Reference card

SQL functions and CFC commands

Miscellaneous functions

Miscellaneous functions are **supporting functions** in SQL that can be used with various data types.

CAST()

Converts a value **from its current data type into a specified data type.**

```
SELECT
      CAST(expression AS datatype)
FROM
      Table_name;
```

CONVERT()

Another function that **converts** a value **from its current data type into a specified data type.**

```
SELECT
      CONVERT(value, datatype)
FROM
      Table_name;
```

IFNULL()

Returns a specified **value if the given expression is null.** Otherwise, it returns the value of the expression itself.

```
SELECT
      IFNULL(expression, alternative_value)
FROM
      Table_name;
```

NULLIF()

Compares two expressions and returns `NULL` if they are **equal.** Otherwise, the first expression is returned.

```
SELECT
      NULLIF(expression1, expression2)
FROM
      Table_name;
```

ISNULL()

Determines if an **expression is** `NULL` or not. If the expression is `NULL`, it returns **1. Otherwise,** it returns **0.**

```
SELECT
      ISNULL(expression)
FROM
      Table_name;
```

COALESCE()

Evaluates expressions from left to right, **returning the first non-**`NULL` value or `NULL` **if all expressions are** `NULL`.

```
SELECT
      COALESCE(value1, value1, ...)
FROM
      Table_name;
```

String functions

String functions are used to **manipulate** and **format string data types** to ensure consistency.

UPPER()

Converts a **string** to **uppercase.**

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

LOWER()

Converts a **string** to **lowercase.**

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

LTRIM()

Removes leading spaces from the **left** end of a string.

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

RTRIM()

Removes trailing spaces from the **right** end of a string.

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

LENGTH()

Determines the length (number of characters) of a **string.**

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

POSITION()

Returns the **position** (index) of the **first occurrence** of a **substring** within a string.

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

LEFT()

Extracts a **specified number of characters** from the **leftmost** side of a **string.**

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

RIGHT()

Extracts a **specified number of characters** from the **rightmost** side of a **string.**

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

SUBSTRING()

Extracts a **substring** from a **string.**

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

CONCAT()

Concatenates or joins **multiple strings together.**

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

REPLACE()

Replaces all occurrences of a **specified substring** within a string with a **new substring.**

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

Datetime functions

Datetime functions allow **manipulation** and **calculation** of **date** and **time values.**

CURRENT_DATE()

Returns the **current date** without the time component. The function is specific to MySQL .

```
SELECT
      CURRENT_DATE() AS Current_date;
```

NOW()

Returns the **current date and time** from the system. The function is specific to MySQL.

```
SELECT
      NOW() AS Current_date;
```

CURRENT_TIMESTAMP()

Returns the **current date and time** from the system. The function is specific to MySQL.

```
SELECT
      CURRENT_TIMESTAMP() AS Current_timestamp;
```

DAY()

Returns the **day** of the month for a specified date.

```
SELECT
      DAY(date_expression) AS Alias;
FROM
      Table_name;
```

MONTH()

Returns the **month** for a specified date.

```
SELECT
      MONTH(date_expression) AS Alias;
FROM
      Table_name;
```

YEAR()

Returns the **year** for a specified date.

```
SELECT
      YEAR(date_expression) AS Alias;
FROM
      Table_name;
```

DATEDIFF()

Calculates the **difference between two dates.**

```
SELECT
      DATEDIFF(date_part, start_date, end_date)
      AS Alias;
FROM
      Table_name;
```

DATE_ADD()

Adds a specified **interval** to a date or datetime value.

```
SELECT
      DATE_ADD(date, INTERVAL value INTERVAL_UNIT)
      AS Alias;
FROM
      Table_name;
```

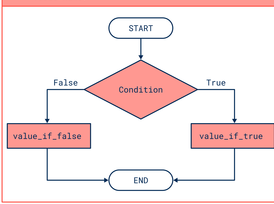
Control flow functions

Control flow functions are used to implement **conditional logic** and **control the flow of execution** within SQL queries. They allow us to perform different actions or return different values **based on specific conditions.**

IF()

Evaluates a condition and returns a particular value if a condition is `TRUE`, or another value if a condition is `FALSE`.

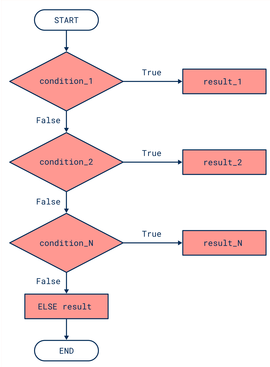
```
SELECT
      IF(condition, value_if_true, value_if_false)
FROM
      Table_name;
```



Searched CASE statement

A list of **conditions** are **evaluated** to either `TRUE` or `FALSE`.

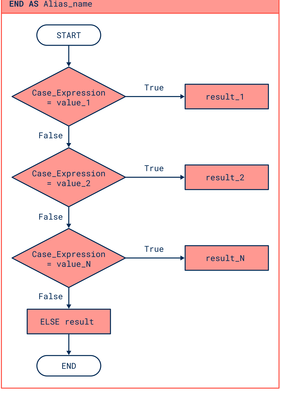
```
CASE Case_Expression
     WHEN condition_1 THEN result_1
     WHEN condition_2 THEN result_2
     WHEN condition_N THEN result_N
     ELSE result
END AS Alias_name
```



Simple CASE statement

A list of **values** are **compared** to a given `CASE` **expression.**

```
CASE Case_Expression
     WHEN value_1 THEN result_1
     WHEN value_2 THEN result_2
     WHEN value_N THEN result_N
     ELSE result
END AS Alias_name
```



Nested conditional statements

One or more conditional statements, like the `IF` and `CASE` control flow functions, **within another conditional statement.**

Nested `IF` statement:	The use of an **`IF` function inside** of another **`IF` function.**
Nested `IF` and `CASE` statement:	The use of an **`IF` function inside** of another **`CASE` function.**