PostgreSQL CHEAT SHEET: STRING FUNCTIONS

by **postgresql-backup.com** with ♥

CONVERSION

ASCII code of the first byte of the argument.

```
ASCII ('x') = 120
ASCII ('Ä') = 1234
```

Convert string to ASCII from another encoding (only supports conversion from LATIN1, LATIN2, LATIN9, and WIN1250 encodings)

```
TO_ASCII ('Karel') = 'Karel'
```

Character with the given code

```
CHR (65) = 'A'
CHR (1234) = 'Ä'
CHR (NULL) = NULL
```

Convert string to dest_encoding

```
CONVERT ('Text', 'UTF8', 'LATIN1') = 'Text'
```

Convert string to the database encoding / dest_encoding

```
CONVERT_FROM ('Text', 'UTF8') = 'Text'
CONVERT_TO ('Text', 'UTF8') = 'Text'
```

Encode / Decode binary data into/from textual representation in string

```
ENCODE (E'1234', 'base64') = 'MTIzNA=='
DECODE ('MTIzAAE=', 'base64') = 123
```

Convert the first letter of each word to upper case and the rest to lower case

```
INITCAP ('hi thomas') = 'Hi Thomas'
INITCAP ('all-in-all') = 'All-In-All'
INITCAP ('all_in_all') = 'All In All'
INITCAP ('go2bed') = 'Go2bed'
```

Convert string to lower / upper case

```
LOWER ('TOM') = 'tom'
UPPER ('tom') = 'TOM'
```

Calculates the MD5 hash of string, returning the result in hexadecimal

```
MD5('abc')='900150983cd24fb0d6963f7d28e17f72'
```

Current client encoding name

```
PG_CLIENT_ENCODING () = 'UTF8'
```

Return the given string suitably quoted to be used as an identifier in an SQL statement string. Quotes are added only if necessary. Embedded quotes are properly doubled

```
QUOTE_IDENT ('ABC DEF') = '"ABC DEF"'
QUOTE_IDENT ('"ABC"') = '"""ABC"""'
```

Return the given string suitably quoted to be used as a string literal in an SQL statement string. Embedded single-quotes and backslashes are properly doubled

```
QUOTE_LITERAL (E'O\'Reilly') = ''O''Reilly''
QUOTE_LITERAL (42.5) = "'42.5'"
QUOTE_LITERAL ('"ABC"') = ''''"ABC"''''
```

Return the given string suitably quoted to be used as a string literal in an SQL statement string; or, if the argument is null, return NULL

```
QUOTE_NULLABLE (NULL) = NULL
QUOTE_NULLABLE (42.5) = "'42.5'"
```

Convert number to its equivalent hexadecimal representation

```
TO_HEX (12) = 'c'
TO_HEX (42) = '2a'
TO_HEX (023150) = '5a6e'
TO_HEX (2147483647) = '7fffffff'
```

MEASUREMENT

Number of bits in string

```
BIT_LENGTH ('J') = 8
BIT_LENGTH ('Ö') = 16
BIT_LENGTH ('jose') = 32
BIT_LENGTH ('JÖSE') = 40
```

Number of characters in string

```
CHAR_LENGTH ('jose') = 4
CHARACTER_LENGTH ('jose') = 4
LENGTH ('jose') = 4
LENGTH ('JÖSE', 'UTF8') = 4
```

Number of bytes in string

```
OCTET_LENGTH ('AB') = 2
OCTET_LENGTH ('Ö') = 2
OCTET_LENGTH ('唅') = 4
```

MODIFICATION

String concatenation

```
'Postgre' || 'SQL' = 'PostgreSQL'
'Value: ' || 42 = 'Value: 42'
```

Remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string

```
BTRIM ('    AB    ') = 'AB'
TRIM ('    AB    ') = 'AB'
BTRIM ('=-AB-=', '-=') = 'AB'
TRIM (both '-=' from '=-AB-=') = 'AB'
LTRIM ('=-AB-=', '-=') = 'AB-='
TRIM (leading '-=' from '=-AB-=') = 'AB-='
RTRIM ('=-AB-=', '-=') = '=-AB'
TRIM (trailing '-=' from '=-AB-=') = '=-AB'
```

Fill up the string to length by prepending / appending the characters fill (a space by default)

```
LPAD ('ABC', 6) = '   ABC'
RPAD ('ABC', 6) = 'ABC   '
LPAD ('123', 6, '0') = '000123'
RPAD ('C', 3, '+') = 'C++'
```

Replace substring

```
OVERLAY('123' placing '-' from 2 for 3)='1-'
OVERLAY('123' placing '-' from 1 for 2)='-3'
OVERLAY('123' placing '-' from 2 for 1)='1-3'
```

Repeat string the specified number of times

```
REPEAT ('Pg', 4) = 'PgPgPgPg'
```

Any character in string that matches a character in the from set is replaced by the corresponding character in the to set

```
TRANSLATE ('12321', '12', 'ab') = 'ab3ba'
```

Replace substring(s) matching a POSIX regular expression

```
REGEXP_REPLACE('Pipe','p','-') = 'Pi-e'
REGEXP_REPLACE('Pipe','p','-','i') = '-ipe'
REGEXP_REPLACE('Pipe','p','-','g') = 'Pi-e'
REGEXP_REPLACE('Pipe','p','-','gi') = '-i-e'
REGEXP_REPLACE('24 h','\d+','00') = '00 h'
REGEXP_REPLACE('24 h','\s','_') = '24_h'
REGEXP_REPLACE('24 h','[\sh]+','?') = '24?'
REGEXP_REPLACE('\abc','\\.+','*') = '*'
```

Split string on delimiter and return the given field (counting from one)

```
SPLIT_PART ('1,2,3', ',', 2) = '2'
```

Extract substring

```
SUBSTRING ('12345' from 2 for 3) = '234'
SUBSTR ('12345', 3, 2) = '34'
```

Extract substring matching POSIX regular expression

```
SUBSTRING ('Regex' from '.{3}$') = 'gex'
```

Extract substring matching SQL regular expression

```
SUBSTRING('ABCDE'from'%#"B_D#"_'for'#')='BCD'
SUBSTRING('ABCDE'from'%#"B-D#"_'for'#')=NULL
```

Replace all occurrences of one substring with other substring

```
REPLACE ('A-B-C', '-','+') = 'A+B+C'
```

SEARCH & MATCHING

Location of specified substring

```
POSITION ('om' in 'Thomas') = 3
STRPOS ('Thomas', 'om') = 3
```

Return all captured substrings resulting from matching a POSIX regular expression against the string

```
REGEXP_MATCHES('1,2','(\d),(\d)')={'1','2'}
REGEXP_MATCHES('1,2','\d','g')={'1'},{'2'}
```

Split string using a POSIX regular expression as the delimiter

```
REGEXP_SPLIT_TO_ARRAY ('ABC DEF', E'\\s+') =
{ABC,DEF}
REGEXP_SPLIT_TO_TABLE ('ABC DEF', E'\\s+') =
'ABC' 'DEF' (2 rows)
```

Return true if the string matches the supplied pattern

```
'ABC' LIKE '_B_' = true
'ABC' NOT LIKE '_B_' = false
'ABC' SIMILAR TO '[ABC]+' = true
'ABC' NOT SIMILAR TO '[ABC]+' = false
```