** , as follows:
	
	`CREATE FUNCTION dbo . fnGetTop3ProductsForCustomer`
	`( @custid US INT ) RETURNS TABLE`
	`AS`
	`RETURN`

4	Run the script so that an inline TVF named **dbo.fnGetTop3ProductsForCustomer will be created** which has a customer ID parameter.

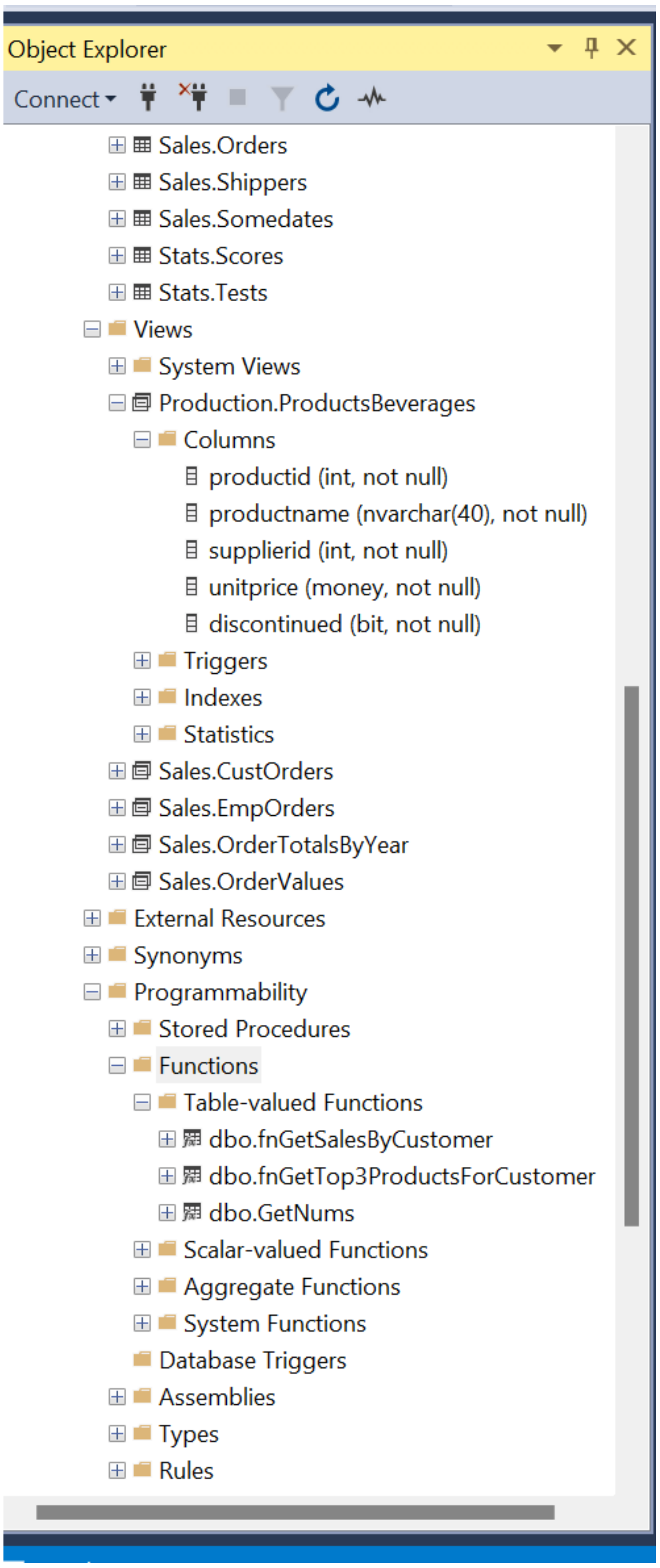
	

```
)
RETURNS TABLE
AS
RETURN
(
    SELECT TOP 3
        p.productid,
        p.productname,
        SUM(od.qty * od.unitprice) AS totalsalesamount
    FROM
        Production.Products p
    JOIN
        Sales.OrderDetails od ON p.productid = od.productid
    JOIN
        Sales.Orders o ON od.orderid = o.orderid
    WHERE
        o.custid = @custid
    GROUP BY
        p.productid,
        p.productname
    ORDER BY
        totalsalesamount DESC
);
```

```
Commands completed successfully.

Completion time: 2024-10-02T10:17:32.3733073+07:00
```

Views
Synonyms
Programmability
Stored Procedures
Functions
Table-valued Functions
dbo.fnGetSalesByCustomer
dbo.fnGetTop3ProductsForCustomer

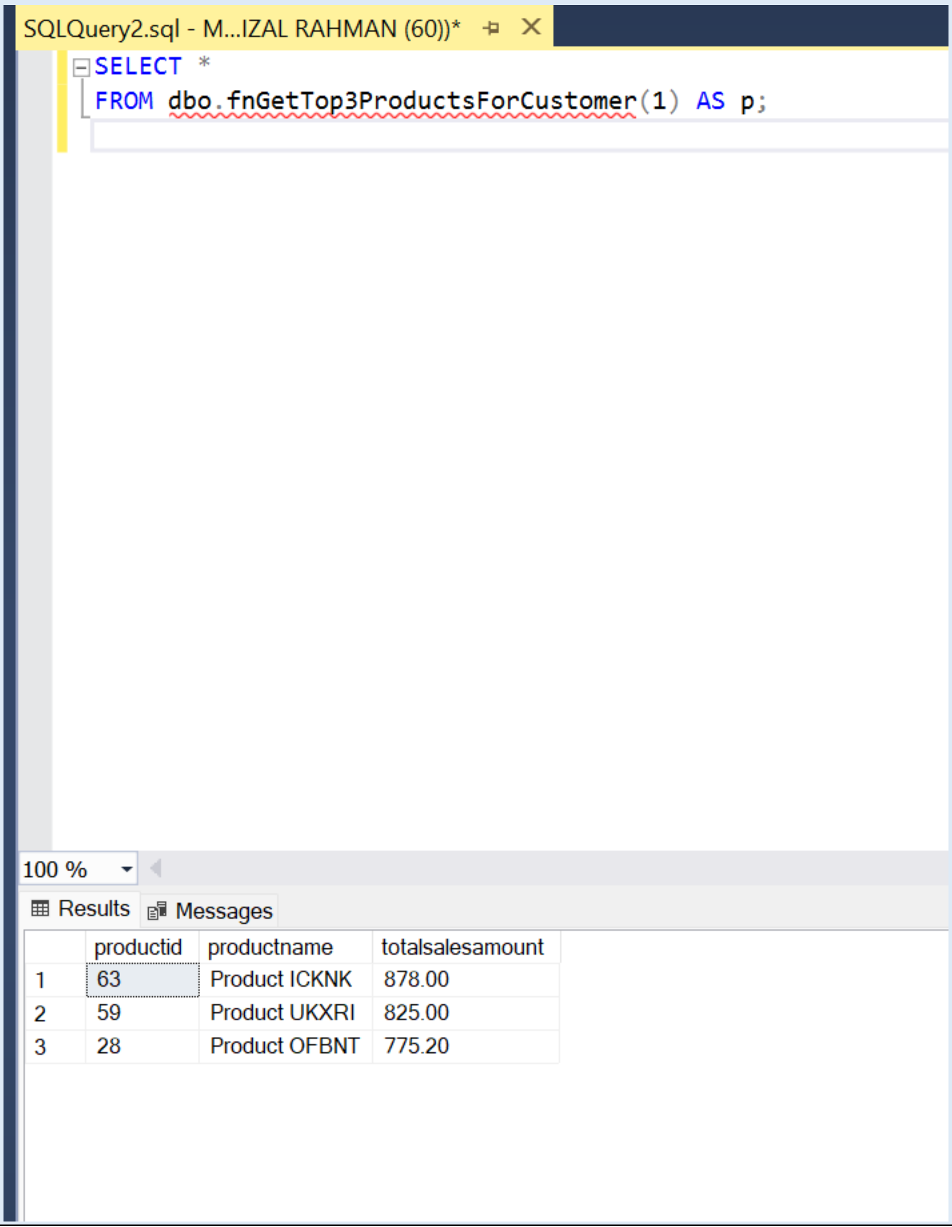


Object Explorer

Connect

```
      ⊞ Sales.Orders
      ⊞ Sales.Shippers
      ⊞ Sales.Somedates
      ⊞ Stats.Scores
      ⊞ Stats.Tests
    ⊟ Views
      ⊞ System Views
      ⊟ Production.ProductsBeverages
        ⊟ Columns
            productid (int, not null)
            productname (nvarchar(40), not null)
            supplierid (int, not null)
            unitprice (money, not null)
            discontinued (bit, not null)
        ⊞ Triggers
        ⊞ Indexes
        ⊞ Statistics
      ⊞ Sales.CustOrders
      ⊞ Sales.EmpOrders
      ⊞ Sales.OrderTotalsByYear
      ⊞ Sales.OrderValues
    ⊞ External Resources
    ⊞ Synonyms
    ⊟ Programmability
      ⊞ Stored Procedures
      ⊟ Functions
        ⊟ Table-valued Functions
          ⊞ dbo.fnGetSalesByCustomer
          ⊞ dbo.fnGetTop3ProductsForCustomer
          ⊞ dbo.GetNums
        ⊞ Scalar-valued Functions
        ⊞ Aggregate Functions
        ⊞ System Functions
        Database Triggers
      ⊞ Assemblies
      ⊞ Types
      ⊞ Rules
```

5	[Question-20] Test it by creating a SELECT query on the inline TVF and insert the value 1 as the customer ID parameter. Display the *productid* , *productname* , *totalsalesamount columns* , and give the alias name *p* for the inline TVF.



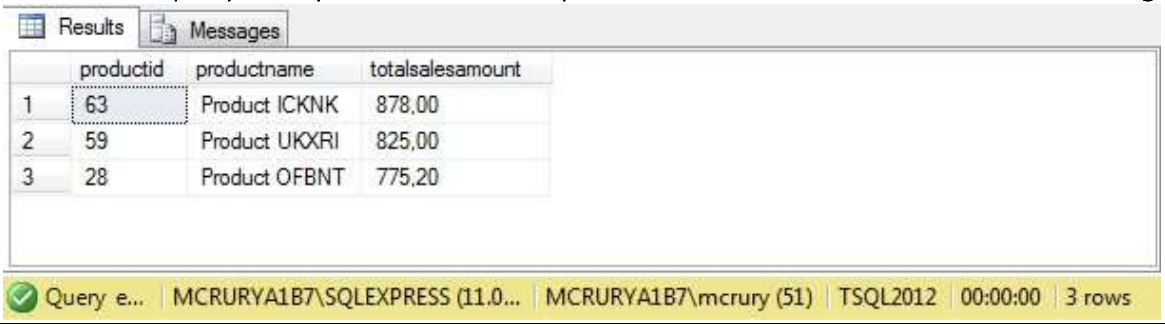
SQLQuery2.sql - M...IZAL RAHMAN (60))*

```sql
SELECT *
FROM dbo.fnGetTop3ProductsForCustomer(1) AS p;
```

100 %

Results | Messages

	productid	productname	totalsalesamount
1	63	Product ICKNK	878.00
2	59	Product UKXRI	825.00
3	28	Product OFBNT	775.20

6	Execute the query in step 1 above and compare it with the results shown in the following display:  Results Messages

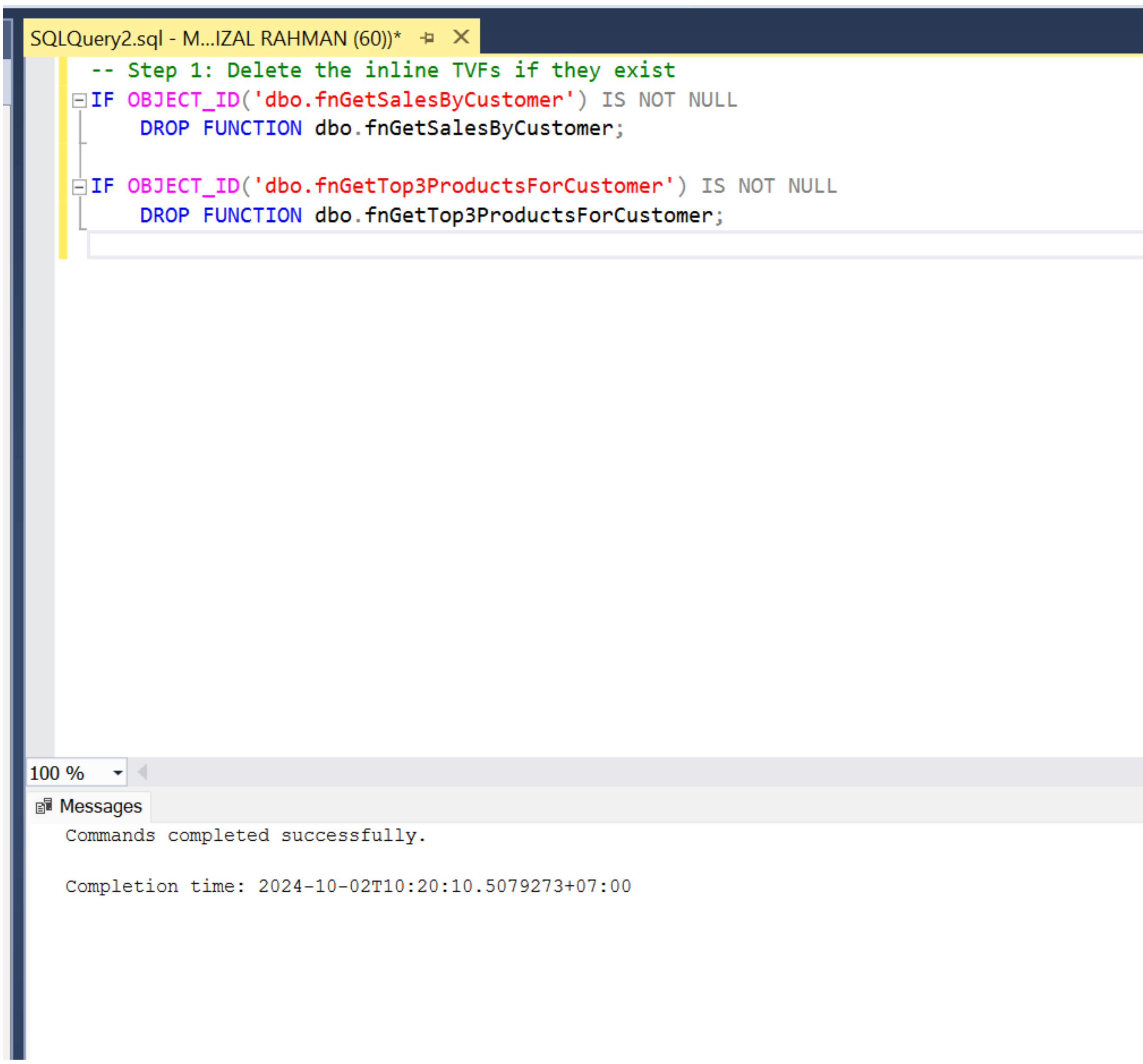
```
   productid  productname     totalsalesamount
1  63         Product ICKNK   878,00
2  59         Product UKXRI   825,00
3  28         Product OFBNT   775,20
```

Query e... | MCRURYA1B7\SQLEXPRESS (11.0... | MCRURYA1B7\mcrury (51) | TSQL2012 | 00:00:00 | 3 rows

<u>Lab – Part 14 : Inline TVF - Deleting inline Table-valued function</u>

Step	**Information**
1	Delete the inline TVF that has been created by running the following script:

```
IF  OBJECT_ID  (  'dbo.fnGetSalesByCustomer'  )  IS  NOT  NULL  DROP  FUNCTION  dbo
fnGetSalesByCustomer ;
IF OBJECT_ID ( 'dbo.fnGetTop3ProductsForCustomer' ) IS NOT NULL DROP FUNCTION dbo .
fnGetTop3ProductsForCustomer ;
```

```sql
-- Step 1: Delete the inline TVFs if they exist
IF OBJECT_ID('dbo.fnGetSalesByCustomer') IS NOT NULL
    DROP FUNCTION dbo.fnGetSalesByCustomer;

IF OBJECT_ID('dbo.fnGetTop3ProductsForCustomer') IS NOT NULL
    DROP FUNCTION dbo.fnGetTop3ProductsForCustomer;
```

```
Commands completed successfully.

Completion time: 2024-10-02T10:20:10.5079273+07:00
```

--- Have a great time doing it ----