

CONVERSION

ASCII code of the first byte of the argument.
<code>ASCII ('x') = 120</code> <code>ASCII ('Ä') = 1234</code>
Convert string to ASCII from another encoding (only supports conversion from LATIN1, LATIN2, LATIN9, and WIN1250 encodings)
<code>TO_ASCII ('Karel') = 'Karel'</code>
Character with the given code
<code>CHR (65) = 'A'</code> <code>CHR (1234) = 'Ä'</code> <code>CHR (NULL) = NULL</code>
Convert string to dest_encoding
<code>CONVERT ('Text', 'UTF8', 'LATIN1') = 'Text'</code>
Convert string to the database encoding / dest_encoding
<code>CONVERT_FROM ('Text', 'UTF8') = 'Text'</code> <code>CONVERT_TO ('Text', 'UTF8') = 'Text'</code>
Encode / Decode binary data into/from textual representation in string
<code>ENCODE (E'1234', 'base64') = 'MTIzNA=='</code> <code>DECODE ('MTIzAAE=', 'base64') = 123</code>
Convert the first letter of each word to upper case and the rest to lower case
<code>INITCAP ('hi thomas') = 'Hi Thomas'</code> <code>INITCAP ('all-in-all') = 'All-In-All'</code> <code>INITCAP ('all_in_all') = 'All In All'</code> <code>INITCAP ('go2bed') = 'Go2bed'</code>
Convert string to lower / upper case
<code>LOWER ('TOM') = 'tom'</code> <code>UPPER ('tom') = 'TOM'</code>
Calculates the MD5 hash of string, returning the result in hexadecimal
<code>MD5 ('abc') = '900150983cd24fb0d6963f7d28e17f72'</code>
Current client encoding name
<code>PG_CLIENT_ENCODING () = 'UTF8'</code>

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
<code>QUOTE_IDENT ('ABC DEF') = '"ABC DEF"'</code> <code>QUOTE_IDENT ('"ABC"') = '"\"ABC\""'</code>
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
<code>QUOTE_LITERAL (E'0\Reilly') = ''0\Reilly''</code> <code>QUOTE_LITERAL (42.5) = '"42.5"'</code> <code>QUOTE_LITERAL ('"ABC"') = '"\"ABC\""'</code>
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
<code>QUOTE_NULLABLE (NULL) = NULL</code> <code>QUOTE_NULLABLE (42.5) = '"42.5"'</code>
Convert number to its equivalent hexadecimal representation
<code>TO_HEX (12) = 'c'</code> <code>TO_HEX (42) = '2a'</code> <code>TO_HEX (023150) = '5a6e'</code> <code>TO_HEX (2147483647) = '7fffffff'</code>

MEASUREMENT

Number of bits in string
<code>BIT_LENGTH ('J') = 8</code> <code>BIT_LENGTH ('ö') = 16</code> <code>BIT_LENGTH ('jose') = 32</code> <code>BIT_LENGTH ('JÖSE') = 40</code>
Number of characters in string
<code>CHAR_LENGTH ('jose') = 4</code> <code>CHARACTER_LENGTH ('jose') = 4</code> <code>LENGTH ('jose') = 4</code> <code>LENGTH ('JÖSE', 'UTF8') = 4</code>
Number of bytes in string
<code>OCTET_LENGTH ('AB') = 2</code> <code>OCTET_LENGTH ('ö') = 2</code> <code>OCTET_LENGTH ('曉') = 4</code>

MODIFICATION

String concatenation
<code>'Postgre' 'SQL' = 'PostgreSQL'</code> <code>'Value: ' 42 = 'Value: 42'</code>
Remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string
<code>BTRIM (' AB ') = 'AB'</code> <code>TRIM (' AB ') = 'AB'</code> <code>BTRIM ('=-AB-=', '-=') = 'AB'</code> <code>TRIM (both '-=' from '=AB-') = 'AB'</code> <code>LTRIM ('=-AB-=', '-=') = 'AB-'</code> <code>TRIM (leading '-=' from '=AB-') = 'AB-'</code> <code>RTRIM ('=-AB-=', '-=') = '=AB'</code> <code>TRIM (trailing '-=' from '=AB-') = '=AB'</code>
Fill up the string to length by prepending / appending the characters fill (a space by default)
<code>LPAD ('ABC', 6) = ' ABC'</code> <code>RPAD ('ABC', 6) = 'ABC '</code> <code>LPAD ('123', 6, '0') = '000123'</code> <code>RPAD ('C', 3, '+') = 'C++'</code>
Replace substring
<code>OVERLAY('123' placing '-' from 2 for 3)='1-'</code> <code>OVERLAY('123' placing '-' from 1 for 2)='-3'</code> <code>OVERLAY('123' placing '-' from 2 for 1)='1-3'</code>
Repeat string the specified number of times
<code>REPEAT ('Pg', 4) = 'PgPgPgPg'</code>
Any character in string that matches a character in the from set is replaced by the corresponding character in the to set
<code>TRANSLATE ('12321', '12', 'ab') = 'ab3ba'</code>
Replace substring(s) matching a POSIX regular expression
<code>REGEXP_REPLACE('Pipe','p','-') = 'Pi-e'</code> <code>REGEXP_REPLACE('Pipe','p','-','i') = '-ipe'</code> <code>REGEXP_REPLACE('Pipe','p','-','g') = 'Pi-e'</code> <code>REGEXP_REPLACE('Pipe','p','-','gi') = '-i-e'</code> <code>REGEXP_REPLACE('24 h','\d+','00') = '00 h'</code> <code>REGEXP_REPLACE('24 h','\s','_') = '24_h'</code> <code>REGEXP_REPLACE('24 h','[sh]+','?') = '24?'</code> <code>REGEXP_REPLACE('\abc','\.+','*') = '*'</code>

Split string on delimiter and return the given field (counting from one)
<code>SPLIT_PART ('1,2,3', ',', 2) = '2'</code>
Extract substring
<code>SUBSTRING ('12345' from 2 for 3) = '234'</code> <code>SUBSTR ('12345', 3, 2) = '34'</code>
Extract substring matching POSIX regular expression
<code>SUBSTRING ('Regex' from '.{3}\$') = 'gex'</code>
Extract substring matching SQL regular expression
<code>SUBSTRING('ABCDE' from '%#B_D#' for '#') = 'BCD'</code> <code>SUBSTRING('ABCDE' from '%#B-D#' for '#') = NULL</code>
Replace all occurrences of one substring with other substring
<code>REPLACE ('A-B-C', '-', '+') = 'A+B+C'</code>

SEARCH & MATCHING

Location of specified substring
<code>POSITION ('om' in 'Thomas') = 3</code> <code>STRPOS ('Thomas', 'om') = 3</code>
Return all captured substrings resulting from matching a POSIX regular expression against the string
<code>REGEXP_MATCHES('1,2','(\d),(\d)')={ '1', '2' }</code> <code>REGEXP_MATCHES('1,2','\d','g')={ '1' }, { '2' }</code>
Split string using a POSIX regular expression as the delimiter
<code>REGEXP_SPLIT_TO_ARRAY ('ABC DEF', E'\\s+') = {ABC,DEF}</code> <code>REGEXP_SPLIT_TO_TABLE ('ABC DEF', E'\\s+') = 'ABC' 'DEF' (2 rows)</code>
Return true if the string matches the supplied pattern
<code>'ABC' LIKE '_B_' = true</code> <code>'ABC' NOT LIKE '_B_' = false</code> <code>'ABC' SIMILAR TO '[ABC]+' = true</code> <code>'ABC' NOT SIMILAR TO '[ABC]+' = false</code>