**In Built Functions**

Concat(str1, str2, …)

-- Returns the string that results from concatenating the arguments.

-- It may have one or more arguments.

-- If all arguments are non-binary strings, the result is a non-binary string. If the arguments include any binary strings, the result is a binary string.

SELECT CONCAT('Hello', 'World'); -- o/p : "HelloWorld"

SELECT CONCAT('Hello', ' ', 'World'); -- o/p : "Hello World"

SELECT CONCAT(first\_name, ' ', last\_name) as Employee\_Full\_Name FROM dev\_schema.employee;

Concat\_WS(separator, str1, str2, …)

-- CONCAT\_WS() stands for Concatenate With Separator and is a special form of CONCAT().

-- The first argument is the separator for the rest of the arguments.

-- The separator is added between the strings to be concatenated.

-- The separator can be a string, as can the rest of the arguments. If the separator is NULL, the result is NULL.

SELECT CONCAT\_WS(',', 'Hello', 'World'); -- o/p : "Hello,World"

SELECT CONCAT\_WS('Hello', 'World'); -- o/p : "World"

SELECT CONCAT\_WS(', ', department\_id, department\_name) FROM dev\_schema.department;

Lower(str)

-- Returns the string str with all characters changed to lowercase according to the current character set mapping.

SELECT LOWER('GDWLKGHDSLGHDG'); -- o/p : "gdwlkghdslghdg"

SELECT LOWER(first\_name) as full\_name FROM dev\_schema.employee;

SELECT LOWER(CONCAT(first\_name, ' ', last\_name)) as full\_name FROM dev\_schema.employee;

String Functions

1. ASCII()
2. CHAR\_LENGTH()

-- This string function returns the length of the specified word. It shows the number of characters from the word.

1. CHARACTER\_LENGTH()

-- This string function returns the length of the given string. It shows the number of all characters and spaces from the sentence.

1. CONCAT()
2. CONCAT\_WS()
3. FORMAT()
4. INSERT()
5. INSTR()
6. LCASE()
7. LEFT()
8. LOCATE()
9. LOWER()
10. LPAD()
11. LTRIM()
12. MID()
13. POSITION()
14. REPEAT()
15. REPLACE()
16. REVERSE()
17. RIGHT()
18. RPAD()
19. RTRIM()
20. SPACE()
21. STRCMP()
22. SUBSTR()
23. SUBSTRING()

-- In PostgreSQL, the substring function is used to extract a substring from a string.

1. SUBSTRING\_INDEX()
2. UCASE()
3. UPPER()

Coalesce()

-- In PostgreSQL, the COALESCE function returns the first non-null argument.

-- It is generally used with the SELECT statement to handle null values effectively.

**Syntax:** COALESCE (argument\_1, argument\_2, …);

-- The COALESCE function accepts an unlimited number of arguments.

-- It returns the first argument that is not null.

-- If all arguments are null, the COALESCE function will return null.

-- The COALESCE function evaluates arguments from left to right until it finds the first non-null argument. All the remaining arguments from the first non-null argument are not evaluated.

SELECT job\_title, (COALESCE(max\_salary, 0)-min\_salary/2) as avg\_salary FROM dev\_schema.job;

Cast()

-- PostgreSQL supports a CAST operator that is used to convert a value of one type to another.

**Syntax:** CAST ( expression AS target\_type );

-- The following statement converts a string constant to an integer:

SELECT CAST('100' AS INTEGER);

-- If the expression cannot be converted to the target type, PostgreSQL will raise an error. See the following:

SELECT CAST('10C' AS INTEGER);

-- This example uses the CAST to convert a string to a date:

SELECT CAST ('2020-01-01' AS DATE), CAST ('01-OCT-2020' AS DATE);

-- This example uses the CAST() to convert the string ‘true’, ‘T’ to true and ‘false’, ‘F’ to false:

SELECT CAST('true' AS BOOLEAN), CAST('false' as BOOLEAN), CAST('T' as BOOLEAN), CAST('F' as BOOLEAN);

SUM()

-- PostgreSQL provides with a SUM() function that is used to get the addition of values of a numeric column.

**Syntax:** SUM(column)

-- The following points needs to be kept in mind while using the above function:

* It ignores all NULL values.
* If used with DISTINCT operator as SUM(DISTINCT column), it skips duplicate values.
* Using SUM() function with SELECT clause returns NULL instead of Zero.

SELECT SUM(salary) as total\_salary FROM dev\_schema.employee;

COUNT()

-- The COUNT() function is an **aggregate function** that enables users to get the number of rows that match a particular requirement of a query.

-- Depending upon the user requirements the COUNT() function can have the following syntaxes:

**Syntax:** COUNT(\*)

**Returns:** All rows including NULL and Duplicates

**Syntax:** COUNT(column)

**Returns:**  All rows except NULL.

**Syntax:** COUNT(DISTINCT column)

**Returns:** All rows without NULL and Duplicates

-- The COUNT() function is used with the SELECT statement.

SELECT COUNT(\*) FROM dev\_schema.employee;

SELECT COUNT(DISTINCT fk\_department\_id) FROM dev\_schema.employee;

EXTRACT()

-- The extract function() is used to retrieves subfields such as year or hour from date/time values.

-- The source must be a value expression of type timestamp, time, or interval.

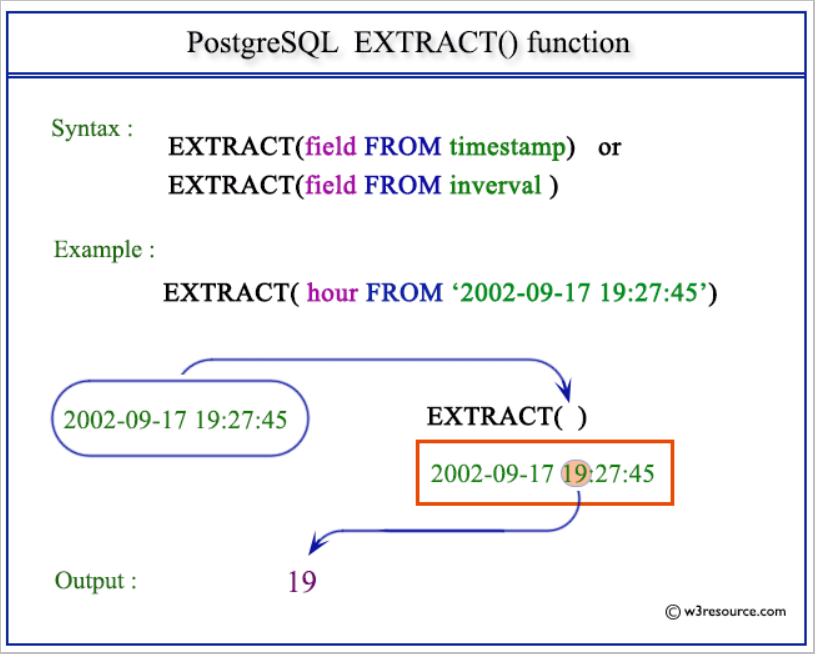
-- The field is an identifier or string that selects what field to be extracted from the source value.

**-- Syntax:**

extract(field from timestamp)

or

extract(field from interval)



-- **EXTRACT() :** century

To extract the century from a given date/time value, you can use the extract() function with the "century" field. The "century" field is an identifier or string that indicates the century subfield.

Code:

SELECT EXTRACT(CENTURY FROM TIMESTAMP '2023-03-11 17:43:17.436');

-- **EXTRACT() :** day

To extract the day from a given date/time value, you can use the EXTRACT function with the "day" field.

Code:

SELECT EXTRACT(DAY FROM TIMESTAMP '2023-03-11 17:43:17.436');

-- To extract the number of days from an interval, you can use the EXTRACT function with the "day" field.

SELECT EXTRACT(DAY FROM INTERVAL '19 days 3 minute');

-- **EXTRACT() :** month

To extract the month from a given timestamp, you can use the EXTRACT function with the "month" field.

Code:

SELECT EXTRACT(MONTH FROM TIMESTAMP '2023-03-11 17:43:17.436');

SELECT EXTRACT(MONTH FROM INTERVAL '3 years 5 months');

-- **EXTRACT() :** dow

To retrieve the day of the week (DOW) from a timestamp value the DOW can be used. It represents the day of the week as an integer, where Sunday is 0 and Saturday is 6.

Code:

SELECT EXTRACT(DOW FROM TIMESTAMP '2023-03-11 17:43:17.436');

This query returns the day of the week as a numeric value. For the given timestamp value '2023-03-11 17:43:17.436', the result is 6, indicating that it falls on a Saturday (since Sunday is represented by 0).

-- **EXTRACT() :** epoch

To retrieve the epoch value from a timestamp the EXTRACT function can be used. The epoch value represents the number of seconds that have elapsed since January 1, 1970 (Coordinated Universal Time - UTC).

Code:

SELECT EXTRACT(EPOCH FROM TIMESTAMP WITH TIME ZONE '2023-03-11 17:43:17.436-07');

SELECT EXTRACT(EPOCH FROM TIMESTAMP '2023-03-11 17:43:17.436');

SELECT EXTRACT(EPOCH FROM INTERVAL '3 days 10 hours');

-- **EXTRACT() :** hour

To extract the hour from a given timestamp, time, or interval value, you can use the EXTRACT function with the "hour" field. The hour field is (0 - 23).

Code:

SELECT EXTRACT(HOUR FROM TIMESTAMP '2023-03-11 17:43:17.436');

SELECT EXTRACT(HOUR FROM interval '3 hours 10 minutes');