EXPLORE AI ACADEMY

SQL string, date, and miscellaneous functions

String functions

Introduction

String functions in SQL are **built-in** functions that operate on **string data types** (for example, VARCHAR, CHAR, TEXT) and allow us to manipulate and work with strings.

- We often work with data that are **not in a format or structure** that is **immediately usable** for the use case at hand.
- **SQL string manipulation** assists in turning unstructured data into a structured format so that **generic transformations** can be performed on the data.

Benefits of SQL string functions

- **Data manipulation**: Transform string data to meet specific requirements.
- **Data cleansing**: Standardise data for improved quality and consistency.
- **Query flexibility**: Construct complex queries by combining string functions with other SQL elements.
- **Reporting and analysis**: Extract meaningful information and present structured data.
- **Database maintenance**: Efficiently update, modify, and correct data values within tables.
- **Compatibility**: Ensure compatibility across different database management systems.

Data overview

To explain SQL string functions, we will use a table, called **`Water_sources_sa_2022`**, that represents **water sources in South Africa for the year 2022**, their **types** (surface water or groundwater), and their **availability levels** (high, medium, or low).

Source_id	Source_name	Water_type	Availability
1	Orange River	Surface Water	High
2	Karoo Aquifer	Groundwater	Medium
3	Vaal Dam	Surface Water	Medium
4	Table Mountain Spring	Groundwater	Low
5	Kruger National Park River	Surface Water	High
6	Cape Town Reservoir	Surface Water	Low

UPPER() and LOWER() functions

The **UPPER()** function is used to **convert a string to uppercase** while the **LOWER()** function is used to **convert a string to lowercase**. Their syntaxes are as follows:

```
SELECT
    UPPER(string) AS Alias
FROM
    Table_name;
```

```
SELECT
    LOWER(string) AS Alias
FROM
    Table_name;
```

The strings to be converted.

UPPER() and LOWER() functions

If we wish to change the case of the values in the **`Source_name`** column, we can utilise the **`UPPER()`** and **`LOWER()`** functions.

Query

```
SELECT
     UPPER(Source_name) AS Upper_source_name,
     LOWER(Source_name) AS Lower_source_name
FROM
     Water_sources_sa_2022;
```

Output

Upper_source_name	Lower_source_name
ORANGE RIVER	orange river
KAROO AQUIFER	karoo aquifer
VAAL DAM	vaal dam
TABLE MOUNTAIN SPRING	table mountain spring
KRUGER NATIONAL PARK RIVER	kruger national park river
CAPE TOWN RESERVOIR	cape town reservoir

LTRIM() and RTRIM() functions

The `LTRIM()` function is used to **remove leading spaces from the left** end of a string while the `RTRIM()` function is used to **remove trailing spaces from the right** end of a string.

```
SELECT
    LTRIM(string) AS Alias
FROM
    Table_name;
```

```
SELECT
    RTRIM(string) AS Alias
FROM
    Table_name;
```

The strings with leading or trailing spaces.

LTRIM() function

If we intend to eliminate the leading and trailing spaces from the column **`Water_type`**, we can utilise the **`LTRIM()`** and **`RTRIM()`** functions respectively.

Query

```
SELECT
      LTRIM(RTRIM(Water_type)) AS Trimmed_water_type
FROM
      Water_sources_sa_2022;
```

Compare the **`Trimmed_water_type`** column to the original column, **`Water_type`**. Do you notice the spaces that have been removed?

Output

Trimmed_water_type
Surface Water
Groundwater
Surface Water
Groundwater
Surface Water
Surface Water

LENGTH() function

The **LENGTH()** function is used to **determine the length** (number of characters) of a string. It counts white spaces as part of the string length.

```
SELECT
    LENGTH(string) AS Alias
FROM
    Table_name;
```

The string whose length is to be determined.

LENGTH() function

If we want to determine the length of the names for all water sources, we can employ the **`LENGTH()`** function.

Query

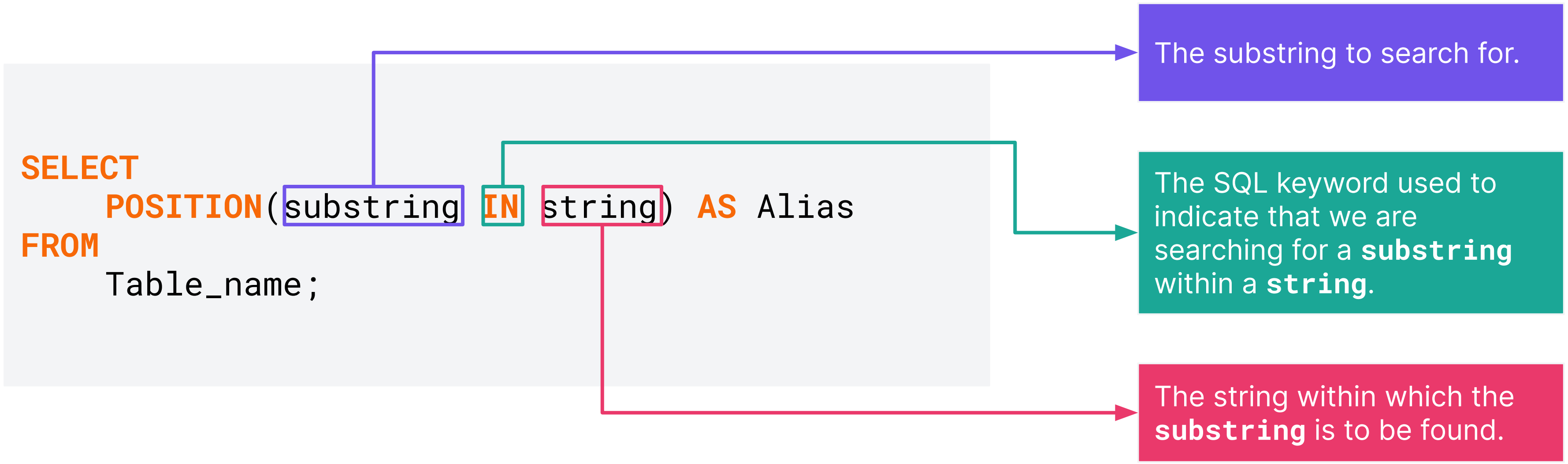
```
SELECT
      Source_name,
      LENGTH(Source_name) AS Name_length
FROM
      Water_sources_sa_2022;
```

Output

Source_name	Name_length
Orange River	12
Karoo Aquifer	13
Vaal Dam	8
Table Mountain Spring	21
Kruger National Park River	26
Cape Town Reservoir	19

POSITION() function

The **`POSITION()`** function is used to return the **position (index) of the first occurrence of a substring within a string**. It takes **two arguments**: the **substring** to search for and the **string** in which to search for that substring. It returns **`0`** if the substring is not found.



```
SELECT
    POSITION(substring IN string) AS Alias
FROM
    Table_name;
```

The substring to search for.

The SQL keyword used to indicate that we are searching for a **`substring`** within a **`string`**.

The string within which the **`substring`** is to be found.

POSITION() function

If we aim to locate the position, if any, of the word “River” in all entries of the **Source_name** column, we can utilise the **POSITION()** function.

Query

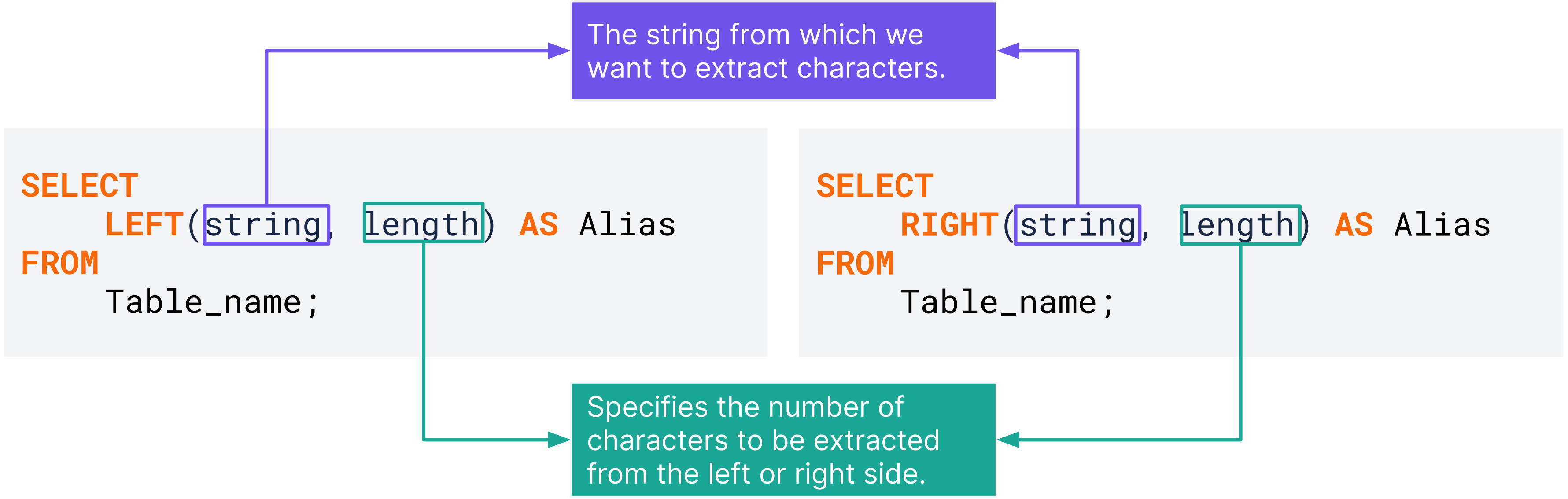
```
SELECT
     Source_name,
     POSITION('River' IN Source_name) AS
     Position
FROM
     Water_sources_sa_2022;
```

Output

Source_name	Position
Orange River	8
Karoo Aquifer	0
Vaal Dam	0
Table Mountain Spring	0
Kruger National Park River	22
Cape Town Reservoir	0

LEFT() and RIGHT() functions

The `LEFT()` function is used to extract a **specified number of characters from the beginning (leftmost side) of a string** while the `RIGHT()` function is used to extract a **specified number of characters from the end (rightmost side) of a string**.



```
SELECT
    LEFT(string, length) AS Alias
FROM
    Table_name;
```

```
SELECT
    RIGHT(string, length) AS Alias
FROM
    Table_name;
```

The string from which we want to extract characters.
Specifies the number of characters to be extracted from the left or right side.

LEFT() and RIGHT() functions

If we intend to retrieve the initial five characters and the last four characters from each entry in the **Source_name** column, including any white space characters, we can use the **LEFT()** and **RIGHT()** functions respectively.

Query

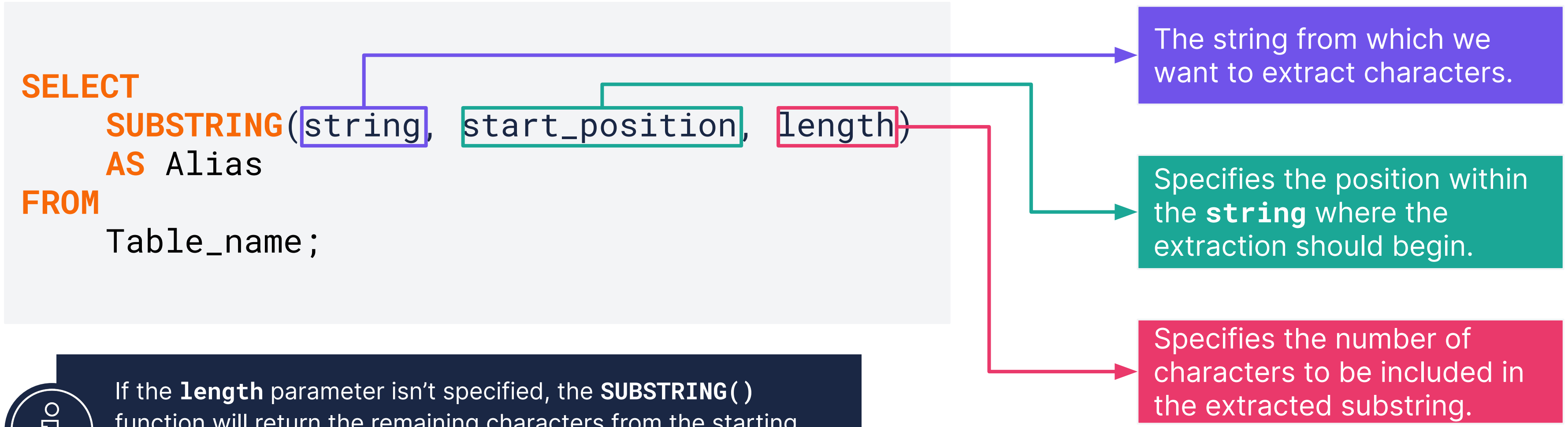
```
SELECT
     Source_name,
     LEFT(Source_name, 5) AS Left_name,
     RIGHT(Source_name, 4) AS Right_name
FROM
     Water_sources_sa_2022;
```

Output

Source_name	Left_name	Right_name
Orange River	Orang	iver
Karoo Aquifer	Karoo	ifer
Vaal Dam	Vaal	Dam
Table Mountain Spring	Table	ring
Kruger National Park River	Kruge	iver
Cape Town Reservoir	Cape	voir

SUBSTRING() function

The **SUBSTRING()** function is used to **extract a substring from a string**. It takes three arguments: the **original string**, the **starting position of the substring**, and optionally, the **length of the substring**.



```
SELECT
    SUBSTRING(string, start_position, length)
    AS Alias
FROM
    Table_name;
```

The string from which we want to extract characters.

Specifies the position within the **string** where the extraction should begin.

Specifies the number of characters to be included in the extracted substring.

If the **length** parameter isn't specified, the **SUBSTRING()** function will return the remaining characters from the starting position to the end of the string.

SUBSTRING() function

To obtain a substring from the **Source_name** column that starts at the first position and spans five characters (including white spaces), we can utilise the **SUBSTRING()** function.

Query

```
SELECT
     Source_name,
     SUBSTRING(Source_name, 1, 5) AS
     Extracted_string
FROM
     Water_sources_sa_2022;
```

Output

Source_name	Extracted_string
Orange River	Orang
Karoo Aquifer	Karoo
Vaal Dam	Vaal
Table Mountain Spring	Table
Kruger National Park River	Kruge
Cape Town Reservoir	Cape

CONCAT() function

The `CONCAT()` function is used to **concatenate or join multiple strings together**. It takes **two or more string arguments** (separated by commas) and returns a **single concatenated string**.

```
SELECT
    CONCAT(string1, string2, ...) AS Alias
FROM
    Table_name;
```

The strings you want to concatenate. You can provide multiple strings as arguments.

CONCAT() function

To provide a summary of the availability status for all water sources, we can combine the entries from the **Source_name** column with their corresponding values from the **Availability** column using the **CONCAT()** function.

Query

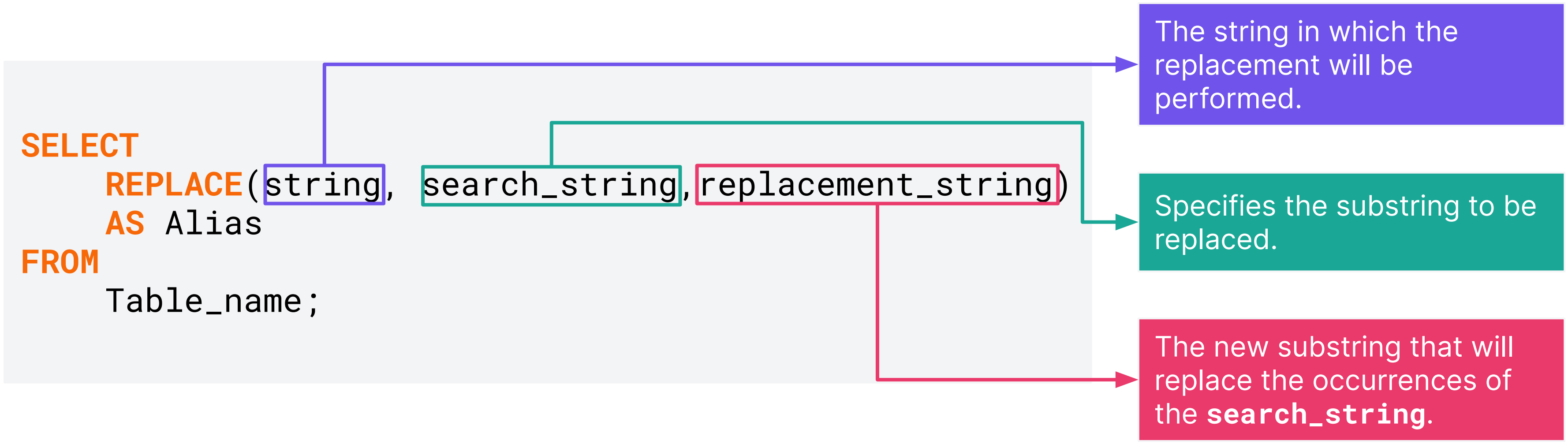
```
SELECT
    CONCAT(Source_name, ' availability is ',
    Availability) AS Availability_status
FROM
    Water_sources_sa_2022;
```

Output

Availability_status
Orange River availability is High
Karoo Aquifer availability is Medium
Vaal Dam availability is Medium
Table Mountain Spring availability is Low
Kruger National Park River availability is High
Cape Town Reservoir availability is Low

REPLACE() function

The `REPLACE()` function is used to **replace all occurrences of a specified substring within a string with a new substring.** It takes three arguments: the **original string**, the **substring to be replaced**, and the **new substring**.



```
SELECT
    REPLACE(string, search_string,replacement_string)
    AS Alias
FROM
    Table_name;
```

`string`: The string in which the replacement will be performed.

`search_string`: Specifies the substring to be replaced.

`replacement_string`: The new substring that will replace the occurrences of the **`search_string`**.

REPLACE() function

To replace the word “River” with the word “Lake” on all entries of the **Source_name** column, we can use the **REPLACE()** function.

Query

```
SELECT
      Source_name,
      REPLACE(Source_name, 'River', 'Lake')
      AS Modified_name
FROM
      Water_sources_sa_2022;
```

Output

Source_name	Modified_name
Orange River	Orange Lake
Karoo Aquifer	Karoo Aquifer
Vaal Dam	Vaal Dam
Table Mountain Spring	Table Mountain Spring
Kruger National Park River	Kruger National Park Lake
Cape Town Reservoir	Cape Town Reservoir