📘 Module 26: String Functions

🧾 What are String Functions?

String functions in SQL are used to **manipulate text data** (also called character or `varchar` data).

These functions help format, clean, extract, or combine text values in your queries.

1️⃣ `LEN()` – Length

✅ Definition:

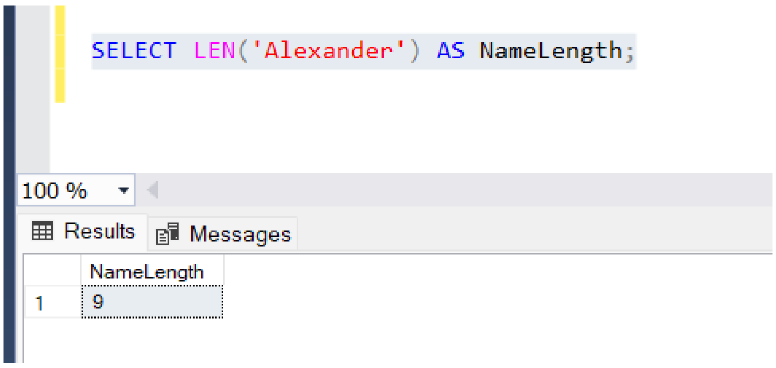
Returns the **number of characters** in a string (excluding trailing spaces).

✅ Syntax:

LEN(string)

📌 Example:

1. SELECT LEN('Alexander') AS NameLength;

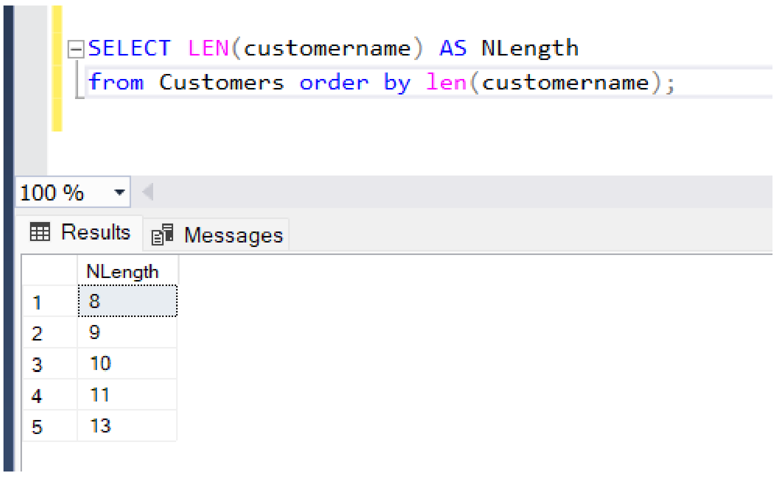


```
SELECT LEN('Alexander') AS NameLength;
```

100 % | Results | Messages

	NameLength
1	9

2. SELECT LEN(customername) AS NLength
   from Customers order by len(customername);



```
SELECT LEN(customername) AS NLength
from Customers order by len(customername);
```

100 % | Results | Messages

	NLength
1	8
2	9
3	10
4	11
5	13

2️⃣ UPPER() and LOWER()

✅ Definition:

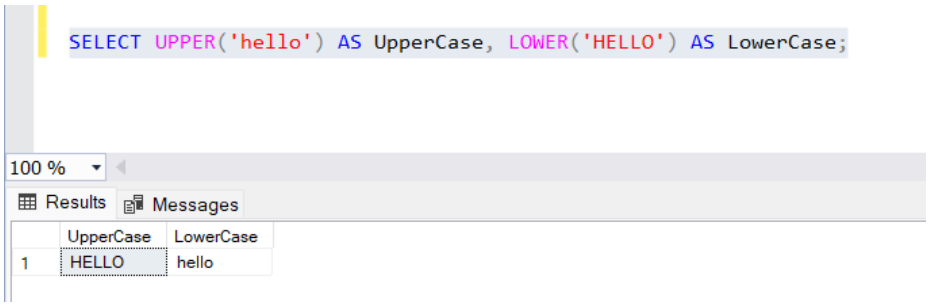
- UPPER() converts text to **uppercase**
- LOWER() converts text to **lowercase**

✅ Syntax:

UPPER(string)
LOWER(string)

📌 Example:

1. SELECT UPPER('hello') AS UpperCase, LOWER('HELLO') AS LowerCase;

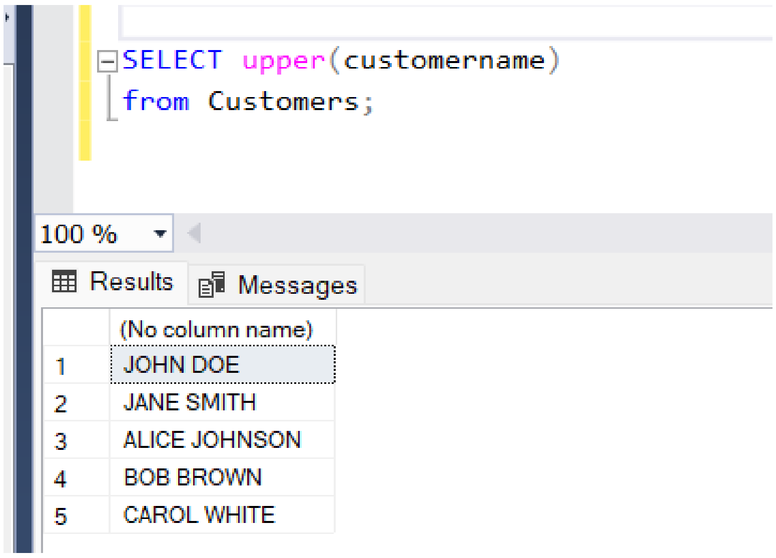


```
SELECT UPPER('hello') AS UpperCase, LOWER('HELLO') AS LowerCase;
```

```
100 %
Results  Messages
   UpperCase  LowerCase
1  HELLO      hello
```

2. SELECT upper(customername)

   from Customers;



```
SELECT upper(customername)
from Customers;
```

```
100 %
Results  Messages
   (No column name)
1  JOHN DOE
2  JANE SMITH
3  ALICE JOHNSON
4  BOB BROWN
5  CAROL WHITE
```

3️⃣ REPLACE()

✅ Definition:

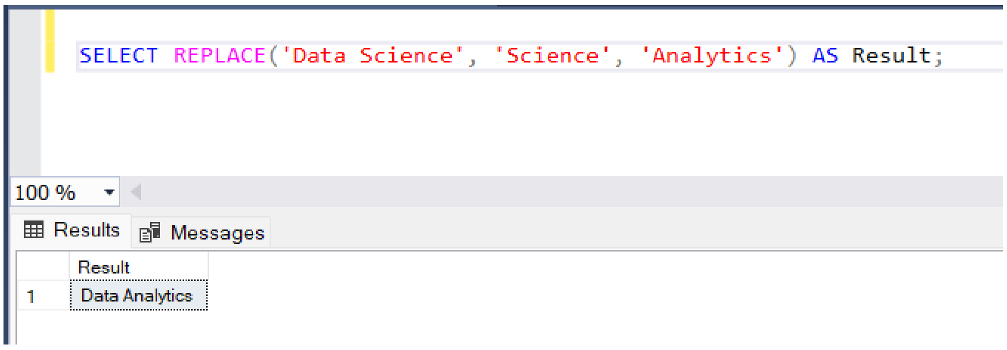
Replaces all **occurrences** of a substring within a string.

✅ Syntax:

REPLACE(original_string, string_to_replace, replacement_string)

📌 Example:

```
SELECT REPLACE('Data Science', 'Science', 'Analytics') AS Result;
-- Output: 'Data Analytics'
```



```
SELECT REPLACE('Data Science', 'Science', 'Analytics') AS Result;
```

```
   Result
1  Data Analytics
```

4️⃣ TRIM() , LTRIM() , RTRIM()

✅ Definitions:

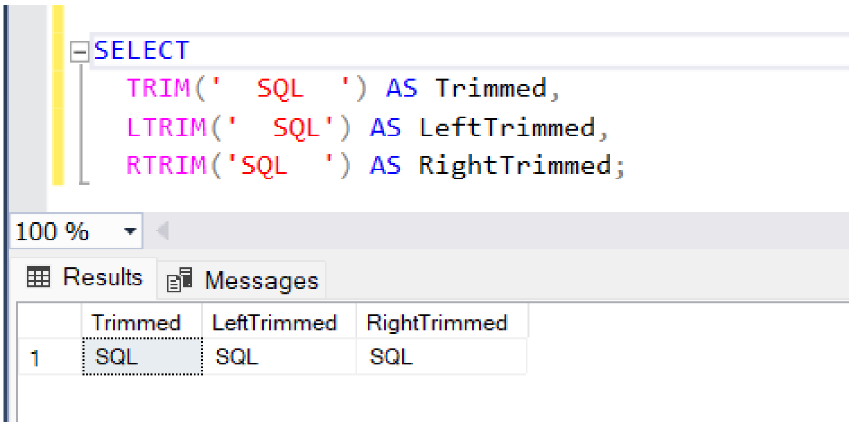
- TRIM() – removes **both leading and trailing spaces**
- LTRIM() – removes spaces from the **left**
- RTRIM() – removes spaces from the **right**

✅ Syntax:

```
TRIM(string)
LTRIM(string)
RTRIM(string)
```

📌 Example:

```
SELECT
  TRIM('  SQL  ') AS Trimmed,
  LTRIM('  SQL') AS LeftTrimmed,
  RTRIM('SQL  ') AS RightTrimmed;
```



```
SELECT
    TRIM('   SQL   ') AS Trimmed,
    LTRIM('   SQL') AS LeftTrimmed,
    RTRIM('SQL   ') AS RightTrimmed;
```

```
   Trimmed  LeftTrimmed  RightTrimmed
1  SQL      SQL          SQL
```

5️⃣ String Concatenation – `+` or `CONCAT()`

✅ Definition:

Combines two or more strings into one.

✅ Syntax:

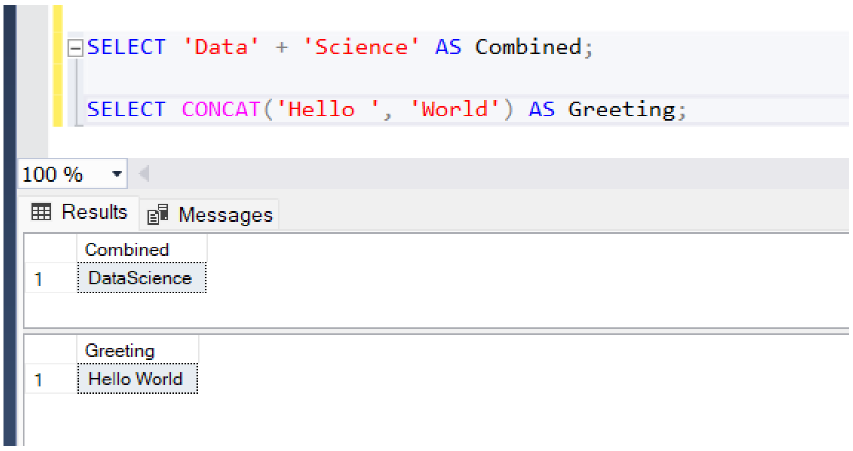
```
-- Method 1: Using +
string1 + string2

-- Method 2: Using CONCAT()
CONCAT(string1, string2, ...)
```

📌 Example:

```
SELECT 'Data' + 'Science' AS Combined;
-- Output: 'DataScience'

SELECT CONCAT('Hello ', 'World') AS Greeting;
-- Output: 'Hello World'
```



```
SELECT 'Data' + 'Science' AS Combined;

SELECT CONCAT('Hello ', 'World') AS Greeting;
```

```
   Combined
1  DataScience

   Greeting
1  Hello World
```

6️⃣ SUBSTRING()

✅ Definition:

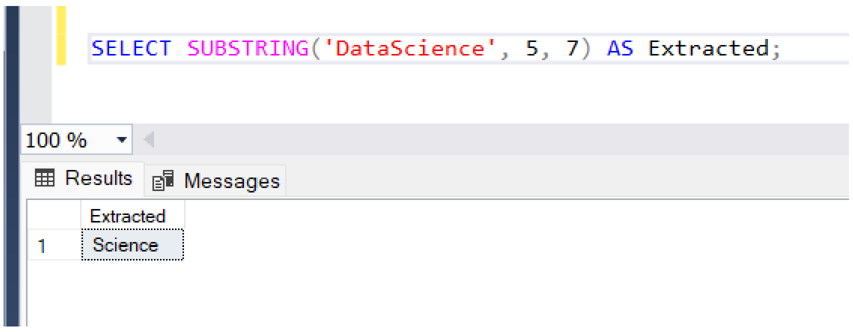
Extracts a part of a string starting from a specific position.

✅ Syntax:

SUBSTRING(string, start_position, length)

📌 Example:

SELECT SUBSTRING('DataScience', 5, 7) AS Extracted;



```
SELECT SUBSTRING('DataScience', 5, 7) AS Extracted;
```

100 %

Results | Messages

```
   Extracted
1  Science
```

7️⃣ STRING_AGG() – String Aggregation

✅ Definition:

Combines multiple string values from **rows** into a **single string**, separated by a delimiter (SQL Server 2017+).

✅ Syntax:

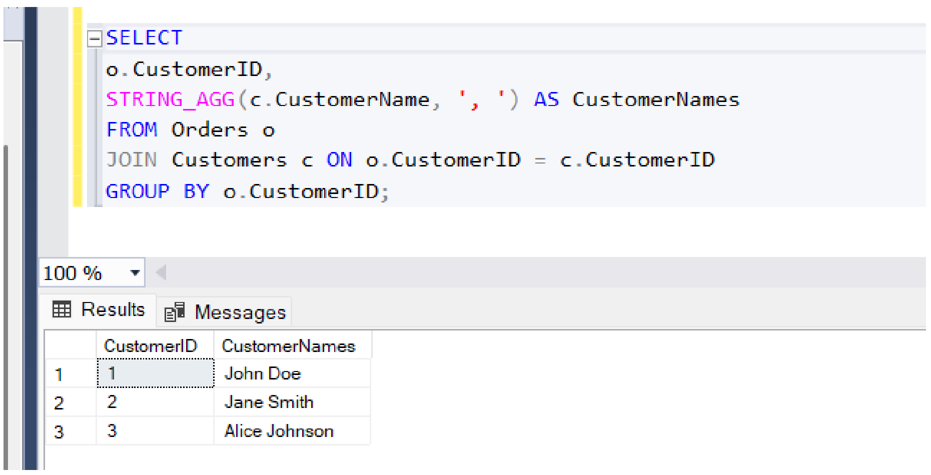
STRING_AGG(column_name, 'separator') AS alias

📌 Example:

SELECT
o.CustomerID,
STRING_AGG(c.CustomerName, ', ') AS CustomerNames
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY o.CustomerID;

🧾 Explanation:

- `STRING_AGG(c.CustomerName, ', ')` combines customer names with a comma and space.
- Grouped by `CustomerID` to show **which customers placed orders**.
- Even if a customer placed multiple orders, their name appears once per ID (because of grouping).



```sql
SELECT
o.CustomerID,
STRING_AGG(c.CustomerName, ', ') AS CustomerNames
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY o.CustomerID;
```

	CustomerID	CustomerNames
1	1	John Doe
2	2	Jane Smith
3	3	Alice Johnson

🧠 Key Points to Remember

✔ `LEN()` does not count trailing spaces

✔ Use `TRIM()` to clean messy data

✔ `REPLACE()` works for partial substitutions inside strings

✔ Use `+` for simple joins and `CONCAT()` when working with NULLs (it handles NULL safely)

✔ `SUBSTRING()` uses **1-based indexing** (starts at 1, not 0)

✔ `STRING_AGG()` is **very useful in reports** – shows grouped data in a single row

✔ Combine string functions for more powerful results (e.g., `UPPER(SUBSTRING(...))`)