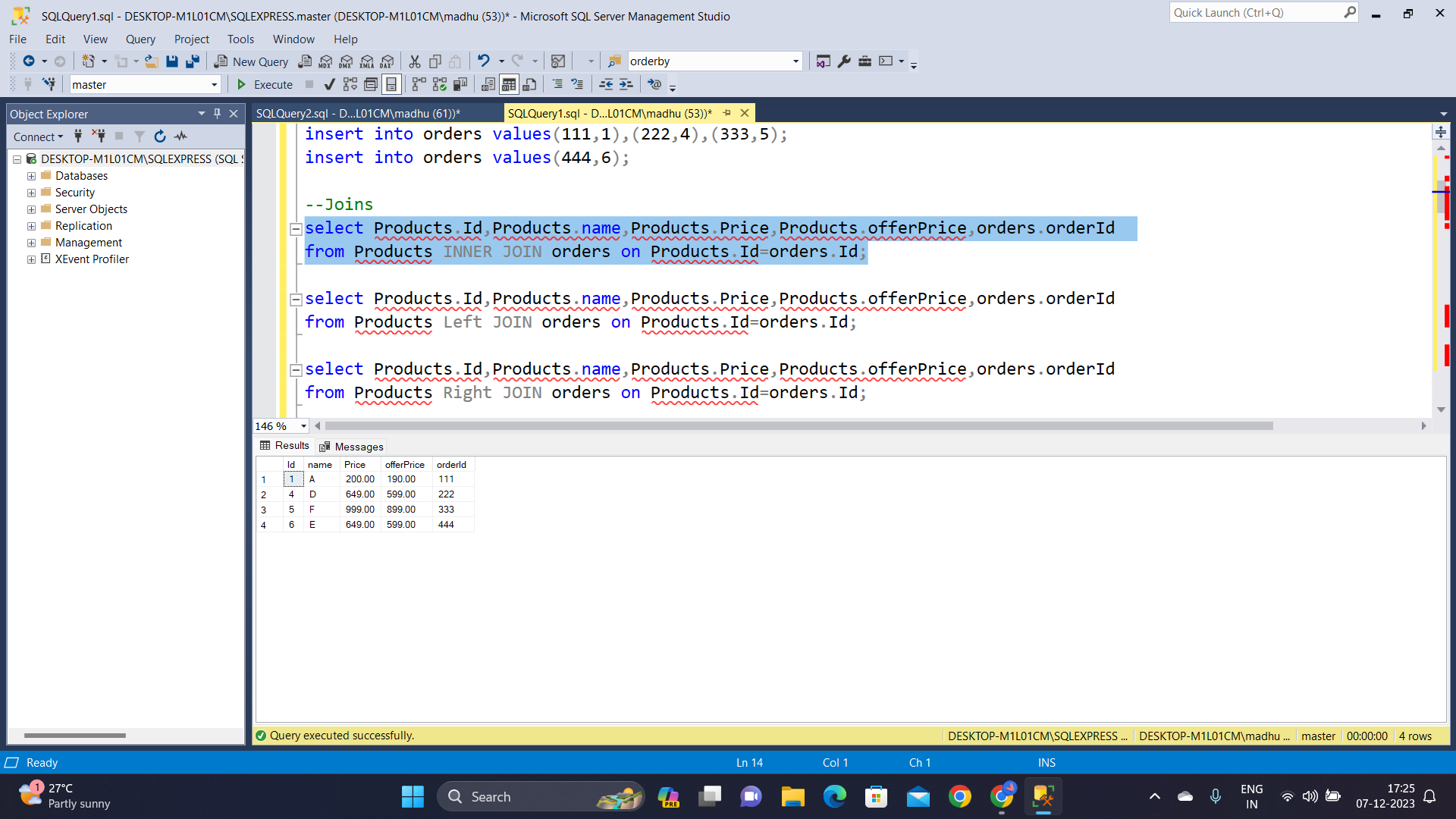
**Day-4**

**Madhu Kalyani Gadi (07-12-2023)**

**Joins-**

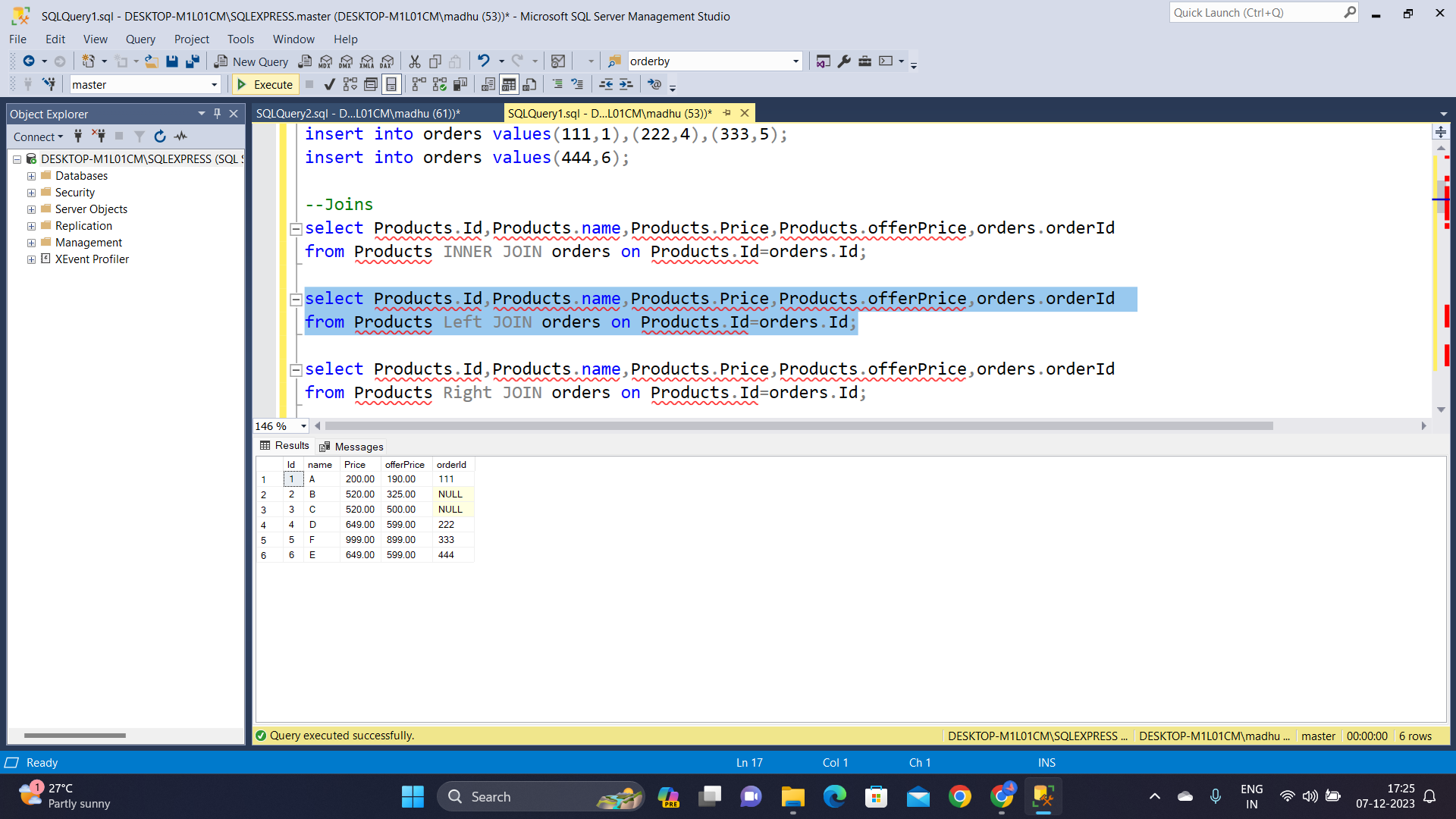
**Inner join**

Here, It joins the "Products" and "orders" tables using an **INNER JOIN** on the "Id" column. It selects the product ID, name, price, offer price, and the associated order ID from the combined dataset.



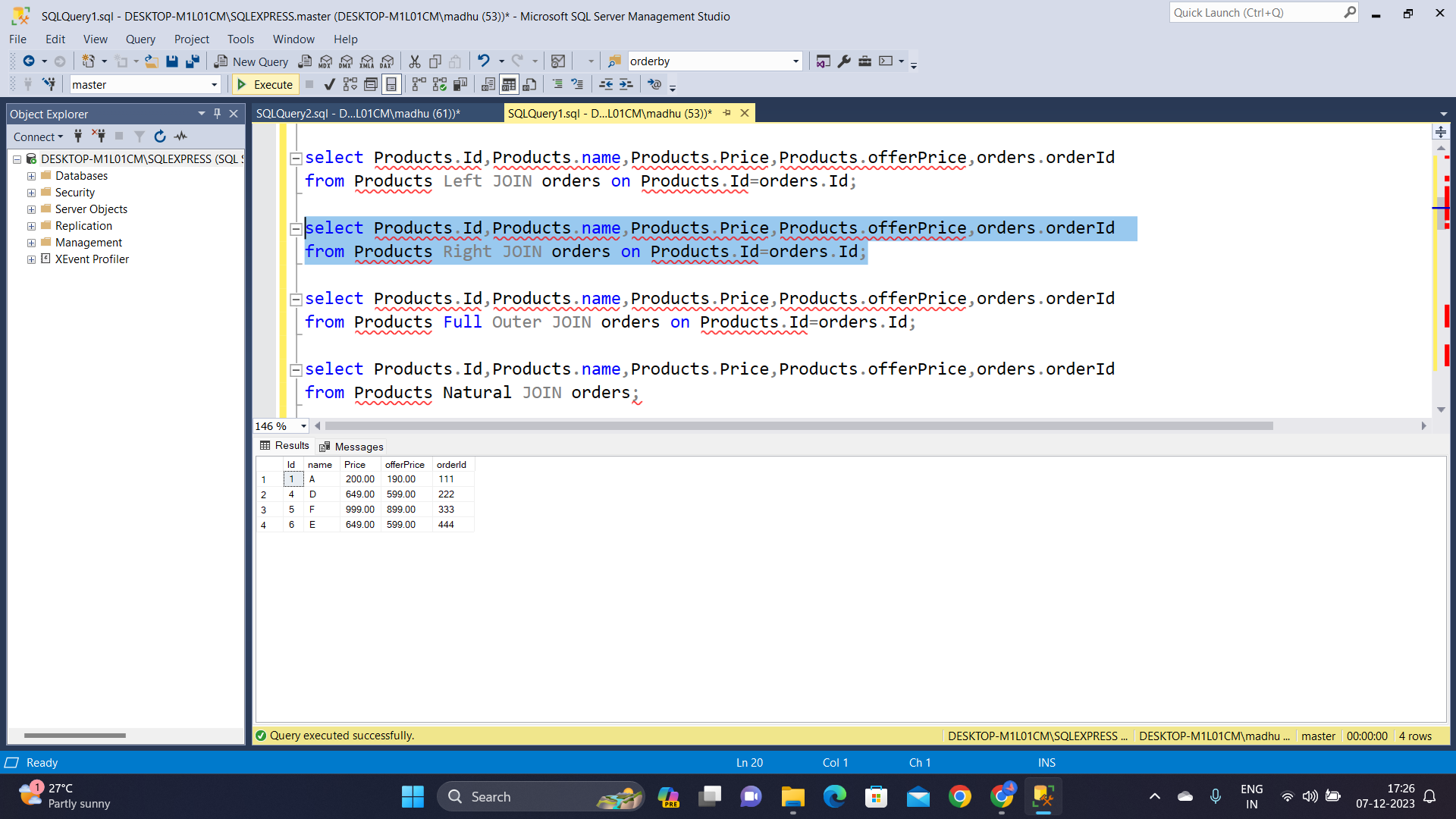
**LEFT JOIN**

This query searches product details and associated order IDs, if any, using a **LEFT JOIN** between "Products" and "orders" based on matching product IDs.



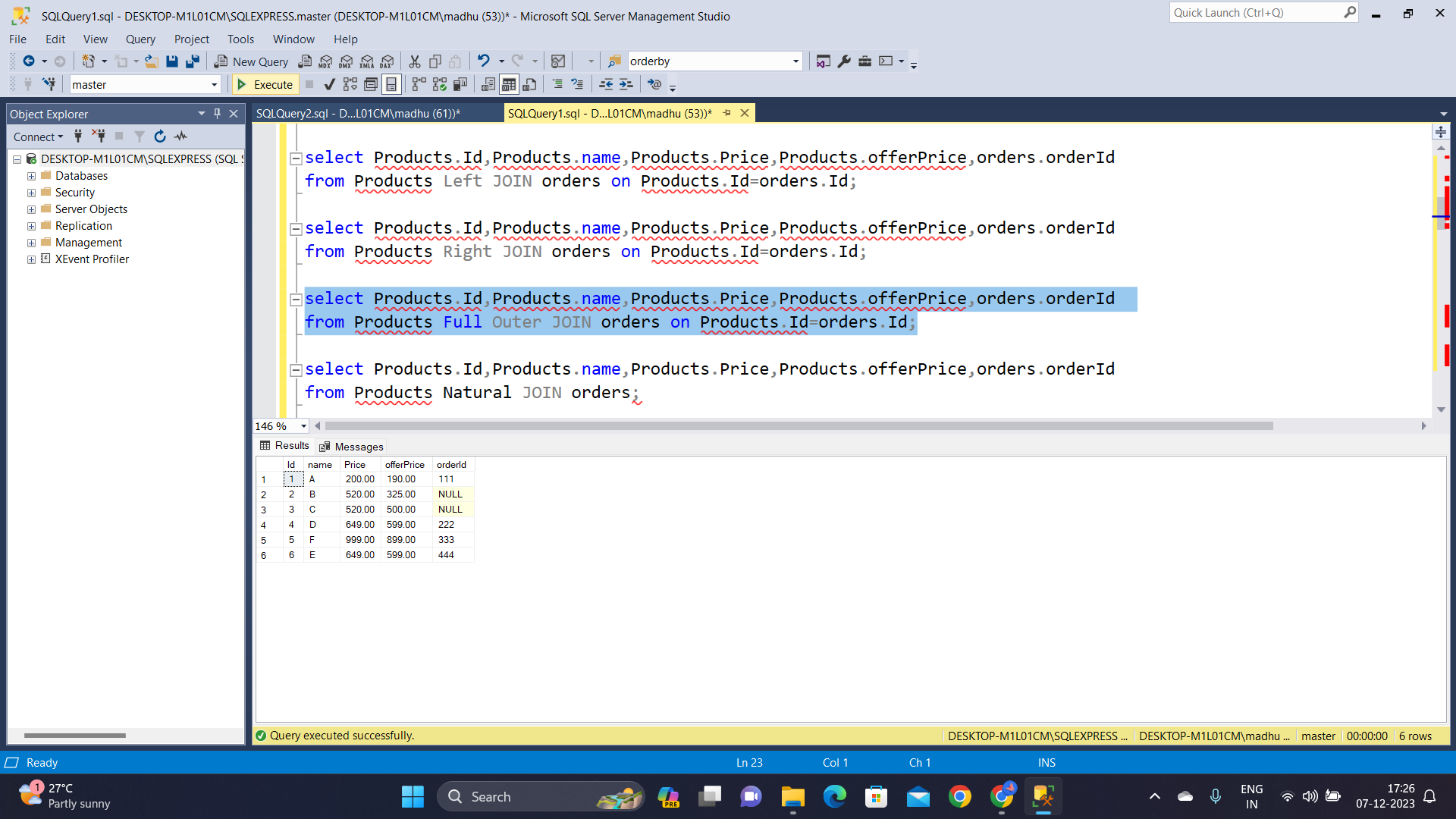
**RIGHT JOIN**

The below retrieves product details and associated order IDs, using a **RIGHT JOIN** between "Products" and "orders" based on matching product IDs. It ensures that all orders, regardless of having associated products, are included in the result.



**FULL JOIN**

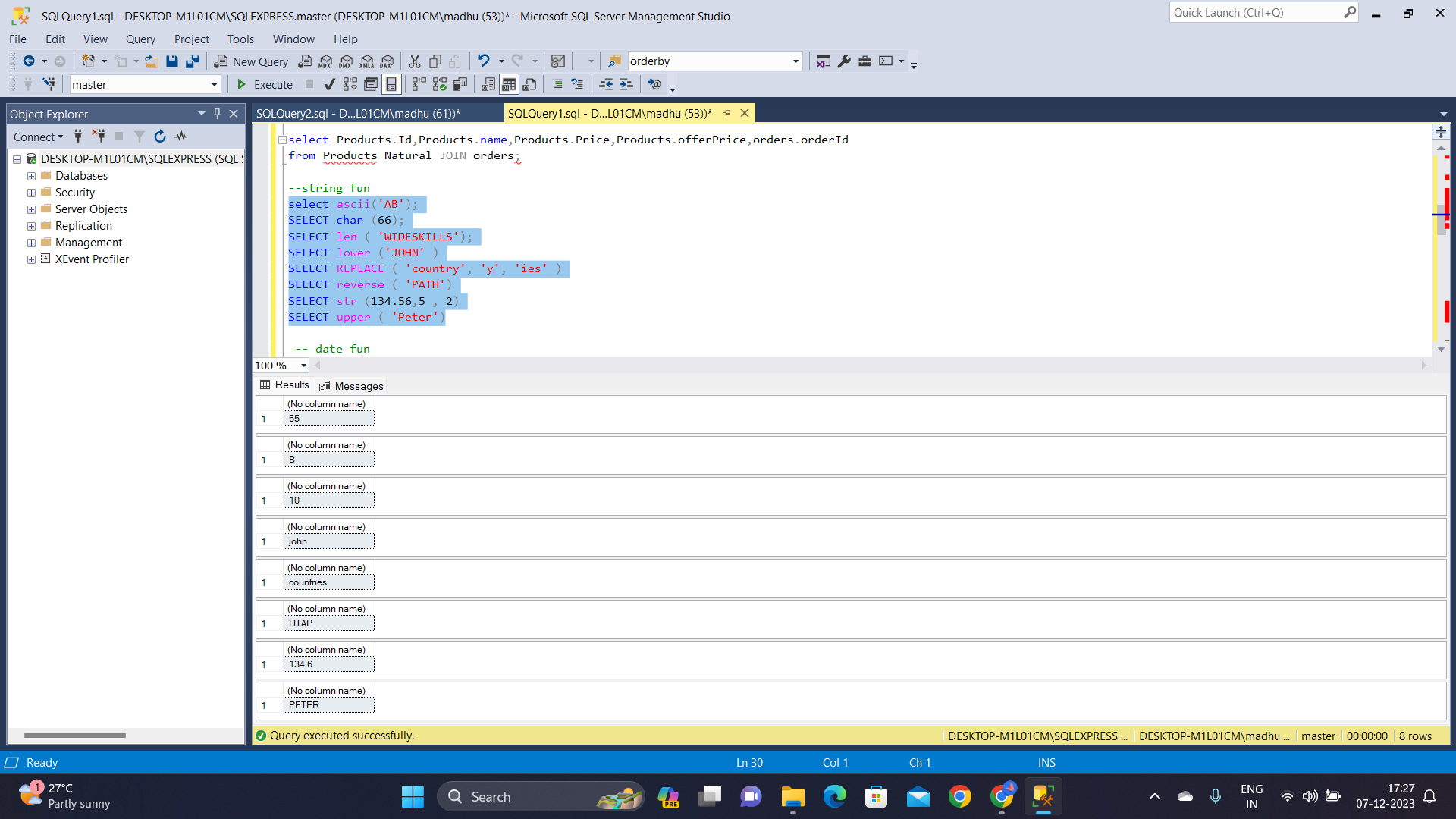
The below query combines product details and order IDs using a **FULL JOIN** between "Products" and "orders," ensuring that all products and orders, regardless of matching IDs, are included in the result.



**SQL Functions**

**String functions**

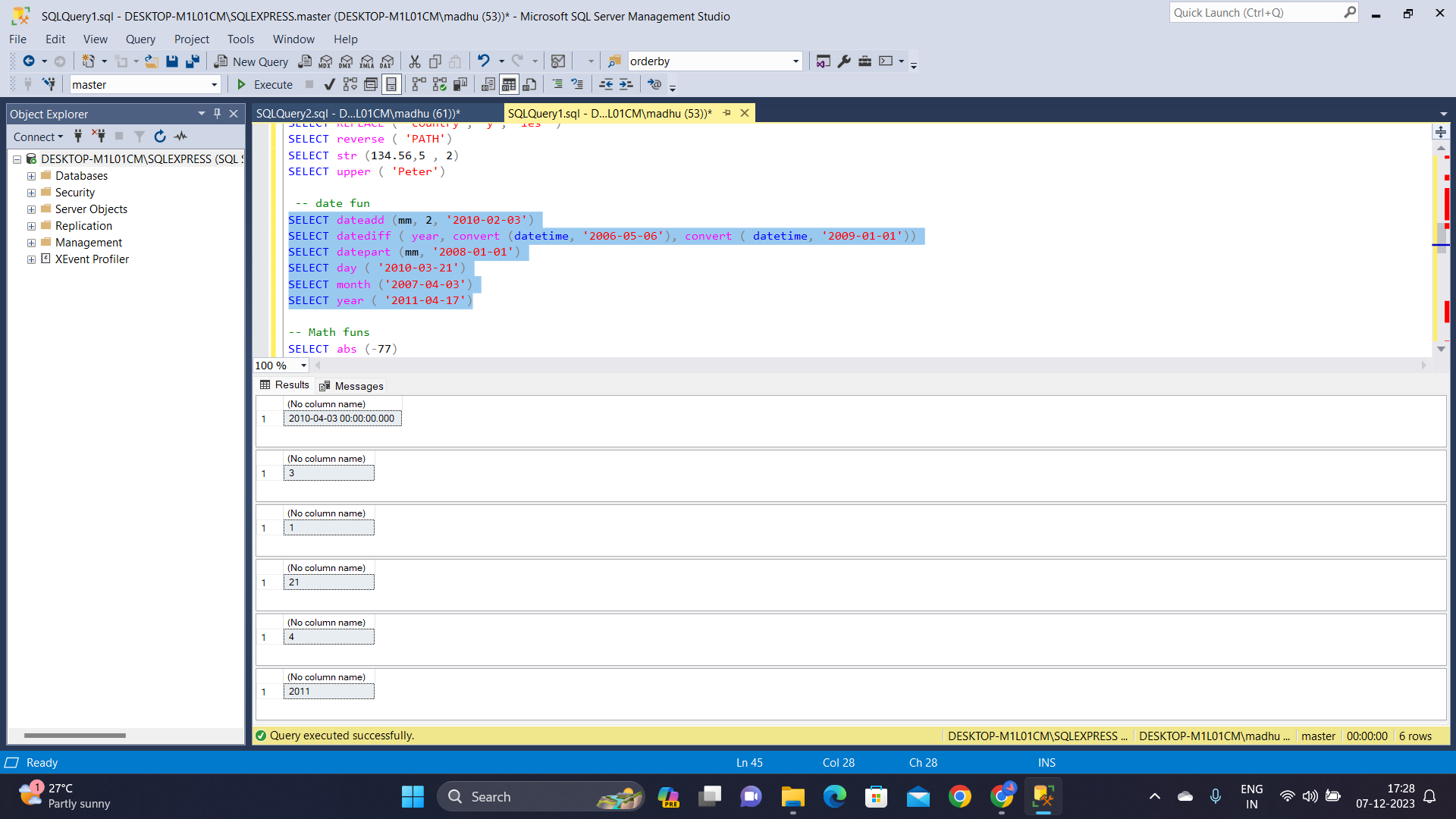
The first query returns the **ASCII** value of the string 'AB', while the second query uses the **CHAR** function to obtain the character corresponding to the ASCII value 66. The third query calculates the length of the string 'WIDESKILLS' using the **LEN** function. Later queries involve converting case, replacing substrings, reversing the string, formatting a numeric value using the **STR** function, and finding the **reverse** of the string 'PATH'.



**Date functions**

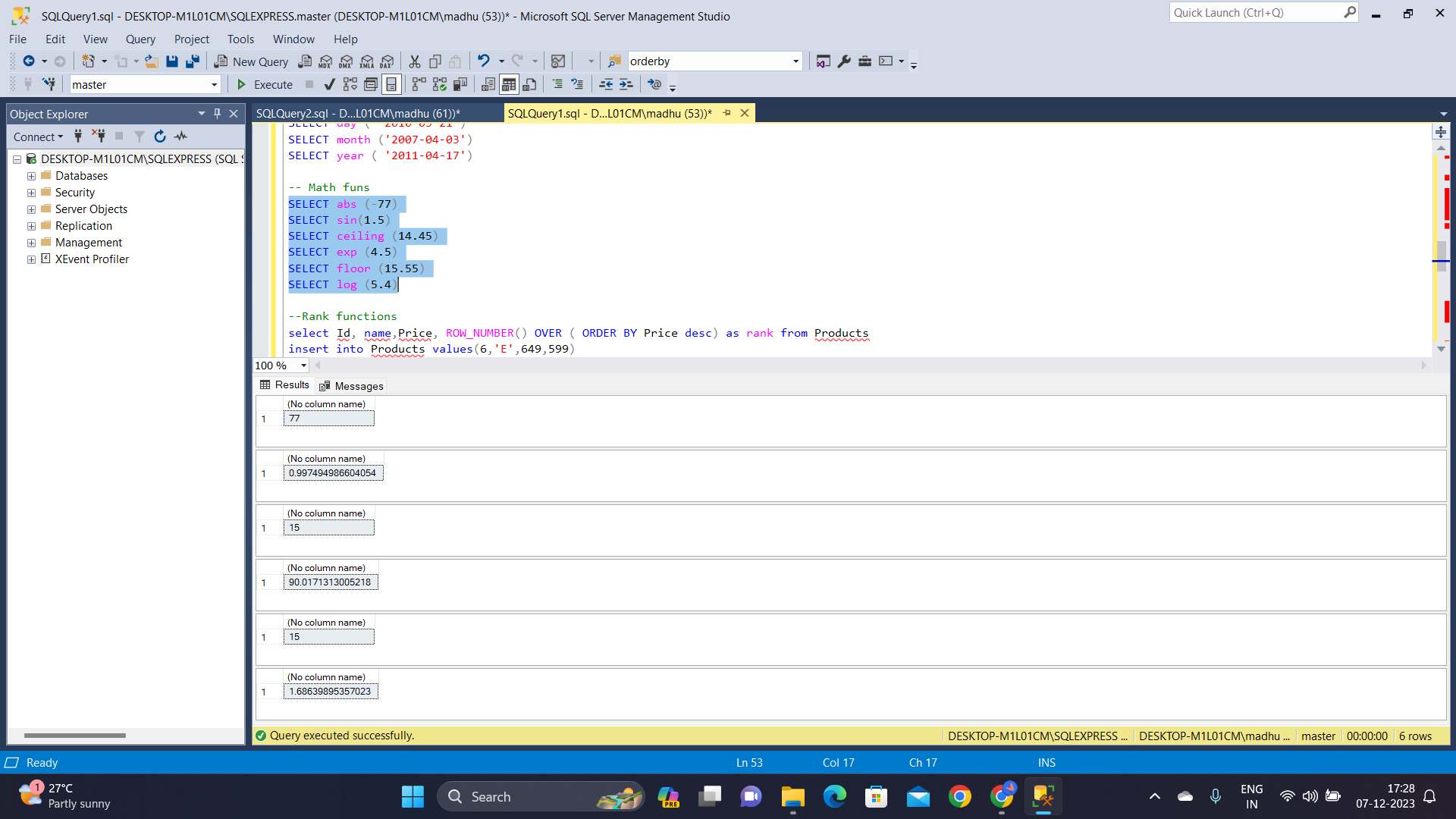
Date functions are performed below.

The **dateadd** adds 2 months to the date '2010-02-03', resulting in '2010-04-03'. The **datediff** calculates the difference in years between '2006-05-06' and '2009-01-01', returning the result 2. Additionally, the queries extract the **month**, **day**, and **year** from specified date values.



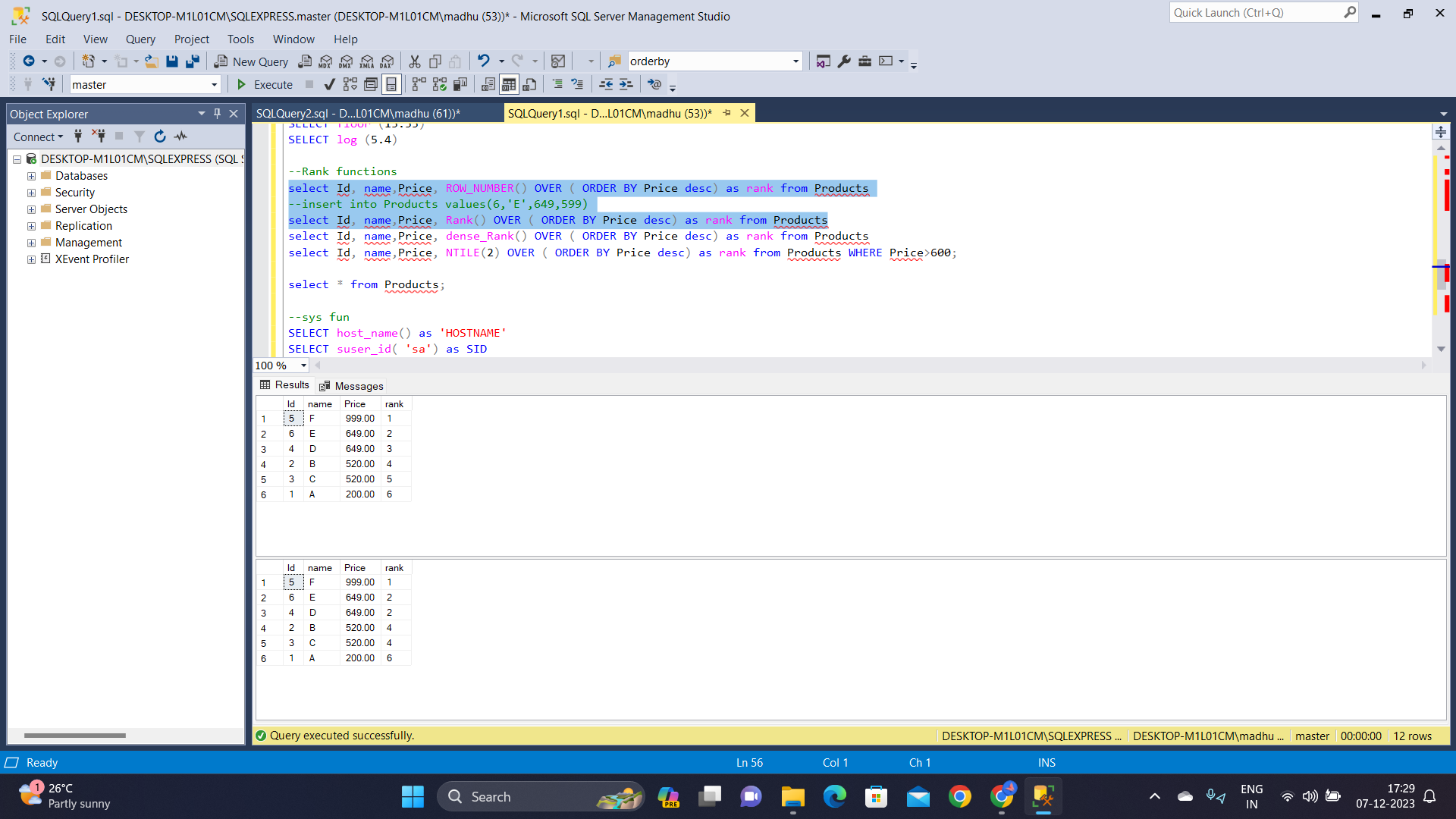
**Math Functions**

The **abs** calculates the absolute value of -77, the **sin** computes the sine of 1.5, the **ceiling** rounds up 14.45 to 15, the **exp** computes the exponential value of 4.5, the **floor** rounds down 15.55 to 15, and the **log** finds the natural logarithm of 5.4.

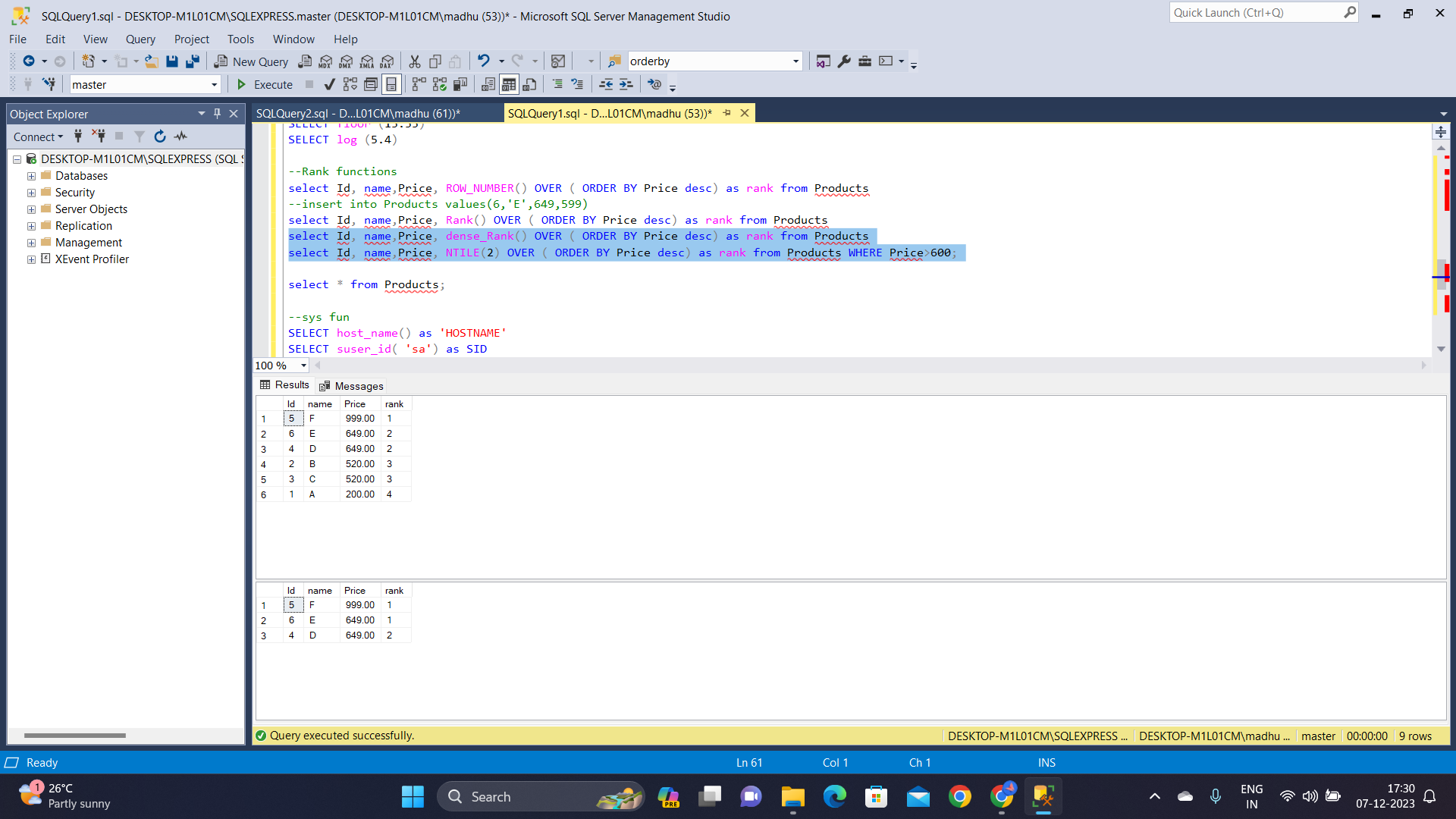


**Rank functions**

These queries assign a ranking to each product in the "Products" table based on their descending prices. The first query uses the **ROW\_NUMBER()** function, and the second uses the **RANK()** function. Both functions generate a numerical rank for each product, reflecting their positions in the ordered list of prices.

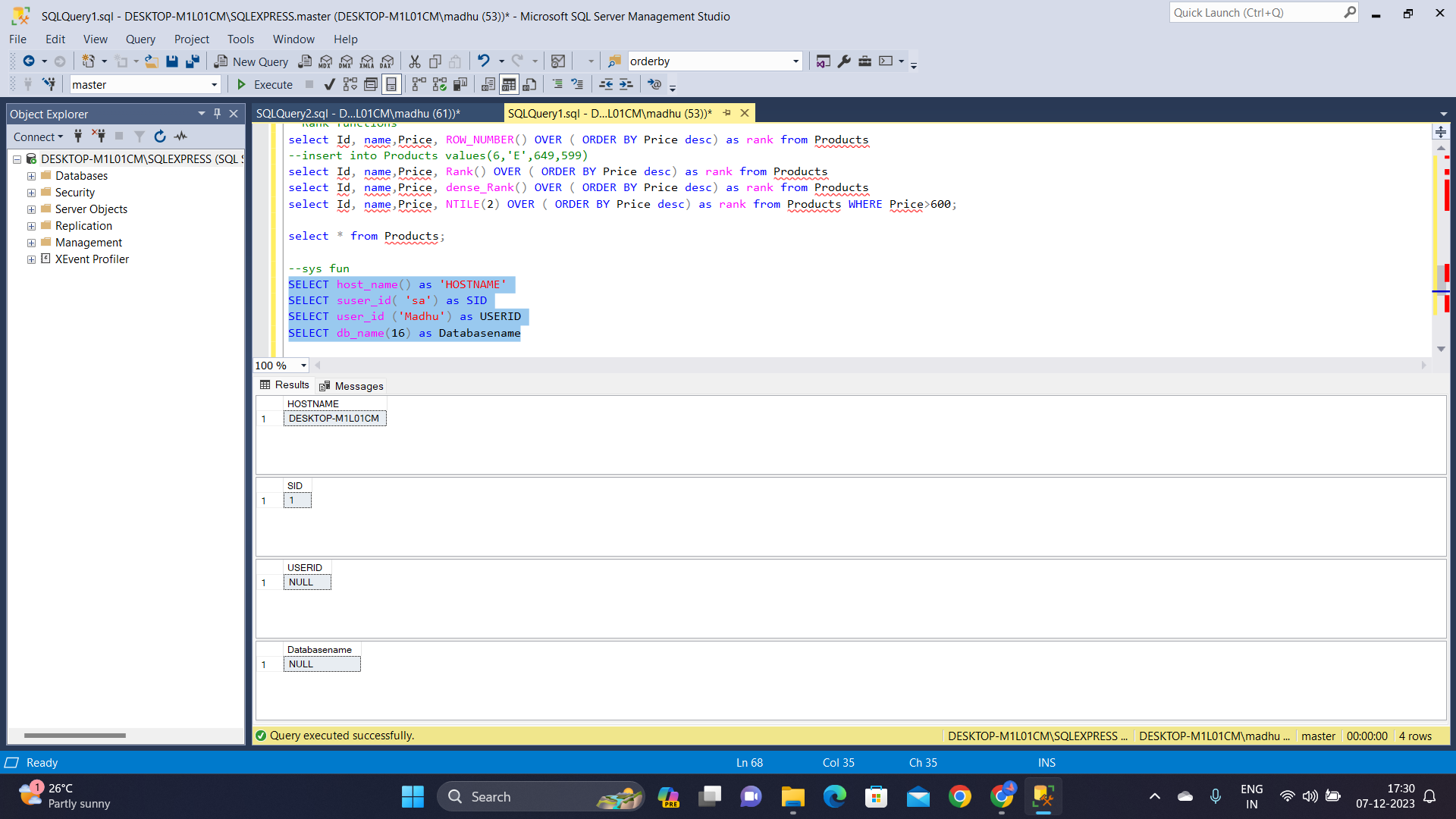


Here, we assign rankings to products based on descending prices. The first query uses the **DENSE\_RANK()** function, and the second employs the **NTILE(2)** function to divide products into two price based groups, considering only those with prices exceeding 600.



**System functions**

Below queries get information related to the SQL Server environment. The first obtains the **host name**, the second fetches the security identifier (SID) for the 'sa' login, the third acquires the user ID for 'Madhu', and the fourth retrieves the database name for the database with ID 16.



**Aggregate functions**

These queries calculate various statistical measures for the "Products" table. The first set computes the **avg()** and **count()** of prices and offer prices, as well as **distinct** price values. The second set determines the **max** and **min** prices and offer prices, retrieves product details with the minimum offer price, and calculates the **sum** of prices and offer prices.

