Lecture 5

Functions.
Conditional Expressions & Operators.

Senior Lecturer: Sholpan Saimassayeva

Overview

- Mathematical functions and operators
- String functions and operators
- Condition expressions and operators

Mathematical functions and operators

Mathematical Functions

Function	Return Type	Description	Example	Result
`abs(x)`	(same as input)	absolute value	`abs(-17.4)`	`17.4`
`cbrt(dp)`	`dp`	cube root	`cbrt(27.0)`	`3`
`ceil(dp or numeric)`	(same as input)	nearest integer greater than or equal to argument	`ceil(-42.8)`	`-42`
`ceiling(dp or numeric)`	(same as input)	nearest integer greater than or equal to argument (same as `ceil`)	`ceiling(-95.3)`	`-95`
`degrees(dp)`	`dp`	radians to degrees	`degrees(0.5)`	`28.6478897565412`
`div(y numeric, x numeric)`	`numeric`	integer quotient of *y*/*x*	`div(9,4)`	`2`
`exp(dp or numeric)`	(same as input)	exponential	`exp(1.0)`	`2.71828182845905`
`floor(dp or numeric)`	(same as input)	nearest integer less than or equal to argument	`floor(-42.8)`	`-43`
`ln(dp or numeric)`	(same as input)	natural logarithm	`ln(2.0)`	`0.693147180559945`
`log(dp or numeric)`	(same as input)	base 10 logarithm	`log(100.0)`	`2`
`log(b numeric, x numeric)`	`numeric`	logarithm to base *b*	`log(2.0, 64.0)`	`6.0000000000`
`mod(y, x)`	(same as argument types)	remainder of *y*/*x*	`mod(9,4)`	`1`

Mathematical Functions

pi()	dp	"π" constant	pi()	3.14159265358979
power(*a* dp, *b* dp)	dp	*a* raised to the power of *b*	power(9.0, 3.0)	729
power(*a* numeric, *b* numeric)	numeric	*a* raised to the power of *b*	power(9.0, 3.0)	729
radians(dp)	dp	degrees to radians	radians(45.0)	0.785398163397448
round(dp or numeric)	(same as input)	round to nearest integer	round(42.4)	42
round(*v* numeric, *s* int)	numeric	round to *s* decimal places	round(42.4382, 2)	42.44
scale(numeric)	integer	scale of the argument (the number of decimal digits in the fractional part)	scale(8.41)	2
sign(dp or numeric)	(same as input)	sign of the argument (-1, 0, +1)	sign(-8.4)	-1
sqrt(dp or numeric)	(same as input)	square root	sqrt(2.0)	1.4142135623731
trunc(dp or numeric)	(same as input)	truncate toward zero	trunc(42.8)	42
trunc(*v* numeric, *s* int)	numeric	truncate to *s* decimal places	trunc(42.4382, 2)	42.43

Mathematical Functions

`width_bucket(`*operand* dp, *b1* dp, *b2* dp, *count* int`)`	int	return the bucket number to which *operand* would be assigned in a histogram having *count* equal-width buckets spanning the range *b1* to *b2*; returns 0 or *count*+1 for an input outside the range	`width_bucket(5.35, 0.024, 10.06, 5)`	3
`width_bucket(`*operand* numeric, *b1* numeric, *b2* numeric, *count* int`)`	int	return the bucket number to which *operand* would be assigned in a histogram having *count* equal-width buckets spanning the range *b1* to *b2*; returns 0 or *count*+1 for an input outside the range	`width_bucket(5.35, 0.024, 10.06, 5)`	3
`width_bucket(`*operand* anyelement, *thresholds* anyarray`)`	int	return the bucket number to which *operand* would be assigned given an array listing the lower bounds of the buckets; returns 0 for an input less than the first lower bound; the *thresholds* array ***must be sorted***, smallest first, or unexpected results will be obtained	`width_bucket(now(), array['yesterday', 'today', 'tomorrow']::timestamptz[])`	2

Random Functions

Function	Return Type	Description
`random()`	`dp`	random value in the range 0.0 <= x < 1.0
`setseed(dp)`	`void`	set seed for subsequent `random()` calls (value between -1.0 and 1.0, inclusive)

Trigonometric Functions

Function (radians)	Function (degrees)	Description
`acos(x)`	`acosd(x)`	inverse cosine
`asin(x)`	`asind(x)`	inverse sine
`atan(x)`	`atand(x)`	inverse tangent
`atan2(y, x)`	`atan2d(y, x)`	inverse tangent of *y*/*x*
`cos(x)`	`cosd(x)`	cosine
`cot(x)`	`cotd(x)`	cotangent
`sin(x)`	`sind(x)`	sine
`tan(x)`	`tand(x)`	tangent

String functions and operators

PostgreSQL FORMAT Function

PostgreSQL **FORMAT**() function formats arguments based on a format string.

The syntax of the PostgreSQL **FORMAT**() function is as follows:

FORMAT(format_string [, format_arg [, ...]])

String Functions and Operators

Function	Return Type	Description	Example	Result
string `||` *string*	text	String concatenation	`'Post' || 'greSQL'`	PostgreSQL
string `||` *non-string* or *non-string* `||` *string*	text	String concatenation with one non-string input	`'Value: ' || 42`	Value: 42
`bit_length(`*string*`)`	int	Number of bits in string	`bit_length('jose')`	32
`char_length(`*string*`)` or `character_length(`*string*`)`	int	Number of characters in string	`char_length('jose')`	4
`lower(`*string*`)`	text	Convert string to lower case	`lower('TOM')`	tom
`octet_length(`*string*`)`	int	Number of bytes in string	`octet_length('jose')`	4
`overlay(`*string* `placing` *string* `from int [for int])`	text	Replace substring	`overlay('Txxxxas' placing 'hom' from 2 for 4)`	Thomas
`position(`*substring* `in` *string*`)`	int	Location of specified substring	`position('om' in 'Thomas')`	3
`substring(`*string* `[from int] [for int])`	text	Extract substring	`substring('Thomas' from 2 for 3)`	hom
`substring(`*string* `from` *pattern*`)`	text	Extract substring matching POSIX regular expression. See Section 9.7 for more information on pattern matching.	`substring('Thomas' from '...$')`	mas
`substring(`*string* `from` *pattern* `for` *escape*`)`	text	Extract substring matching SQL regular expression. See Section 9.7 for more information on pattern matching.	`substring('Thomas' from '%#"o_a#"_' for '#')`	oma
`trim([leading | trailing | both] [`*characters*`] from` *string*`)`	text	Remove the longest string containing only characters from *characters* (a space by default) from the start, end, or both ends (`both` is the default) of *string*	`trim(both 'xyz' from 'yxTomxx')`	Tom
`trim([leading | trailing | both] [from]` *string* `[,` *characters*`] )`	text	Non-standard syntax for `trim()`	`trim(both from 'yxTomxx', 'xyz')`	Tom
`upper(`*string*`)`	text	Convert string to upper case	`upper('tom')`	TOM

Examples

```
SELECT FORMAT('Hello, %s','PostgreSQL');
```

Data Output | Explain | Messag

	format text
1	Hello, PostgreSQL

```
SELECT
FORMAT('%s, %s', last_name, first_name) full_name  FROM
passengers
ORDER BY full_name;
```

Data Output | Explain | Mes

	full_name text
1	Abethell, Adele
2	Adan, Lyndy
3	Addekin, Hermon
4	Aglione, Honor
5	Akam, Creight
6	Albone, Randy
7	Aldcorn, Bess
8	Alliston, Alexander

PostgreSQL Letter Case Functions

- PostgreSQL **LOWER** function
- PostgreSQL **UPPER** function
- PostgreSQL **INITCAP** function

Examples

```
--LOWER() FUNCTION
SELECT LOWER(last_name)
FROM
passengers  ORDER BY last_name;
```

Data Output | Expla

	lower text
1	abethell
2	adan
3	addekin
4	aglione

```
--UPPER() FUNCTION
SELECT UPPER(last_name)
FROM
passengers ORDER BY last_name;
```

Data Output | Explain

	upper text
1	ABETHELL
2	ADAN
3	ADDEKIN
4	AGLIONE

Examples

```
SELECT INITCAP('hello world') as example;
```

Data Output | Expl

	example text
1	Hello World

```
--INITCAP() FUNCTION
SELECT INITCAP(CONCAT (first_name, ' ', last_name))
FROM
passengers  ORDER BY first_name;
```

	initcap text
1	Abbie Gatland
2	Abner Philippet
3	Adamo Biggs
4	Adele Abethell
5	Ailene Nuzzi
6	Aldo Arnison

PostgreSQL LEFT Function

The PostgreSQL **LEFT**() function returns the first n acters in the string.

The following illustrates the syntax of the PostgreSQL **LEFT**() function:

```
LEFT(string, n)
```

Examples

```
SELECT LEFT('ABC',1);--A
```

```
SELECT LEFT('ABC',2);--AB
```

PostgreSQL RIGHT Function

The PostgreSQL **RIGHT**() function returns the last n characters in a string.

The following shows the syntax of the PostgreSQL **RIGHT**() function:

```
RIGHT(string, n)
```

Examples

```
SELECT RIGHT('XYZ', 2);--YZ

SELECT RIGHT('XYZ', 1);--Z
```

```
SELECT last_name  FROM passengers
WHERE RIGHT(last_name,3) = 'son';
```

Data Output | Explain | M

	last_name character varying (50)
1	Davison
2	Dotson
3	Imason
4	Arnison
5	Edison
6	Kristoffersson
7	Utterson

PostgreSQL TRIM Function

- The **LTRIM**() function removes all characters, spaces by default, from the beginning of a string.

- The **RTRIM**() function removes all characters, spaces by default, from the end of a string.

- The **BTRIM**() function is the combination of the **LTRIM**() and **RTRIM**() functions.

Examples

```
SELECT
LTRIM('enterprise', 'e');--nterprise

SELECT
RTRIM('enterprise', 'e');--enterpris

SELECT
BTRIM('enterprise', 'e');--nterpris
```

PostgreSQL POSITION Function

The PostgreSQL **POSITION**() function returns the location of a substring in a string.

The following illustrates the syntax of the PostgreSQL POSITION() function:

POSITION(substring in string)

Example

```
SELECT POSITION('example' IN 'PostgreSQL example');--12

SELECT POSITION('example' IN 'PostgreSQL Example');--0
```

PostgreSQL REPLACE Function

To search and replace all occurrences of a string with a new one, you use the **REPLACE**() function.

The following illustrates the syntax of the PostgreSQL **REPLACE**() function:

REPLACE(source, old_text, new_text);

Examples

```
SELECT
REPLACE ('ABC AA', 'A', 'Z');--ZBC ZZ
```

```
UPDATE
customer
SET
email = REPLACE (email,  'test.com',
'example.com')
```

PostgreSQL Substring Function

The **substring** function returns a part of string. The following illustrates the syntax of the **substring** function:

SUBSTRING (string ,start_position , length)

Examples

```
SELECT
SUBSTRING ('PostgreSQL', 1, 8); -- PostgreS

SELECT
SUBSTRING ('PostgreSQL', 8); -- SQL

SELECT
SUBSTRING ('PostgreSQL' FROM 1 FOR 8); -- PostgreS
```

Conditional expressions and operators

Conditional expressions

- Allow you to execute different logic based on conditions.
- The most used conditional expressions are **CASE**, **COALESCE**, **NULLIF**, and conditional operators like **GREATEST** and **LEAST**.

CASE

- The SQL CASE expression is a generic conditional expression, similar to if/else statements in other programming languages:

```
CASE WHEN condition THEN result
     [WHEN ...]
     [ELSE result]
END
```

CASE

```sql
SELECT
    booking_id,
    price,
    CASE
        WHEN price > 5000 THEN 'High'
        WHEN price BETWEEN 3000 AND 5000 THEN 'Medium'
        ELSE 'Low'
    END AS PriceLevel
FROM booking;
```

Data Output | Explain | Messages | Notifications

	booking_id [PK] integer	price numeric (7,2)	pricelevel text
1	1	7462.13	High
2	2	4216.60	Medium
3	3	5782.37	High
4	4	102.96	Low
5	5	6711.04	High
6	6	7813.55	High
7	7	1346.71	Low
8	8	5711.67	High
9	9	7015.82	High
10	10	1668.38	Low

COALESCE

- The COALESCE function returns the first of its arguments that is not null. Null is returned only if all arguments are null.

```
COALESCE(value [, ...])
```

```
SELECT COALESCE(description, short_description, '(none)') ...
```

Example

```
1  SELECT
2      passenger_id,
3      COALESCE(passport_number, 'No number') AS passport_of_passenger
4  FROM passengers order by passenger_id;
```

Data Output | Explain | Messages | Notifications

	passenger_id [PK] integer	passport_of_passenger character varying
1	1	176443701-2
2	2	229690188-3
3	3	595940366-8
4	4	604001503-9
5	5	No number
6	6	865501891-0
7	7	670764566-8
8	8	951717035-1
9	9	482708473-4

NULLIF

- The `NULLIF` function returns a null value if *`value1`* equals *`value2`*; otherwise it returns *`value1`*. This can be used to perform the inverse operation of the `COALESCE` example given above:

```
NULLIF(value1, value2)
```

```
SELECT NULLIF(value, '(none)') ...
```

Example

```
SELECT
    booking_id,
    NULLIF(price, 0) as price
FROM booking ORDER BY booking_id;
```

Output | Explain | Messages | Notifications

booking_id [PK] integer	price numeric (7,2)
1	8101.06
2	7947.11
3	[null]
4	9247.57
5	5765.23
6	1447.67
7	2624.79

GREATEST & LEAST

- The `GREATEST` and `LEAST` functions select the largest or smallest value from a list of any number of expressions.

```
GREATEST(value [, ...])

LEAST(value [, ...])
```

Example

```
SELECT
    booking_id,
    GREATEST(10000, price, 5000) AS Max,
    LEAST(10000, price, 5000) AS Min
FROM booking;
```

Output | Explain | Messages | Notifications

booking_id [PK] integer	max numeric	min numeric
4	10000	5000
5	10000	5000
6	10000	1447.67
7	10000	2624.79
8	10000	5000
9	10000	1204.56
10	10000	2365.17
11	10000	2236.74
12	10000	2753.61