Chapter: String Manipulation Magic

Topic: String Functions in SQL

Section 1: Learn

1.1 What Are String Functions in SQL?

In SQL, string functions are **predefined operations** that allow us to manipulate and process textual data stored in string-type columns like VARCHAR, CHAR, or TEXT. These functions are essential when dealing with names, emails, addresses, descriptions, or any other kind of text.

These functions help in:

- **Cleaning** or **standardizing** messy data.
- **Reformatting** data to match reporting or output needs.
- Extracting **meaningful parts** of a string.
- **Combining multiple columns** into one for better presentation.

1.2 CONCAT(): Combining Strings

What It Does:

- Joins two or more strings into a single string.

Syntax:

```
CONCAT(string1, string2, ..., stringN)
```

Example:

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

Combines the first_name and last_name with a space in between.

Use Case:

In reports, we often want to show the **full name** instead of separate first and last names. This function is perfect for that.

1.3 SUBSTRING(): Extracting Part of a String

What It Does:

- Returns a part of a string starting from a given position for a specified length.

Syntax:

```
SUBSTRING(string, start_position, length)
```

Example:

```
SELECT SUBSTRING(email, 1, 5) AS start_email
FROM users;
```

Extracts the first 5 characters of the email field.

Use Case:

Used when we want to extract:

- Initials from names

- Area code from phone numbers
- Domain from email IDs

1.4 LENGTH(): Counting Characters in a String

What It Does:

- Returns the number of characters in a string.

Syntax:

```
LENGTH(string)
```

Example:

```
SELECT LENGTH(name) AS name_length

FROM customers;
```

Gives the number of characters in each name.

Use Case:

Helpful in:

- Validating string length (e.g., check if passwords are at least 8 characters).
- Filtering out unusually long or short entries.

1.5 LOCATE(): Finding the Position of a Substring

What It Does:

- Returns the position of the first occurrence of a substring within a string.

Syntax:

```
LOCATE(substring, string)
```

Example:

```
SELECT LOCATE('@', email) AS at_position
FROM users;
```

> Finds the position of '@' in each email ID.

Use Case:

- Used with SUBSTRING() to extract dynamic portions of strings based on symbol location.
- Common in email domain extractions or URL parsing.

1.6 CONCAT_WS(): Combine with Separator

What It Does:

- Similar to CONCAT(), but allows specifying a separator.

Syntax:

```
CONCAT_WS(separator, string1, string2, ...)
```

Example:

```
SELECT CONCAT_WS('-', area_code, phone_number) AS formatted_phone
FROM contacts;
```

> Combines the area code and phone number with a dash in between.

Use Case:

This is particularly useful for formatting:

- Phone numbers
- Dates
- Codes (e.g., invoice numbers like INV-2024-001)

1.7 UPPER() and LOWER(): Changing Case

What They Do:

- UPPER() converts all characters to uppercase.
- LOWER() converts all characters to lowercase.

Example:

```
SELECT UPPER(name), LOWER(name)

FROM users;
```

Useful for creating consistent case formats in output.

Section 2: Practise

Exercise 1: Combine First and Last Names

Display full names by combining first and last names.

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name

FROM employees;
```

Exercise 2: Extract Email Provider (e.g., gmail, yahoo)

```
SELECT SUBSTRING(email, LOCATE('@', email) + 1, LOCATE('.', email) -
LOCATE('@', email) - 1) AS provider
FROM users;
```

Exercise 3: Validate Name Lengths

Find all customers whose names are longer than 10 characters.

```
SELECT name
FROM customers
WHERE LENGTH(name) > 10;
```

Exercise 4: Format Phone Number with Dash

Display phone numbers in the format: area-code-number.

```
SELECT CONCAT_WS('-', area_code, phone_number) AS formatted_number
FROM contacts;
```

Exercise 5: Convert to Uppercase and Lowercase

```
SELECT name, UPPER(name) AS upper_name, LOWER(name) AS lower_name
FROM students;
```

Exercise 6: Generate Student Usernames

Create a user ID using the first three letters of the student’s name and last 4 digits of their mobile number.

```
SELECT CONCAT(SUBSTRING(name, 1, 3), SUBSTRING(phone, -4)) AS
username
FROM students;
```

Section 3: FAQ – Know More

Q1. What happens if one of the fields in CONCAT() is NULL?

- If any argument in CONCAT() is NULL, the entire result becomes NULL.
- To avoid this, use CONCAT_WS() which skips NULL values.

Q2. Can we extract text between two symbols using SQL functions?

Yes, by using SUBSTRING() and LOCATE() together.

```
-- Extract domain from email
SELECT SUBSTRING(email, LOCATE('@', email) + 1)
FROM users;
```

Q3. How do I trim whitespace from a string?

Use the TRIM() function:

```
SELECT TRIM('   Hello   ') AS trimmed;
```

Returns 'Hello' with spaces removed.

Q4. Are string functions case-sensitive?

Yes, by default. For case-insensitive comparisons, you can:

- Use LOWER() or UPPER() before comparison.

```
SELECT * FROM users

WHERE LOWER(name) = 'ajay';
```

Q5. What if I want to replace part of a string?

Use REPLACE():

```
SELECT REPLACE(name, 'a', '@') AS modified_name

FROM users;
```

Replaces all 'a' with '@' in names.

End of Notes for Chapter: String Manipulation Magic