SQL String Functions

String functions in SQL are used to manipulate and analyze text data. These functions are helpful for operations such as formatting, searching, extracting, and transforming strings.

---

1. CONCAT()

- **Description:** Combines two or more strings into one.
- **Syntax:**

```sql
SELECT CONCAT(string1, string2, ...) AS combined_string;
```

- **Example:**

```sql
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM employees;
```

Result: Combines `first_name` and `last_name` with a space in between.

---

2. LENGTH() (or LEN() in some databases)

- **Description:** Returns the length of a string in characters.
- **Syntax:**

```sql
SELECT LENGTH(string) AS string_length;
```

- **Example:**

```sql
SELECT LENGTH(name) AS name_length
FROM employees;
```

Result: Length of each employee's name.

---

3. LOWER() / UPPER()

- **Description:** Converts a string to lowercase or uppercase.
- **Syntax:**

```sql
SELECT LOWER(string) AS lowercase_string;
SELECT UPPER(string) AS uppercase_string;
```

- **Example:**

```sql
SELECT LOWER(first_name) AS lowercase_name,
       UPPER(last_name) AS uppercase_name
FROM employees;
```

---

4. SUBSTRING() / SUBSTR()

- **Description:** Extracts a part of a string from a specified starting position and length.
- **Syntax:**

```sql
SELECT SUBSTRING(string, start_position, length) AS
substring;
```

- **Example:**

```sql
SELECT SUBSTRING(name, 1, 3) AS first_three_chars
FROM employees;
```

Result: Extracts the first three characters of the name.

---

5. TRIM()

- **Description:** Removes leading and trailing spaces (or specified characters) from a string.
- **Syntax:**

```sql
SELECT TRIM(string) AS trimmed_string;
```

- **Example:**

```sql
SELECT TRIM('  John Doe  ') AS cleaned_name;
```

Result: `'John Doe'` without leading or trailing spaces.

- **LTRIM():** Removes only leading spaces.

```sql
SELECT LTRIM('  John') AS trimmed_left;
```

- **RTRIM():** Removes only trailing spaces.

```sql
SELECT RTRIM('John  ') AS trimmed_right;
```

6. REPLACE()

- **Description:** Replaces occurrences of a substring with another substring.
- **Syntax:**

```sql
SELECT REPLACE(string, old_substring, new_substring)
AS replaced_string;
```

- **Example:**

```sql
SELECT REPLACE(name, 'John', 'Jonathan') AS
updated_name
FROM employees;
```

Result: Replaces all occurrences of 'John' with 'Jonathan'.

7. POSITION() (or CHARINDEX() in SQL Server)

- **Description:** Finds the position of a substring within a string (case-sensitive).
- **Syntax:**

```sql
SELECT POSITION(substring IN string) AS position;
```

- **Example:**

```sql
SELECT POSITION('a' IN name) AS first_a_position
FROM employees;
```

Result: Position of the first occurrence of 'a' in each name.

8. LEFT() / RIGHT()

- **Description:** Extracts a specified number of characters from the start or end of a string.
- **Syntax:**

```sql
SELECT LEFT(string, number) AS left_part;
SELECT RIGHT(string, number) AS right_part;
```

- **Example:**

```sql
SELECT LEFT(name, 4) AS first_four_chars,
       RIGHT(name, 4) AS last_four_chars
FROM employees;
```

9. REVERSE()

- **Description:** Reverses the characters in a string.
- **Syntax:**

```sql
SELECT REVERSE(string) AS reversed_string;
```

- **Example:**

```sql
SELECT REVERSE(name) AS reversed_name
FROM employees;
```

10. FORMAT()

- **Description:** Formats a string or numeric value based on a specified format (common in SQL Server).
- **Syntax:**

```sql
SELECT FORMAT(value, 'format') AS formatted_value;
```

- **Example:**

```sql
SELECT FORMAT(123456789, '###,###') AS formatted_number;
```

Result: `123,456,789`

11. STRING Functions with Aggregates

You can use string functions with aggregate functions like `GROUP BY`.

Example: Group Employees by Uppercase Department Names

```sql
SELECT UPPER(department) AS department_name,
       COUNT(employee_id) AS total_employees
FROM employees
GROUP BY UPPER(department);
```

12. CONCAT_WS() (MySQL Specific)

- **Description:** Concatenates strings with a separator.
- **Syntax:**

```sql
SELECT CONCAT_WS(separator, string1, string2, ...) AS combined_string;
```

- **Example:**

```sql
SELECT CONCAT_WS(', ', first_name, last_name) AS full_name
FROM employees;
```

Result: Combines names with a comma and space as separators.

---

13. CASE for String Manipulation

- **Description:** Conditionally manipulate strings.
- **Example:**

```sql
SELECT name,
       CASE
         WHEN LENGTH(name) > 10 THEN 'Long Name'
         ELSE 'Short Name'
       END AS name_category
FROM employees;
```

Practical Tips:

1. **Use TRIM** to clean messy input strings.
2. **Combine CONCAT** for formatted outputs in reports.
3. **Use REPLACE** to sanitize data like email or phone numbers.
4. **Use SUBSTRING** for extracting portions like area codes from phone numbers.

Would you like examples using a sample dataset?