Built-in Functions 内置功能

!

! expr - Logical not.

! expr-逻辑不。

Examples:

例子:

```
> SELECT ! true;
 false
> SELECT ! false;
 true
> SELECT ! NULL;
 NULL

```

Since: 1.0.0

自: 1.0.0

!=

expr1 != expr2 - Returns true if `expr1` is not equal to `expr2`, or false otherwise.

Expr1! = expr2-如果 expr1不等于 expr2，返回 true，否则返回 false。

Arguments:

论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be used in equality comparison. Map type is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是可以在相等比较中使用的类型。不支持映射类型。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 1 != 2;
 true
> SELECT 1 != '2';
 true
> SELECT true != NULL;
```

```
 NULL
> SELECT NULL != NULL;
 NULL

```

Since: 1.0.0
自: 1.0.0

%

expr1 % expr2 - Returns the remainder after `expr1`/`expr2`.
返回 expr1/expr2之后的余数。
Examples:
例子:

```
> SELECT 2 % 1.8;
 0.2
> SELECT MOD(2, 1.8);
 0.2

```

Since: 1.0.0
自: 1.0.0

&

expr1 & expr2 - Returns the result of bitwise AND of `expr1` and `expr2`.
返回 expr1和 expr2的按位 AND 结果。
Examples:
例子:

```
> SELECT 3 & 5;
 1

```

Since: 1.4.0
自: 1.4.0

*

expr1 * expr2 - Returns `expr1`*`expr2`.

Expr1 * expr2-Returns expr1 * expr2.

Examples:

例子:

```
> SELECT 2 * 3;
 6

```

Since: 1.0.0

自: 1.0.0

+

expr1 + expr2 - Returns `expr1`+`expr2`.

Expr1 + expr2-Returns expr1 + expr2.

Examples:

例子:

```
> SELECT 1 + 2;
 3

```

Since: 1.0.0

自: 1.0.0

-

expr1 - expr2 - Returns `expr1`-`expr2`.

Expr1-expr2-Returns expr1-expr2.

Examples:

例子:

```
> SELECT 2 - 1;
 1

```

Since: 1.0.0

自: 1.0.0

/

expr1 / expr2 - Returns `expr1`/`expr2`. It always performs floating point division.

Expr1/expr2-Returns expr1/expr2。它总是执行浮点除法。

Examples:

例子:

```
> SELECT 3 / 2;
 1.5
> SELECT 2L / 2L;
 1.0
```

Since: 1.0.0

自: 1.0.0

<

expr1 < expr2 - Returns true if `expr1` is less than `expr2`.

Expr1 < expr2-如果 expr1小于 expr2，返回 true。

Arguments:

论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be ordered. For example, map type is not orderable, so it is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是一个可以排序的类型。例如，map 类型不可排序，因此不支持它。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 1 < 2;
 true
> SELECT 1.1 < '1';
 false
> SELECT to_date('2009-07-30 04:17:52') < to_date('2009-07-30 04:17:52');
 false
> SELECT to_date('2009-07-30 04:17:52') < to_date('2009-08-01 04:17:52');
 true
```

```
> SELECT 1 < NULL;
 NULL
```

Since: 1.0.0

自: 1.0.0

<=

expr1 <= expr2 - Returns true if `expr1` is less than or equal to `expr2`.

Expr1 < = expr2-如果 expr1小于或等于 expr2，返回 true。

Arguments:

论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be ordered. For example, map type is not orderable, so it is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是一个可以排序的类型。例如，map 类型不可排序，因此不支持它。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 2 <= 2;
 true
> SELECT 1.0 <= '1';
 true
> SELECT to_date('2009-07-30 04:17:52') <= to_date('2009-07-30 04:17:52');
 true
> SELECT to_date('2009-07-30 04:17:52') <= to_date('2009-08-01 04:17:52');
 true
> SELECT 1 <= NULL;
 NULL
```

Since: 1.0.0

自: 1.0.0

<=>

expr1 <=> expr2 - Returns same result as the EQUAL(=) operator for non-null operands, but returns true if both are null, false if one of the them is null.
Expr1 < = > expr2-返回与非空操作数的 EQUAL (=)操作符相同的结果，但是如果两个操作数都为 null，返回 true，如果其中一个为 null，返回 false。

Arguments:

论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be used in equality comparison. Map type is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是可以在相等比较中使用的类型。不支持映射类型。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 2 <=> 2;
 true
> SELECT 1 <=> '1';
 true
> SELECT true <=> NULL;
 false
> SELECT NULL <=> NULL;
 true

```

Since: 1.1.0

自: 1.1.0

< >

expr1 != expr2 - Returns true if `expr1` is not equal to `expr2`, or false otherwise.
Expr1! = expr2-如果 expr1不等于 expr2，返回 true，否则返回 false。

Arguments:

论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be used in equality comparison. Map type is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是可以在相等比较中使用的类型。不支持映射类型。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 1 != 2;
 true
> SELECT 1 != '2';
 true
> SELECT true != NULL;
 NULL
> SELECT NULL != NULL;
 NULL

```

Since: 1.0.0

自: 1.0.0

=

expr1 = expr2 - Returns true if `expr1` equals `expr2`, or false otherwise.

Expr1 = expr2-如果 expr1等于 expr2，返回 true，否则返回 false。

Arguments:

论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be used in equality comparison. Map type is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是可以在相等比较中使用的类型。不支持映射类型。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 2 = 2;
 true
> SELECT 1 = '1';
 true
> SELECT true = NULL;
 NULL
> SELECT NULL = NULL;
 NULL

```

Since: 1.0.0
自: 1.0.0

==

expr1 == expr2 - Returns true if `expr1` equals `expr2`, or false otherwise.
Expr1 = = expr2-如果 expr1 = expr2返回 true，否则返回 false。
Arguments:
论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be used in equality comparison. Map type is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是可以在相等比较中使用的类型。不支持映射类型。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:
例子:

```
> SELECT 2 == 2;
 true
> SELECT 1 == '1';
 true
> SELECT true == NULL;
 NULL
> SELECT NULL == NULL;
 NULL

```

Since: 1.0.0
自: 1.0.0

>

expr1 > expr2 - Returns true if `expr1` is greater than `expr2`.
如果 expr1大于 expr2，返回 true。
Arguments:
论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be ordered. For example, map type is not orderable, so it is not supported. For complex types such

array/struct, the data types of fields must be orderable.

- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是一个可以排序的类型。例如，map 类型不可排序，因此不支持它。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 2 > 1;
 true
> SELECT 2 > '1.1';
 true
> SELECT to_date('2009-07-30 04:17:52') > to_date('2009-07-30 04:17:52');
 false
> SELECT to_date('2009-07-30 04:17:52') > to_date('2009-08-01 04:17:52');
 false
> SELECT 1 > NULL;
 NULL

```

Since: 1.0.0

自: 1.0.0

>=

expr1 >= expr2 - Returns true if `expr1` is greater than or equal to `expr2`.

Expr1 > = expr2-如果 expr1大于或等于 expr2，返回 true。

Arguments:

论点:

- expr1, expr2 - the two expressions must be same type or can be casted to a common type, and must be a type that can be ordered. For example, map type is not orderable, so it is not supported. For complex types such array/struct, the data types of fields must be orderable.
- Expr1，expr2-这两个表达式必须是相同的类型，或者可以铸造成一个公共类型，并且必须是一个可以排序的类型。例如，map 类型不可排序，因此不支持它。对于数组/结构这样的复杂类型，字段的数据类型必须是可排序的。

Examples:

例子:

```
> SELECT 2 >= 1;
 true
```

```
> SELECT 2.0 >= '2.1';
 false
> SELECT to_date('2009-07-30 04:17:52') >= to_date('2009-07-30 04:17:52');
 true
> SELECT to_date('2009-07-30 04:17:52') >= to_date('2009-08-01 04:17:52');
 false
> SELECT 1 >= NULL;
 NULL

```

Since: 1.0.0
自: 1.0.0

^

expr1 ^ expr2 - Returns the result of bitwise exclusive OR of `expr1` and `expr2`.
Expr1 ^ expr2-返回 expr1和 expr2的按位异或结果。
Examples:
例子:

```
> SELECT 3 ^ 5;
 6

```

Since: 1.4.0
自: 1.4.0

abs 腹肌

abs(expr) - Returns the absolute value of the numeric value.
返回数值的绝对值。
Examples:
例子:

```
> SELECT abs(-1);
 1

```

Since: 1.2.0
自: 1.2.0

acos

acos(expr) - Returns the inverse cosine (a.k.a. arc cosine) of `expr`, as if computed by `java.lang.Math.acos`.

Acos (expr)-返回 expr 的反余弦(又名 arc cosine)，就像是由 java.lang.Math.acos 计算一样。

Examples:

例子:

```
> SELECT acos(1);
 0.0
> SELECT acos(2);
 NaN

```

Since: 1.4.0

自: 1.4.0

acosh 一点点

acosh(expr) - Returns inverse hyperbolic cosine of `expr`.

返回 expr 的反双曲余弦。

Examples:

例子:

```
> SELECT acosh(1);
 0.0
> SELECT acosh(0);
 NaN

```

Since: 3.0.0

自: 3.0.0

add_months 再加个月

add_months(start_date, num_months) - Returns the date that is `num_months` after `start_date`.

Add _ months (start _ date，num _ months)-返回在 start _ date 之后的 num _ months 日期。

Examples:

例子:

```
> SELECT add_months('2016-08-31', 1);
 2016-09-30

```

Since: 1.5.0
自: 1.5.0

aggregate 集合

aggregate(expr, start, merge, finish) - Applies a binary operator to an initial state and all elements in the array, and reduces this to a single state. The final state is converted into the final result by applying a finish function.
Aggregate (expr、 start、 merge、 finish)——将二进制运算符应用于初始状态和数组中的所有元素，并将其减少为单个状态。最终状态通过应用 finish 函数转换为最终结果。
Examples:
例子:

```
> SELECT aggregate(array(1, 2, 3), 0, (acc, x) -> acc + x);
 6
> SELECT aggregate(array(1, 2, 3), 0, (acc, x) -> acc + x, acc -> acc * 10);
 60

```

Since: 2.4.0
自: 2.4.0

and 及

expr1 and expr2 - Logical AND.
Expr1和 expr2-逻辑与。
Examples:
例子:

```
> SELECT true and true;
 true
> SELECT true and false;
 false
> SELECT true and NULL;
```

```
 NULL
> SELECT false and NULL;
 false

```

Since: 1.0.0

自: 1.0.0

any 任何

any(expr) - Returns true if at least one value of `expr` is true.

如果至少有一个 expr 值为真，则返回 true。

Examples:

例子:

```
> SELECT any(col) FROM VALUES (true), (false), (false) AS tab(col);
 true
> SELECT any(col) FROM VALUES (NULL), (true), (false) AS tab(col);
 true
> SELECT any(col) FROM VALUES (false), (false), (NULL) AS tab(col);
 false

```

Since: 3.0.0

自: 3.0.0

approx_count_distinct 大约计数不同

approx_count_distinct(expr[, relativeSD]) - Returns the estimated cardinality by HyperLogLog++. `relativeSD` defines the maximum relative standard deviation allowed.

Approx _ count _ distinct (expr [， relativeSD])-返回 HyperLogLog + + 的估计基数，relativeSD 定义允许的最大相对标准差。

Examples:

例子:

```
> SELECT approx_count_distinct(col1) FROM VALUES (1), (1), (2), (2), (3) tab(col1);
 3

```

Since: 1.6.0

自: 1.6.0

approx_percentile 大约百分比

approx_percentile(col, percentage [, accuracy]) - Returns the approximate `percentile` of the numeric column `col` which is the smallest value in the ordered `col` values (sorted from least to greatest) such that no more than `percentage` of `col` values is less than the value or equal to that value. The value of percentage must be between 0.0 and 1.0. The `accuracy` parameter (default: 10000) is a positive numeric literal which controls approximation accuracy at the cost of memory. Higher value of `accuracy` yields better accuracy, `1.0/accuracy` is the relative error of the approximation. When `percentage` is an array, each value of the percentage array must be between 0.0 and 1.0. In this case, returns the approximate percentile array of column `col` at the given percentage array.

Approx _ percentile (col，percentage [，accuracy])——返回数字列 col 的近似百分位数，它是有序 col 值中的最小值(从最小到最大排序) ，以便 col 值的百分比不小于该值或等于该值。百分比值必须介于0.0和1.0之间。精度参数(默认值: 10000)是一个正数值，它以内存为代价来控制近似精度。较高的精度值产生较好的精度，1.0/精度是近似值的相对误差。当 percentage 为数组时，百分比数组的每个值必须介于0.0和1.0之间。在本例中，返回给定百分比数组中列 col 的近似百分位数组。

Examples:

例子:

```
> SELECT approx_percentile(col, array(0.5, 0.4, 0.1), 100) FROM VALUES (0), (1), (2), (10) AS tab(col);
 [1,1,0]
> SELECT approx_percentile(col, 0.5, 100) FROM VALUES (0), (6), (7), (9), (10) AS tab(col);
 7

```

Since: 2.1.0

自: 2.1.0

array 数组

array(expr, ...) - Returns an array with the given elements.

Array (expr，...)-返回具有给定元素的数组。

Examples:

例子:

```
> SELECT array(1, 2, 3);
 [1,2,3]

```

Since: 1.1.0
自: 1.1.0

array_contains 包含

array_contains(array, value) - Returns true if the array contains the value.
Array _ contains (array，value)-如果数组包含值，则返回 true。
Examples:
例子:

```
> SELECT array_contains(array(1, 2, 3), 2);
 true

```

Since: 1.5.0
自: 1.5.0

array_distinct 数组独立性

array_distinct(array) - Removes duplicate values from the array.
Array _ distinct (array)-从数组中移除重复的值。
Examples:
例子:

```
> SELECT array_distinct(array(1, 2, 3, null, 3));
 [1,2,3,null]

```

Since: 2.4.0
自: 2.4.0

array_except 数组除外

array_except(array1, array2) - Returns an array of the elements in array1 but not in array2, without duplicates.

Array _ except (array1，array2)——返回数组1中的元素数组，但不返回数组2中的元素数组，不带重复项。

Examples:

例子:

```
> SELECT array_except(array(1, 2, 3), array(1, 3, 5));
 [2]

```

Since: 2.4.0

自: 2.4.0

array_intersect 数组 _ intersect

array_intersect(array1, array2) - Returns an array of the elements in the intersection of array1 and array2, without duplicates.

Array _ intersect (array1，array2)——返回数组1和数组2交叉处的元素数组，没有重复项。

Examples:

例子:

```
> SELECT array_intersect(array(1, 2, 3), array(1, 3, 5));
 [1,3]

```

Since: 2.4.0

自: 2.4.0

array_join 数组 _ join

array_join(array, delimiter[, nullReplacement]) - Concatenates the elements of the given array using the delimiter and an optional string to replace nulls. If no value is set for nullReplacement, any null value is filtered.

Array _ join (array，delimiter [，nullReplacement])-使用分隔符和可选字符串串联给定数组的元素以替换空值。如果没有为 nullReplacement 设置值，则会筛选任何空值。

Examples:

例子:

```
> SELECT array_join(array('hello', 'world'), ' ');
 hello world
```

```
> SELECT array_join(array('hello', null ,'world'), ' ');
 hello world
> SELECT array_join(array('hello', null ,'world'), ' ', ',');
 hello , world
```

Since: 2.4.0

自: 2.4.0

array_max 最大阵列

array_max(array) - Returns the maximum value in the array. NaN is greater than any non-NaN elements for double/float type. NULL elements are skipped.

Array _ max (array)-返回数组中的最大值。对于 double/float 类型，NaN 大于任何非 NaN 元素。跳过 NULL 元素。

Examples:

例子:

```
> SELECT array_max(array(1, 20, null, 3));
 20
```

Since: 2.4.0

自: 2.4.0

array_min 数组 _ min

array_min(array) - Returns the minimum value in the array. NaN is greater than any non-NaN elements for double/float type. NULL elements are skipped.

Array _ min (array)-返回数组中的最小值。对于 double/float 类型，NaN 大于任何非 NaN 元素。跳过 NULL 元素。

Examples:

例子:

```
> SELECT array_min(array(1, 20, null, 3));
 1
```

Since: 2.4.0

自: 2.4.0

array_position 阵列 _ 位置

array_position(array, element) - Returns the (1-based) index of the first element of the array as long.

Array _ position (array，element)-返回数组的第一个元素的(基于1的)索引为 long。

Examples:

例子:

```
> SELECT array_position(array(3, 2, 1), 1);
 3

```

Since: 2.4.0

自: 2.4.0

array_remove 数组移除

array_remove(array, element) - Remove all elements that equal to element from array.

Array _ Remove (array，element)-从 array 中删除所有等于 element 的元素。

Examples:

例子:

```
> SELECT array_remove(array(1, 2, 3, null, 3), 3);
 [1,2,null]

```

Since: 2.4.0

自: 2.4.0

array_repeat 数组重复

array_repeat(element, count) - Returns the array containing element count times.

Array _ repeat (element，count)-返回包含元素计数次数的数组。

Examples:

例子:

```
> SELECT array_repeat('123', 2);
 ["123","123"]
```

```
3
```

Since: 2.4.0
自: 2.4.0

array_sort 数组排序

array_sort(expr, func) - Sorts the input array. If func is omitted, sort in ascending order. The elements of the input array must be orderable. NaN is greater than any non-NaN elements for double/float type. Null elements will be placed at the end of the returned array. Since 3.0.0 this function also sorts and returns the array based on the given comparator function. The comparator will take two arguments representing two elements of the array. It returns -1, 0, or 1 as the first element is less than, equal to, or greater than the second element. If the comparator function returns other values (including null), the function will fail and raise an error.
Array _ sort (expr，func)-对输入数组排序。如果省略了 func，则按升序排序。输入数组的元素必须是可排序的。对于 double/float 类型，NaN 大于任何非 NaN 元素。空元素将放置在返回的数组的末尾。由于3.0.0，这个函数也根据给定的比较器函数对数组进行排序和返回。比较器将采用两个参数，表示数组的两个元素。它返回-1、0或1，因为第一个元素小于、等于或大于第二个元素。如果 comparator 函数返回其他值(包括 null)，则该函数将失败并引发错误。

Examples:
例子:

```
1 > SELECT array_sort(array(5, 6, 1), (left, right) -> case when left < right then -1 when
   left > right then 1 else 0 end);
2  [1,5,6]
3 > SELECT array_sort(array('bc', 'ab', 'dc'), (left, right) -> case when left is null and
   right is null then 0 when left is null then -1 when right is null then 1 when left <
  right then 1 when left > right then -1 else 0 end);
4  ["dc","bc","ab"]
5 > SELECT array_sort(array('b', 'd', null, 'c', 'a'));
6  ["a","b","c","d",null]
7
```

Since: 2.4.0
自: 2.4.0

array_union 阵列联合

array_union(array1, array2) - Returns an array of the elements in the union of array1 and array2, without duplicates.
Array _ union (array1，array2)——返回数组1和数组2的联合中的元素数组，不带重复项。
Examples:
例子:

```
> SELECT array_union(array(1, 2, 3), array(1, 3, 5));
 [1,2,3,5]

```

Since: 2.4.0
自: 2.4.0

arrays_overlap 数组重叠

arrays_overlap(a1, a2) - Returns true if a1 contains at least a non-null element present also in a2. If the arrays have no common element and they are both non-empty and either of them contains a null element null is returned, false otherwise.
数组 _ overlap (a1，a2)-如果 a1至少包含 a2中的非空元素，则返回 true。如果数组没有公共元素，并且它们都是非空的，且其中任何一个都包含 null 元素，则返回 false。
Examples:
例子:

```
> SELECT arrays_overlap(array(1, 2, 3), array(3, 4, 5));
 true

```

Since: 2.4.0
自: 2.4.0

arrays_zip 数组 zip

arrays_zip(a1, a2, ...) - Returns a merged array of structs in which the N-th struct contains all N-th values of input arrays.
数组 _ zip (a1，a2，...)-返回合并的结构数组，其中 n 次结构包含输入数组的所有 n 次值。
Examples:
例子:

```
> SELECT arrays_zip(array(1, 2, 3), array(2, 3, 4));
```

```
 [{"0":1,"1":2},{"0":2,"1":3},{"0":3,"1":4}]
> SELECT arrays_zip(array(1, 2), array(2, 3), array(3, 4));
 [{"0":1,"1":2,"2":3},{"0":2,"1":3,"2":4}]

```

Since: 2.4.0

自: 2.4.0

ascii 美国信息交换码

ascii(str) - Returns the numeric value of the first character of `str`.

Ascii (str)-返回 str 的第一个字符的数值。

Examples:

例子:

```
> SELECT ascii('222');
 50
> SELECT ascii(2);
 50

```

Since: 1.5.0

自: 1.5.0

asin 阿辛

asin(expr) - Returns the inverse sine (a.k.a. arc sine) the arc sin of `expr`, as if computed by `java.lang.Math.asin`.

Asin (expr)-返回 expr 的弧 sin 的逆正弦(也称为弧正弦)，如同由 java.lang.Math.asin 计算一样。

Examples:

例子:

```
> SELECT asin(0);
 0.0
> SELECT asin(2);
 NaN

```

Since: 1.4.0

自: 1.4.0

asinh 阿辛

asinh(expr) - Returns inverse hyperbolic sine of `expr`.

返回 expr 的反双曲正弦值。

Examples:

例子:

```
> SELECT asinh(0);
 0.0

```

Since: 3.0.0

自: 3.0.0

assert_true 断言真实

assert_true(expr) - Throws an exception if `expr` is not true.

Assert _ true (expr)-如果 expr 不为真，则抛出异常。

Examples:

例子:

```
> SELECT assert_true(0 < 1);
 NULL

```

Since: 2.0.0

自: 2.0.0

atan

atan(expr) - Returns the inverse tangent (a.k.a. arc tangent) of `expr`, as if computed by `java.lang.Math.atan`

返回 expr 的反正切(又名反正切)，就像是由 java.lang.Math.atan 计算一样

Examples:

例子:

```
> SELECT atan(0);
 0.0
```

```
3
```

Since: 1.4.0

自: 1.4.0

atan2

atan2(exprY, exprX) - Returns the angle in radians between the positive x-axis of a plane and the point given by the coordinates (`exprX`, `exprY`), as if computed by `java.lang.Math.atan2`.

Atan2(ex撬，exprX)——返回平面的正 x 轴与坐标(exprX，ex撬)给出的点之间的弧度角，就像用 java.lang 计算一样。2.

Arguments:

论点:

- exprY - coordinate on y-axis Y 轴上的 ex撬坐标
- exprX - coordinate on x-axis X 轴坐标

Examples:

例子:

```
> SELECT atan2(0, 0);
 0.0

```

Since: 1.4.0

自: 1.4.0

atanh 阿坦

atanh(expr) - Returns inverse hyperbolic tangent of `expr`.

返回 expr 的反双曲正切。

Examples:

例子:

```
> SELECT atanh(0);
 0.0
> SELECT atanh(2);
 NaN

```

Since: 3.0.0

自: 3.0.0

avg 平均

avg(expr) - Returns the mean calculated from values of a group.

Avg (expr)-返回从组的值计算出的平均值。

Examples:

例子:

```
> SELECT avg(col) FROM VALUES (1), (2), (3) AS tab(col);
 2.0
> SELECT avg(col) FROM VALUES (1), (2), (NULL) AS tab(col);
 1.5

```

Since: 1.0.0

自: 1.0.0

base64

base64(bin) - Converts the argument from a binary `bin` to a base 64 string.

Base64(bin)-将参数从一个二进制 bin 转换为一个基64字符串。

Examples:

例子:

```
> SELECT base64('Spark SQL');
 U3BhcmsgU1FM

```

Since: 1.5.0

自: 1.5.0

between 中间

expr1 [NOT] BETWEEN expr2 AND expr3 - evaluate if `expr1` is [not] in between `expr2` and `expr3`.

Expr1[NOT] BETWEEN expr2 AND expr3-evaluate if expr1 is [NOT] in BETWEEN expr2 AND expr3.

Examples:

例子:

```
> SELECT col1 FROM VALUES 1, 3, 5, 7 WHERE col1 BETWEEN 2 AND 5;
 3
 5

```

Since: 1.0.0

自: 1.0.0

bigint 双子星

bigint(expr) - Casts the value `expr` to the target data type `bigint`.

Bigint (expr)-将值 expr 强制转换为目标数据类型 bigint。

Since: 2.0.1

自: 2.0.1

bin 垃圾桶

bin(expr) - Returns the string representation of the long value `expr` represented in binary.

Bin (expr)-返回用二进制表示的长值 expr 的字符串表示形式。

Examples:

例子:

```
> SELECT bin(13);
 1101
> SELECT bin(-13);
 1111111111111111111111111111111111111111111111111111111111110011
> SELECT bin(13.3);
 1101

```

Since: 1.5.0

自: 1.5.0

binary 二进制

binary(expr) - Casts the value `expr` to the target data type `binary`.

Binary (expr)-将值 expr 强制转换为目标数据类型 binary。

Since: 2.0.1

自: 2.0.1

bit_and 比特和

bit_and(expr) - Returns the bitwise AND of all non-null input values, or null if none.
返回所有非 null 输入值的按位 AND，如果没有，返回 null。

Examples:
例子:

```
> SELECT bit_and(col) FROM VALUES (3), (5) AS tab(col);
 1

```

Since: 3.0.0
自: 3.0.0

bit_count 位计数

bit_count(expr) - Returns the number of bits that are set in the argument expr as an unsigned 64-bit integer, or NULL if the argument is NULL.
Bit _ count (expr)-返回参数 expr 中设置为无符号64位整数的位数，如果参数为 NULL，则返回 NULL。

Examples:
例子:

```
> SELECT bit_count(0);
 0

```

Since: 3.0.0
自: 3.0.0

bit_get 比特获得

bit_get(expr, pos) - Returns the value of the bit (0 or 1) at the specified position. The positions are numbered from right to left, starting at zero. The position argument cannot be negative.
Bit _ get (expr，pos)-返回指定位置的位(0或1)值。位置从右到左编号，从零开始。位置参数不能是否定的。

Examples:
例子:

```
> SELECT bit_get(11, 0);
 1
> SELECT bit_get(11, 2);
 0

```

Since: 3.2.0
3.2.0

bit_length 位长度

bit_length(expr) - Returns the bit length of string data or number of bits of binary data.
返回字符串数据的位长度或二进制数据的位数。
Examples:
例子:

```
> SELECT bit_length('Spark SQL');
 72

```

Since: 2.3.0
自: 2.3.0

bit_or 比特或

bit_or(expr) - Returns the bitwise OR of all non-null input values, or null if none.
返回所有非 null 输入值的按位 OR，如果没有，返回 null。
Examples:
例子:

```
> SELECT bit_or(col) FROM VALUES (3), (5) AS tab(col);
 7

```

Since: 3.0.0
自: 3.0.0

bit_xor 比特克索尔

bit_xor(expr) - Returns the bitwise XOR of all non-null input values, or null if none.

返回所有非空输入值的按位异或，如果没有，返回 null。

Examples:

例子:

```
> SELECT bit_xor(col) FROM VALUES (3), (5) AS tab(col);
 6
```

Since: 3.0.0

自: 3.0.0

bool_and 布尔和

bool_and(expr) - Returns true if all values of `expr` are true.

Bool _ and (expr)-如果 expr 的所有值都为 true，则返回 true。

Examples:

例子:

```
> SELECT bool_and(col) FROM VALUES (true), (true), (true) AS tab(col);
 true
> SELECT bool_and(col) FROM VALUES (NULL), (true), (true) AS tab(col);
 true
> SELECT bool_and(col) FROM VALUES (true), (false), (true) AS tab(col);
 false
```

Since: 3.0.0

自: 3.0.0

bool_or 我不知道你在说什么

bool_or(expr) - Returns true if at least one value of `expr` is true.

Bool _ or (expr)-如果 expr 的至少一个值为 true，则返回 true。

Examples:

例子:

```
> SELECT bool_or(col) FROM VALUES (true), (false), (false) AS tab(col);
 true
> SELECT bool_or(col) FROM VALUES (NULL), (true), (false) AS tab(col);
 true
> SELECT bool_or(col) FROM VALUES (false), (false), (NULL) AS tab(col);
 false

```

Since: 3.0.0

自: 3.0.0

boolean 布尔

boolean(expr) - Casts the value `expr` to the target data type `boolean`.

Boolean (expr)-将值 expr 强制转换为目标数据类型 boolean。

Since: 2.0.1

自: 2.0.1

bround 自负的

bround(expr, d) - Returns `expr` rounded to `d` decimal places using HALF_EVEN rounding mode.

Bround (expr，d)-使用 HALF _ even 四舍五入模式返回四舍五入到小数点后 d 位的 expr。

Examples:

例子:

```
> SELECT bround(2.5, 0);
 2

```

Since: 2.0.0

自: 2.0.0

btrim 边缘装饰

btrim(str) - Removes the leading and trailing space characters from `str`.

Btrim (str)-从 str 中移除前导和尾随空格字符。

btrim(str, trimStr) - Remove the leading and trailing `trimStr` characters from `str`.

Btrim (str，trimStr)-从 str 中删除前导和尾随的 trimStr 字符。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- trimStr - the trim string characters to trim, the default value is a single space
- trimStr ——要修剪的修剪字符串字符，默认值是一个空格

Examples:

例子:

```
> SELECT btrim('    SparkSQL   ');
 SparkSQL
> SELECT btrim(encode('    SparkSQL   ', 'utf-8'));
 SparkSQL
> SELECT btrim('SSparkSQLS', 'SL');
 parkSQ
> SELECT btrim(encode('SSparkSQLS', 'utf-8'), encode('SL', 'utf-8'));
 parkSQ

```

Since: 3.2.0

3.2.0

cardinality 基数

cardinality(expr) - Returns the size of an array or a map. The function returns null for null input if spark.sql.legacy.sizeOfNull is set to false or spark.sql.ansi.enabled is set to true. Otherwise, the function returns -1 for null input. With the default settings, the function returns -1 for null input.

基数(expr)-返回数组或映射的大小。如果 spark.sql.legacy.sizeOfNull 设置为 false 或 spark.sql.ansi.enabled 设置为 true，那么该函数将对 null 输入返回 null。否则，函数返回 -1表示空输入。使用默认设置，函数返回 -1表示空输入。

Examples:

例子:

```
> SELECT cardinality(array('b', 'd', 'c', 'a'));
 4
> SELECT cardinality(map('a', 1, 'b', 2));
 2
> SELECT cardinality(NULL);
 -1
```

```
7
```

Since: 1.5.0

自: 1.5.0

case 案件

CASE expr1 WHEN expr2 THEN expr3 [WHEN expr4 THEN expr5]* [ELSE expr6] END - When `expr1` = `expr2`, returns `expr3`; when `expr1` = `expr4`, return `expr5`; else return `expr6`.

当 expr1 = expr3，当 expr1 = expr4，返回 expr5，ELSE 返回 expr6。

Arguments:

论点:

- expr1 - the expression which is one operand of comparison.
- Expr1-一个比较操作数的表达式。
- expr2, expr4 - the expressions each of which is the other operand of comparison.
- 表达式，每个表达式都是比较的另一个操作数。
- expr3, expr5, expr6 - the branch value expressions and else value expression should all be same type or coercible to a common type.
- Expr3，expr5，expr6——分支值表达式和 else 值表达式应该是相同的类型或者可以强制到一个公共类型。

Examples:

例子:

```
> SELECT CASE col1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE '?' END FROM VALUES 1, 2, 3;
 one
 two
 ?
> SELECT CASE col1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' END FROM VALUES 1, 2, 3;
 one
 two
 NULL

```

Since: 1.0.1

自: 1.0.1

cast 铸造

cast(expr AS type) - Casts the value `expr` to the target data type `type`.

Cast (expr AS type)-将值 expr 强制转换为目标数据类型。

Examples:

例子:

```
> SELECT cast('10' as int);
 10

```

Since: 1.0.0

自: 1.0.0

cbrt

cbrt(expr) - Returns the cube root of `expr`.

返回 expr 的立方根。

Examples:

例子:

```
> SELECT cbrt(27.0);
 3.0

```

Since: 1.4.0

自: 1.4.0

ceil 天花板

ceil(expr) - Returns the smallest integer not smaller than `expr`.

Ceil (expr)-返回不小于 expr 的最小整数。

Examples:

例子:

```
> SELECT ceil(-0.1);
 0
> SELECT ceil(5);
 5

```

Since: 1.4.0

自: 1.4.0

ceiling 天花板

ceiling(expr) - Returns the smallest integer not smaller than `expr`.

返回不小于 expr 的最小整数。

Examples:

例子:

```
> SELECT ceiling(-0.1);
 0
> SELECT ceiling(5);
 5

```

Since: 1.4.0

自: 1.4.0

char 炭

char(expr) - Returns the ASCII character having the binary equivalent to `expr`. If n is larger than 256 the result is equivalent to chr(n % 256)

Char (expr)-返回与 expr 等效的二进制 ASCII 字符。如果 n 大于256，则结果等价于公里数(n% 256)

Examples:

例子:

```
> SELECT char(65);
 A

```

Since: 2.3.0

自: 2.3.0

char_length 焦长

char_length(expr) - Returns the character length of string data or number of bytes of binary data. The length of string data includes the trailing spaces. The length of binary data includes binary zeros.

Char _ length (expr)-返回字符串数据的字符长度或二进制数据的字节数。字符串数据的长度包括尾随空格。二进制数据的长度包括二进制零。

Examples:

例子:

```
> SELECT char_length('Spark SQL ');
 10
> SELECT CHAR_LENGTH('Spark SQL ');
 10
> SELECT CHARACTER_LENGTH('Spark SQL ');
 10

```

Since: 1.5.0

自: 1.5.0

character_length 字符长

character_length(expr) - Returns the character length of string data or number of bytes of binary data. The length of string data includes the trailing spaces. The length of binary data includes binary zeros.

返回字符串数据的字符长度或二进制数据的字节数。字符串数据的长度包括尾随空格。二进制数据的长度包括二进制零。

Examples:

例子:

```
> SELECT character_length('Spark SQL ');
 10
> SELECT CHAR_LENGTH('Spark SQL ');
 10
> SELECT CHARACTER_LENGTH('Spark SQL ');
 10

```

Since: 1.5.0

自: 1.5.0

chr 克里特人

chr(expr) - Returns the ASCII character having the binary equivalent to `expr`. If n is larger than 256 the result is equivalent to chr(n % 256)

Chr (expr)-返回与 expr 等效的二进制 ASCII 字符。如果 n 大于256，则结果等价于公里数(n% 256)

Examples:
例子:

```
> SELECT chr(65);
 A

```

Since: 2.3.0
自: 2.3.0

coalesce 合并

coalesce(expr1, expr2, ...) - Returns the first non-null argument if exists. Otherwise, null.
合并(expr1，expr2，...)-如果存在，返回第一个非空参数。
Examples:
例子:

```
> SELECT coalesce(NULL, 1, NULL);
 1

```

Since: 1.0.0
自: 1.0.0

collect_list 收集列表

collect_list(expr) - Collects and returns a list of non-unique elements.
Collect _ list (expr)-收集并返回非唯一元素的列表。
Examples:
例子:

```
> SELECT collect_list(col) FROM VALUES (1), (2), (1) AS tab(col);
 [1,2,1]

```

Note:
注意:
The function is non-deterministic because the order of collected results depends on the order of the rows which may be non-deterministic after a shuffle.
该函数是不确定的，因为所收集结果的顺序取决于洗牌后可能不确定的行的顺序。

Since: 2.0.0
自: 2.0.0

collect_set 收集集合

collect_set(expr) - Collects and returns a set of unique elements.
Collect _ set (expr)-收集并返回一组惟一的元素。
Examples:
例子:

```
> SELECT collect_set(col) FROM VALUES (1), (2), (1) AS tab(col);
 [1,2]

```

Note:
注意:
The function is non-deterministic because the order of collected results depends on the order of the rows which may be non-deterministic after a shuffle.
该函数是不确定的，因为所收集结果的顺序取决于洗牌后可能不确定的行的顺序。
Since: 2.0.0
自: 2.0.0

concat 集中

concat(col1, col2, ..., colN) - Returns the concatenation of col1, col2, ..., colN.
Concat (col1，col2，... ，colN)-返回 col1，col2，... ，colN 的连接。
Examples:
例子:

```
> SELECT concat('Spark', 'SQL');
 SparkSQL
> SELECT concat(array(1, 2, 3), array(4, 5), array(6));
 [1,2,3,4,5,6]

```

Note:
注意:
Concat logic for arrays is available since 2.4.0.
自2.4.0以来，数组的 Concat 逻辑一直可用。

Since: 1.5.0
自: 1.5.0

concat_ws 让我们来看看

concat_ws(sep[, str | array(str)]+) - Returns the concatenation of the strings separated by `sep`.
Concat _ ws (sep [， str | array (str)] +)-返回由 sep. 分隔的字符串的连接。
Examples:
例子:

```
> SELECT concat_ws(' ', 'Spark', 'SQL');
  Spark SQL
> SELECT concat_ws('s');

```

Since: 1.5.0
自: 1.5.0

conv 同 “ conv”

conv(num, from_base, to_base) - Convert `num` from `from_base` to `to_base`.
Conv (num，from _ base，to _ base)-将 num 从 _ base 转换为 _ base。
Examples:
例子:

```
> SELECT conv('100', 2, 10);
 4
> SELECT conv(-10, 16, -10);
 -16

```

Since: 1.5.0
自: 1.5.0

corr 同 “ corr”

corr(expr1, expr2) - Returns Pearson coefficient of correlation between a set of number pairs.
返回一组数对之间的皮尔逊相关系数。
Examples:

例子:

```
> SELECT corr(c1, c2) FROM VALUES (3, 2), (3, 3), (6, 4) as tab(c1, c2);
 0.8660254037844387

```

Since: 1.6.0

自: 1.6.0

cos 因为

cos(expr) - Returns the cosine of `expr`, as if computed by `java.lang.Math.cos`.

Cos (expr)-返回 expr 的余弦值，如同由 java.lang.Math.cos 计算一样。

Arguments:

论点:

- expr - angle in radians 弧度扩展角

Examples:

例子:

```
> SELECT cos(0);
 1.0

```

Since: 1.4.0

自: 1.4.0

cosh

cosh(expr) - Returns the hyperbolic cosine of `expr`, as if computed by `java.lang.Math.cosh`.

返回 expr 的双曲余弦，如同由 java.lang.Math.cosh 计算一样。

Arguments:

论点:

- expr - hyperbolic angle 双曲角

Examples:

例子:

```
> SELECT cosh(0);
 1.0

```

Since: 1.4.0

自: 1.4.0

cot 帆布床

cot(expr) - Returns the cotangent of `expr`, as if computed by `1/java.lang.Math.tan`.

Cot (expr)-返回 expr 的余切，如同由1/java.lang.Math.tan 计算一样。

Arguments:

论点:

- expr - angle in radians 弧度扩展角

Examples:

例子:

```
> SELECT cot(1);
 0.6420926159343306

```

Since: 2.3.0

自: 2.3.0

count 计数

count(*) - Returns the total number of retrieved rows, including rows containing null.

返回检索到的行的总数，包括包含 null 的行。

count(expr[, expr...]) - Returns the number of rows for which the supplied expression(s) are all non-null.

Count (expr [， expr...])-返回所提供的表达式全部为非 null 的行数。

count(DISTINCT expr[, expr...]) - Returns the number of rows for which the supplied expression(s) are unique and non-null.

Count (DISTINCT expr [， expr...])-返回所提供的表达式唯一且非 null 的行数。

Examples:

例子:

```
> SELECT count(*) FROM VALUES (NULL), (5), (5), (20) AS tab(col);
 4
> SELECT count(col) FROM VALUES (NULL), (5), (5), (20) AS tab(col);
 3
> SELECT count(DISTINCT col) FROM VALUES (NULL), (5), (5), (10) AS tab(col);
```

```
 2
```

Since: 1.0.0

自: 1.0.0

count_if 如果

count_if(expr) - Returns the number of `TRUE` values for the expression.

Count _ if (expr)-返回表达式的 TRUE 值个数。

Examples:

例子:

```
> SELECT count_if(col % 2 = 0) FROM VALUES (NULL), (0), (1), (2), (3) AS tab(col);
 2
> SELECT count_if(col IS NULL) FROM VALUES (NULL), (0), (1), (2), (3) AS tab(col);
 1
```

Since: 3.0.0

自: 3.0.0

count_min_sketch 我的小品计数

count_min_sketch(col, eps, confidence, seed) - Returns a count-min sketch of a column with the given esp, confidence and seed. The result is an array of bytes, which can be deserialized to a `CountMinSketch` before usage. Count-min sketch is a probabilistic data structure used for cardinality estimation using sub-linear space.

Count _ min _ sketch (col，eps，confidence，seed)——返回一个包含给定 esp，confidence 和 seed 的列的 count-min sketch。结果是一个字节数组，可以在使用之前反序列化为 CountMinSketch。计数最小示意图是一种利用子线性空间估计基数的概率数据结构。

Examples:

例子:

```
> SELECT hex(count_min_sketch(col, 0.5d, 0.5d, 1)) FROM VALUES (1), (2), (1) AS
tab(col);

0000000100000000000000030000000100000004000000005D8D6AB90000000000000000000000000000000200000000000000010000000000000000

```

Since: 2.2.0
自: 2.2.0

covar_pop Covar pop

covar_pop(expr1, expr2) - Returns the population covariance of a set of number pairs.
Covar _ pop (expr1，expr2)-返回一组数字对的总体协方差。
Examples:
例子:

```
> SELECT covar_pop(c1, c2) FROM VALUES (1,1), (2,2), (3,3) AS tab(c1, c2);
 0.6666666666666666

```

Since: 2.0.0
自: 2.0.0

covar_samp Covar _ samp

covar_samp(expr1, expr2) - Returns the sample covariance of a set of number pairs.
Covar _ samp (expr1，expr2)-返回一组数字对的样本协方差。
Examples:
例子:

```
> SELECT covar_samp(c1, c2) FROM VALUES (1,1), (2,2), (3,3) AS tab(c1, c2);
 1.0

```

Since: 2.0.0
自: 2.0.0

crc32

crc32(expr) - Returns a cyclic redundancy check value of the `expr` as a bigint.
Crc32(expr)-将 expr 的循环冗余校验值作为 bigint 返回。
Examples:
例子:

```
> SELECT crc32('Spark');
```

```
1557323817
```

Since: 1.5.0

自: 1.5.0

cume_dist [咒语]

cume_dist() - Computes the position of a value relative to all values in the partition.

计算一个值相对于分区中所有值的位置。

Examples:

例子:

```
> SELECT a, b, cume_dist() OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   0.6666666666666666
 A1 1   0.6666666666666666
 A1 2   1.0
 A2 3   1.0
```

Since: 2.0.0

自: 2.0.0

current_catalog 当前目录

current_catalog() - Returns the current catalog.

返回当前目录。

Examples:

例子:

```
> SELECT current_catalog();
 spark_catalog
```

Since: 3.1.0

3.1.0

current_database 当前 _ 数据库

current_database() - Returns the current database.
返回当前数据库。
Examples:
例子:

```
> SELECT current_database();
 default

```

Since: 1.6.0
自: 1.6.0

current_date 当前日期

current_date() - Returns the current date at the start of query evaluation. All calls of current_date within the same query return the same value.
返回查询计算开始时的当前日期。同一查询中的所有 current _ date 调用返回相同的值。
current_date - Returns the current date at the start of query evaluation.
返回查询开始时的当前日期。
Examples:
例子:

```
> SELECT current_date();
 2020-04-25
> SELECT current_date;
 2020-04-25

```

Note:
注意:
The syntax without braces has been supported since 2.0.1.
从2.0.1开始就支持不带大括号的语法。
Since: 1.5.0
自: 1.5.0

current_timestamp 当前时间戳

current_timestamp() - Returns the current timestamp at the start of query evaluation. All calls of current_timestamp within the same query return the same value.

Current _ timestamp ()-返回查询计算开始时的当前时间戳。同一查询中的所有 current _ timestamp 调用都返回相同的值。

current_timestamp - Returns the current timestamp at the start of query evaluation.

返回查询计算开始时的当前时间戳。

Examples:

例子:

```
> SELECT current_timestamp();
 2020-04-25 15:49:11.914
> SELECT current_timestamp;
 2020-04-25 15:49:11.914

```

Note:

注意:

The syntax without braces has been supported since 2.0.1.

从2.0.1开始就支持不带大括号的语法。

Since: 1.5.0

自: 1.5.0

current_timezone 当前时区

current_timezone() - Returns the current session local timezone.

返回当前会话本地时区。

Examples:

例子:

```
> SELECT current_timezone();
 Asia/Shanghai

```

Since: 3.1.0

3.1.0

current_user 当前用户

current_user() - user name of current execution context.

Current _ user ()-当前执行上下文的用户名。

Examples:

例子:

```
> SELECT current_user();
 mockingjay

```

Since: 3.2.0

3.2.0

date 日期

date(expr) - Casts the value `expr` to the target data type `date`.

Date (expr)-将值 expr 强制转换为目标数据类型 date。

Since: 2.0.1

自: 2.0.1

date_add 2012年10月12日

date_add(start_date, num_days) - Returns the date that is `num_days` after `start_date`.

Date _ add (start _ date，num _ days)-返回开始日期后 num _ 天的日期。

Examples:

例子:

```
> SELECT date_add('2016-07-30', 1);
 2016-07-31

```

Since: 1.5.0

自: 1.5.0

date_format 格式

date_format(timestamp, fmt) - Converts `timestamp` to a value of string in the format specified by the date format `fmt`.

Date _ format (timestamp，fmt)-将时间戳转换为日期格式 fmt 指定的格式的字符串值。

Arguments:

论点:

- timestamp - A date/timestamp or string to be converted to the given format.
- Timestamp ——转换为给定格式的日期/时间戳或字符串。

- fmt - Date/time format pattern to follow. See Datetime Patterns for valid date and time format patterns.
- 要遵循的日期/时间格式模式。有效的日期和时间格式模式，请参见日期时间模式。

Examples:

例子:

```
> SELECT date_format('2016-04-08', 'y');
 2016

```

Since: 1.5.0

自: 1.5.0

date_from_unix_date 日期从 unix 的日期

date_from_unix_date(days) - Create date from the number of days since 1970-01-01.

Date _ from _ unix _ date (days)-Create date from the number of days since 1970-01-01.

Examples:

例子:

```
> SELECT date_from_unix_date(1);
 1970-01-02

```

Since: 3.1.0

3.1.0

date_part 日期部分

date_part(field, source) - Extracts a part of the date/timestamp or interval source.

Date _ part (field，source)-提取日期/时间戳或间隔源的一部分。

Arguments:

论点:

- field - selects which part of the source should be extracted, and supported string values are as same as the fields of the equivalent function EXTRACT.
- 字段-选择应该提取源的哪一部分，并且支持的字符串值与等效函数 EXTRACT 的字段相同。
- source - a date/timestamp or interval column from where field should be extracted
- Source ——应从其中提取字段的日期/时间戳或间隔列

Examples:

例子:

```
> SELECT date_part('YEAR', TIMESTAMP '2019-08-12 01:00:00.123456');
 2019
> SELECT date_part('week', timestamp'2019-08-12 01:00:00.123456');
 33
> SELECT date_part('doy', DATE'2019-08-12');
 224
> SELECT date_part('SECONDS', timestamp'2019-10-01 00:00:01.000001');
 1.000001
> SELECT date_part('days', interval 5 days 3 hours 7 minutes);
 5
> SELECT date_part('seconds', interval 5 hours 30 seconds 1 milliseconds 1
microseconds);
 30.001001
> SELECT date_part('MONTH', INTERVAL '2021-11' YEAR TO MONTH);
 11
> SELECT date_part('MINUTE', INTERVAL '123 23:55:59.002001' DAY TO SECOND);
 55

```

Note:

注意:

The date_part function is equivalent to the SQL-standard function `EXTRACT(field FROM source)`

Date _ part 函数等效于 sql 标准函数 EXTRACT (字段 FROM source)

Since: 3.0.0

自: 3.0.0

date_sub 约会潜水艇

date_sub(start_date, num_days) - Returns the date that is `num_days` before `start_date`.

Date _ sub (start _ date，num _ days)-返回开始 _ date 之前 num _ 天的日期。

Examples:

例子:

```
> SELECT date_sub('2016-07-30', 1);
 2016-07-29

```

Since: 1.5.0

自: 1.5.0

date_trunc 日期，中继

date_trunc(fmt, ts) - Returns timestamp `ts` truncated to the unit specified by the format model `fmt`.

Date_trunk (fmt，ts)-返回截断到格式模型 fmt 指定的单元的时间戳。

Arguments:

论点:

- fmt - the format representing the unit to be truncated to Fmt-表示要截断的单元的格式
  - "YEAR", "YYYY", "YY" - truncate to the first date of the year that the ts falls in, the time part will be zero out
  - “年”、“ YYYY”、“广州欢聚时代”-截断到第一个日期的一年，其中的时间部分将为零
  - "QUARTER" - truncate to the first date of the quarter that the ts falls in, the time part will be zero out
  - “ QUARTER”-截断到第一个日期的季度的 ts 下降，时间部分将零出
  - "MONTH", "MM", "MON" - truncate to the first date of the month that the ts falls in, the time part will be zero out
  - “ MONTH”、“ MM”、“ MON”-截断到 ts 所在月份的第一个日期，时间部分将为零
  - "WEEK" - truncate to the Monday of the week that the ts falls in, the time part will be zero out
  - “ WEEK”-截断到 ts 下降的那个星期的星期一，时间部分将为零
  - "DAY", "DD" - zero out the time part
  - “日”，“ DD”-零的时间部分
  - "HOUR" - zero out the minute and second with fraction part
  - “小时”-零分钟和分数部分秒
  - "MINUTE"- zero out the second with fraction part
  - “ MINUTE”-零出第二与分数部分
  - "SECOND" - zero out the second fraction part
  - “第二部分”——零出第二部分
  - "MILLISECOND" - zero out the microseconds “毫秒”——零毫秒
  - "MICROSECOND" - everything remains “微秒”——一切都保留了下来
- ts - datetime value or valid timestamp string
- Ts-datetime 值或有效的时间戳字符串

Examples:

例子:

```
> SELECT date_trunc('YEAR', '2015-03-05T09:32:05.359');
 2015-01-01 00:00:00
> SELECT date_trunc('MM', '2015-03-05T09:32:05.359');
 2015-03-01 00:00:00
```

```
> SELECT date_trunc('DD', '2015-03-05T09:32:05.359');
 2015-03-05 00:00:00
> SELECT date_trunc('HOUR', '2015-03-05T09:32:05.359');
 2015-03-05 09:00:00
> SELECT date_trunc('MILLISECOND', '2015-03-05T09:32:05.123456');
 2015-03-05 09:32:05.123

```

Since: 2.3.0

自: 2.3.0

datediff 日期差

datediff(endDate, startDate) - Returns the number of days from `startDate` to `endDate`.

Datediff (endDate，startDate)-返回从 startDate 到 endDate 的天数。

Examples:

例子:

```
> SELECT datediff('2009-07-31', '2009-07-30');
 1

> SELECT datediff('2009-07-30', '2009-07-31');
 -1

```

Since: 1.5.0

自: 1.5.0

day 日

day(date) - Returns the day of month of the date/timestamp.

天(日期)-返回日期/时间戳的月份日期。

Examples:

例子:

```
> SELECT day('2009-07-30');
 30

```

Since: 1.5.0

自: 1.5.0

dayofmonth 每月日数

dayofmonth(date) - Returns the day of month of the date/timestamp.

Dayofmonth (date)-返回日期/时间戳的月份日期。

Examples:

例子:

```
> SELECT dayofmonth('2009-07-30');
 30

```

Since: 1.5.0

自: 1.5.0

dayofweek 每周一天

dayofweek(date) - Returns the day of the week for date/timestamp (1 = Sunday, 2 = Monday, ..., 7 = Saturday).

Dayofweek (date)-返回日期/时间戳的星期几(1 = 星期日，2 = 星期一，...，7 = 星期六)。

Examples:

例子:

```
> SELECT dayofweek('2009-07-30');
 5

```

Since: 2.3.0

自: 2.3.0

dayofyear 每日

dayofyear(date) - Returns the day of year of the date/timestamp.

Dayofyear (date)-返回日期/时间戳的年份日期。

Examples:

例子:

```
> SELECT dayofyear('2016-04-09');
```

```
 100
```

Since: 1.5.0

自: 1.5.0

decimal 十进制

decimal(expr) - Casts the value `expr` to the target data type `decimal`.

Decimal (expr)-将值 expr 强制转换为目标数据类型 decimal。

Since: 2.0.1

自: 2.0.1

decode 译码

decode(bin, charset) - Decodes the first argument using the second argument character set. 解码(bin，charset)-使用第二个参数字符集解码第一个参数。
decode(expr, search, result [, search, result] ... [, default]) - Decode compares expr Decode (expr，search，result [，search，result] ... [，default])-Decode 比较 expr
to each search value one by one. If expr is equal to a search, returns the corresponding result. 如果 expr 等于一个搜索，则返回相应的结果。
If no match is found, then Oracle returns default. If default is omitted, returns null. 如果没有找到匹配项，则 Oracle 返回默认值。如果省略默认值，则返回 null。

Examples:

例子:

```
> SELECT decode(encode('abc', 'utf-8'), 'utf-8');
 abc
> SELECT decode(2, 1, 'Southlake', 2, 'San Francisco', 3, 'New Jersey', 4, 'Seattle',
'Non domestic');
 San Francisco
> SELECT decode(6, 1, 'Southlake', 2, 'San Francisco', 3, 'New Jersey', 4, 'Seattle',
'Non domestic');
 Non domestic
> SELECT decode(6, 1, 'Southlake', 2, 'San Francisco', 3, 'New Jersey', 4, 'Seattle');
 NULL
```

Since: 3.2.0

3.2.0

degrees 度

degrees(expr) - Converts radians to degrees.

Degrees (expr)-将弧度转换为度。

Arguments:

论点:

- expr - angle in radians 弧度扩展角

Examples:

例子:

```
> SELECT degrees(3.141592653589793);
 180.0

```

Since: 1.4.0

自: 1.4.0

dense_rank 稠密等级

dense_rank() - Computes the rank of a value in a group of values. The result is one plus the previously assigned rank value. Unlike the function rank, dense_rank will not produce gaps in the ranking sequence.

稠密 _ 秩()-计算一组值中某个值的秩。结果是一加上先前指定的等级值。与函数等级不同，稠密等级不会在等级序列中产生差距。

Arguments:

论点:

- children - this is to base the rank on; a change in the value of one the children will trigger a change in rank. This is an internal parameter and will be assigned by the Analyser.
- 子女——这是基于排名; 一个子女的值的变化将引发排名的变化。这是一个内部参数，将由分析器分配。

Examples:

例子:

```
> SELECT a, b, dense_rank(b) OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2),
('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   1
 A1 1   1
 A1 2   2
```

```
A2 3   1
```

Since: 2.0.0
自: 2.0.0

div 迪夫

expr1 div expr2 - Divide `expr1` by `expr2`. It returns NULL if an operand is NULL or `expr2` is 0. The result is casted to long.
Expr1 div expr2-用 expr2除以 expr1。如果操作数为 NULL 或 expr2为0，则返回 NULL。结果是长期的。
Examples:
例子:

```
> SELECT 3 div 2;
 1
```

Since: 3.0.0
自: 3.0.0

double 双

double(expr) - Casts the value `expr` to the target data type `double`.
Double (expr)-将值 expr 强制转换为目标数据类型 double。
Since: 2.0.1
自: 2.0.1

e

e() - Returns Euler's number, e.
E ()-返回欧拉数 e。
Examples:
例子:

```
> SELECT e();
 2.718281828459045
```

Since: 1.5.0
自: 1.5.0

element_at 元素 _ at

element_at(array, index) - Returns element of array at given (1-based) index. If index < 0, accesses elements from the last to the first. The function returns NULL if the index exceeds the length of the array and `spark.sql.ansi.enabled` is set to false. If `spark.sql.ansi.enabled` is set to true, it throws ArrayIndexOutOfBoundsException for invalid indices.
Element _ at (array，index)-返回给定(基于1)索引处的数组元素。如果索引 < 0，则从最后到第一个访问元素。如果索引超过数组的长度，那么该函数返回 NULL，并且 spark.sql.ansi.enabled 设置为 false。如果 spark.sql.ansi.enabled 设置为 true，则会为无效索引抛出 ArrayIndexOutOfBoundsException。
element_at(map, key) - Returns value for given key. The function returns NULL if the key is not contained in the map and `spark.sql.ansi.enabled` is set to false. If `spark.sql.ansi.enabled` is set to true, it throws NoSuchElementException instead.
元素 _ at (map，key)-返回给定键的值。如果 map 中不包含键，并且 spark.sql.ansi.enabled 被设置为 false，则该函数返回 NULL。如果 spark.sql.ansi.enabled 设置为 true，则会抛出 NoSuchElementException。
Examples:
例子:

```
> SELECT element_at(array(1, 2, 3), 2);
 2
> SELECT element_at(map(1, 'a', 2, 'b'), 2);
 b

```

Since: 2.4.0
自: 2.4.0

elt “ elt”的复数

elt(n, input1, input2, ...) - Returns the `n`-th input, e.g., returns `input2` when `n` is 2. The function returns NULL if the index exceeds the length of the array and `spark.sql.ansi.enabled` is set to false. If `spark.sql.ansi.enabled` is set to true, it throws ArrayIndexOutOfBoundsException for invalid indices.

Elt (n，input1，input2，...)-返回第 n 个输入，例如，当 n = 2时返回 input2。如果索引超过数组的长度，那么该函数返回 NULL，并且 spark.sql.ansi.enabled 设置为 false。如果 spark.sql.ansi.enabled 设置为 true，则会为无效索引抛出 ArrayIndexOutOfBoundsException。

Examples:

例子:

```
> SELECT elt(1, 'scala', 'java');
 scala

```

Since: 2.0.0

自: 2.0.0

encode 编码

encode(str, charset) - Encodes the first argument using the second argument character set.

Encode (str，charset)-使用第二个参数字符集对第一个参数进行编码。

Examples:

例子:

```
> SELECT encode('abc', 'utf-8');
 abc

```

Since: 1.5.0

自: 1.5.0

every 每一个

every(expr) - Returns true if all values of `expr` are true.

Every (expr)-如果 expr 的所有值都为 true，则返回 true。

Examples:

例子:

```
> SELECT every(col) FROM VALUES (true), (true), (true) AS tab(col);
 true
> SELECT every(col) FROM VALUES (NULL), (true), (true) AS tab(col);
 true
> SELECT every(col) FROM VALUES (true), (false), (true) AS tab(col);
 false
```

```
7
```

Since: 3.0.0

自: 3.0.0

exists 存在

exists(expr, pred) - Tests whether a predicate holds for one or more elements in the array.

Exists (expr，pred)-测试谓词是否保留数组中的一个或多个元素。

Examples:

例子:

```
> SELECT exists(array(1, 2, 3), x -> x % 2 == 0);
 true
> SELECT exists(array(1, 2, 3), x -> x % 2 == 10);
 false
> SELECT exists(array(1, null, 3), x -> x % 2 == 0);
 NULL
> SELECT exists(array(0, null, 2, 3, null), x -> x IS NULL);
 true
> SELECT exists(array(1, 2, 3), x -> x IS NULL);
 false

```

Since: 2.4.0

自: 2.4.0

exp 经验

exp(expr) - Returns e to the power of `expr`.

Exp (expr)-将 e 返回到 expr 的幂。

Examples:

例子:

```
> SELECT exp(0);
 1.0

```

Since: 1.4.0

自: 1.4.0

explode 爆炸

explode(expr) - Separates the elements of array `expr` into multiple rows, or the elements of map `expr` into multiple rows and columns. Unless specified otherwise, uses the default column name `col` for elements of the array or `key` and `value` for the elements of the map.
Explode (expr)-将数组 expr 的元素分解为多行，或者将映射 expr 的元素分解为多行和多列。除非另有指定，否则对数组的元素使用默认列名 col，对映射的元素使用键和值。

Examples:

例子:

```
> SELECT explode(array(10, 20));
 10
 20

```

Since: 1.0.0

自: 1.0.0

explode_outer 外部爆炸

explode_outer(expr) - Separates the elements of array `expr` into multiple rows, or the elements of map `expr` into multiple rows and columns. Unless specified otherwise, uses the default column name `col` for elements of the array or `key` and `value` for the elements of the map.
Explode _ outer (expr)-将数组扩展的元素分解为多行，或者将映射扩展的元素分解为多行和多列。除非另有指定，否则对数组的元素使用默认列名 col，对映射的元素使用键和值。

Examples:

例子:

```
> SELECT explode_outer(array(10, 20));
 10
 20

```

Since: 1.0.0

自: 1.0.0

expm1 爆炸

expm1(expr) - Returns exp(expr) - 1.

Expm1(expr)-Returns exp (expr)-1.

Examples:

例子:

```
> SELECT expm1(0);
 0.0

```

Since: 1.4.0

自: 1.4.0

extract 提取

extract(field FROM source) - Extracts a part of the date/timestamp or interval source.

提取(从源中提取字段)-提取日期/时间戳或间隔源的一部分。

Arguments:

论点:

- field - selects which part of the source should be extracted 字段选择应该提取源的哪个部分
  - Supported string values of 支持的字符串值field for dates and timestamps are(case insensitive): 对于日期和时间戳是(不区分大小写) :
    - "YEAR", ("Y", "YEARS", "YR", "YRS") - the year field
    - “年”，(“ y”，“岁”，“ YR”，“ YRS”)-年的领域
    - "YEAROFWEEK" - the ISO 8601 week-numbering year that the datetime falls in. For example, 2005-01-02 is part of the 53rd week of year 2004, so the result is 2004
    - “ YEAROFWEEK”—— iso8601周——日期时间所属的年份。例如，2005-01-02是2004年第53周的一部分，所以结果是2004年
    - "QUARTER", ("QTR") - the quarter (1 - 4) of the year that the datetime falls in
    - “ QUARTER”，(“ QTR”)-日期时间所在年份的季度(1-4)
    - "MONTH", ("MON", "MONS", "MONTHS") - the month field (1 - 12)
    - “ MONTH”，(“ MON”，“ MONS”，“ MONTHS”)-the MONTH field (1-12)
    - "WEEK", ("W", "WEEKS") - the number of the ISO 8601 week-of-week-based-year. A week is considered to start on a Monday and week 1 is the first week with >3 days. In the ISO week-numbering system, it is possible for early-January dates to be part of the 52nd or 53rd week of the previous year, and for late-December dates to be part of the first week of the next year. For example, 2005-01-02 is part of the 53rd week of year 2004, while 2012-12-31 is part of the first week of 2013

        - “ WEEK”(“ w”，“ WEEKS”)——以每周为基础的 ISO 8601年的数字。一周被认为是从星期一开始，第一周是第一个超过3天的星期。在 ISO 周编号系统中，1月初的日期可以是前一年第52周或第53周的一部分，12月底的日期可以是下一年第一周的一部分。例如，2005-01-02是2004年第53周的一部分，而2012-12-31是2013年第一周的一部分
        - "DAY", ("D", "DAYS") - the day of the month field (1 - 31)
        - “ DAY”，(“ d”，“ DAYS”)-月份的日子字段(1-31)
        - "DAYOFWEEK",("DOW") - the day of the week for datetime as Sunday(1) to Saturday(7)
        - “ DAYOFWEEK”，(“ DOW”)-一周中的某一天，日期时间为星期日(1)到星期六(7)
        - "DAYOFWEEK_ISO",("DOW_ISO") - ISO 8601 based day of the week for datetime as Monday(1) to Sunday(7)
        - “ DAYOFWEEK _ ISO”，(“ DOW _ ISO”)-基于 ISO 8601的每周日期时间为星期一(1)到星期日(7)
        - "DOY" - the day of the year (1 - 365/366)
        - “ DOY”-一年中的一天(1-365/366)
        - "HOUR", ("H", "HOURS", "HR", "HRS") - The hour field (0 - 23)
        - “小时”，(“ h”，“小时”，“ HR”，“ HRS”)-小时字段(0-23)
        - "MINUTE", ("M", "MIN", "MINS", "MINUTES") - the minutes field (0 - 59)
        - “ MINUTE”，(“ m”，“ MIN”，“ MINS”，“ MINUTES”)-分钟字段(0-59)
        - "SECOND", ("S", "SEC", "SECONDS", "SECS") - the seconds field, including fractional parts
        - “ SECOND”，(“ s”，“ SEC”，“ SECONDS”，“ SECS”)-秒场，包括小数部分
    - Supported string values of 支持的字符串值field for interval(which consists of 对于 interval (包括 months, days, microseconds) are(case insensitive):)(不区分大小写) :
        - "YEAR", ("Y", "YEARS", "YR", "YRS") - the total months / 12
        - “年”，(“ y”，“岁”，“ YR”，“ YRS”)-总月数/12
        - "MONTH", ("MON", "MONS", "MONTHS") - the total months % 12
        - “ MONTH”，(“ MON”，“ MONS”，“ MONTHS”)——总月数% 12
        - "DAY", ("D", "DAYS") - the days part of interval
        - “ DAY”，(“ d”，“ DAYS”)——间隔的天部分
        - "HOUR", ("H", "HOURS", "HR", "HRS") - how many hours the microseconds contains
        - “ HOUR”(“ h”，“ HOURS”，“ HR”，“ HRS”)-微秒包含多少小时
        - "MINUTE", ("M", "MIN", "MINS", "MINUTES") - how many minutes left after taking hours from microseconds
        - “ MINUTE”，(“ m”，“ MIN”，“ MINS”，“ MINUTES”)-从微秒到小时还剩多少分钟
        - "SECOND", ("S", "SEC", "SECONDS", "SECS") - how many second with fractions left after taking hours and minutes from microseconds
        - “秒”，(“ s”，“ SEC”，“ SECONDS”，“ SECS”)-从微秒到小时和分钟，还剩下多少秒
- source - a date/timestamp or interval column from where field should be extracted
- Source ——应从其中提取字段的日期/时间戳或间隔列

Examples:

例子:

```
> SELECT extract(YEAR FROM TIMESTAMP '2019-08-12 01:00:00.123456');
 2019
> SELECT extract(week FROM timestamp'2019-08-12 01:00:00.123456');
 33
> SELECT extract(doy FROM DATE'2019-08-12');
 224
> SELECT extract(SECONDS FROM timestamp'2019-10-01 00:00:01.000001');
 1.000001
> SELECT extract(days FROM interval 5 days 3 hours 7 minutes);
 5
> SELECT extract(seconds FROM interval 5 hours 30 seconds 1 milliseconds 1
microseconds);
 30.001001
> SELECT extract(MONTH FROM INTERVAL '2021-11' YEAR TO MONTH);
 11
> SELECT extract(MINUTE FROM INTERVAL '123 23:55:59.002001' DAY TO SECOND);
 55

```

Note:

注意:

The extract function is equivalent to `date_part(field, source)`.

Extract 函数等价于 date _ part (field，source) 。

Since: 3.0.0

自: 3.0.0

factorial 阶乘的

factorial(expr) - Returns the factorial of `expr`. `expr` is [0..20]. Otherwise, null.

Factorial (expr)-返回 expr.expr 的阶乘为[0. . 20]。

Examples:

例子:

```
> SELECT factorial(5);
 120

```

Since: 1.5.0
自: 1.5.0

filter 过滤器

filter(expr, func) - Filters the input array using the given predicate.
Filter (expr，func)-使用给定的谓词对输入数组进行筛选。
Examples:
例子:

```
> SELECT filter(array(1, 2, 3), x -> x % 2 == 1);
 [1,3]
> SELECT filter(array(0, 2, 3), (x, i) -> x > i);
 [2,3]
> SELECT filter(array(0, null, 2, 3, null), x -> x IS NOT NULL);
 [0,2,3]

```

Note:
注意:
The inner function may use the index argument since 3.0.0.
内部函数可以使用3.0.0以来的索引参数。
Since: 2.4.0
自: 2.4.0

find_in_set 在片场找到

find_in_set(str, str_array) - Returns the index (1-based) of the given string (`str`) in the comma-delimited list (`str_array`). Returns 0, if the string was not found or if the given string (`str`) contains a comma.
Find _ in _ set (str，str _ array)-返回逗号分隔列表(str _ array)中给定字符串(str)的索引(基于1)。如果找不到字符串或给定的字符串(str)包含逗号，则返回0。
Examples:
例子:

```
> SELECT find_in_set('ab','abc,b,ab,c,def');
 3

```

Since: 1.5.0
自: 1.5.0

first 第一

first(expr[, isIgnoreNull]) - Returns the first value of `expr` for a group of rows. If `isIgnoreNull` is true, returns only non-null values.
First (expr [， isIgnoreNull])-为一组行返回 expr 的第一个值。如果 isIgnoreNull 为 true，则只返回非空值。
Examples:
例子:

```
> SELECT first(col) FROM VALUES (10), (5), (20) AS tab(col);
 10
> SELECT first(col) FROM VALUES (NULL), (5), (20) AS tab(col);
 NULL
> SELECT first(col, true) FROM VALUES (NULL), (5), (20) AS tab(col);
 5

```

Note:
注意:
The function is non-deterministic because its results depends on the order of the rows which may be non-deterministic after a shuffle.
该函数是不确定的，因为它的结果取决于洗牌后可能不确定的行的顺序。
Since: 2.0.0
自: 2.0.0

first_value 第一值

first_value(expr[, isIgnoreNull]) - Returns the first value of `expr` for a group of rows.
If `isIgnoreNull` is true, returns only non-null values.
First _ value (expr [， isIgnoreNull])-为一组行返回 expr 的第一个值。如果 isIgnoreNull 为 true，则只返回非空值。
Examples:
例子:

```
> SELECT first_value(col) FROM VALUES (10), (5), (20) AS tab(col);
```

```
 10
> SELECT first_value(col) FROM VALUES (NULL), (5), (20) AS tab(col);
 NULL
> SELECT first_value(col, true) FROM VALUES (NULL), (5), (20) AS tab(col);
 5

```

Note:

注意:

The function is non-deterministic because its results depends on the order of the rows which may be non-deterministic after a shuffle.

该函数是不确定的，因为它的结果取决于洗牌后可能不确定的行的顺序。

Since: 2.0.0

自: 2.0.0

flatten 压平

flatten(arrayOfArrays) - Transforms an array of arrays into a single array.

平面化(arrayOfArrays)-将数组数组转换为单个数组。

Examples:

例子:

```
> SELECT flatten(array(array(1, 2), array(3, 4)));
 [1,2,3,4]

```

Since: 2.4.0

自: 2.4.0

float 浮动

float(expr) - Casts the value `expr` to the target data type `float`.

Float (expr)-将值 expr 强制转换为目标数据类型 float。

Since: 2.0.1

自: 2.0.1

floor 地板

floor(expr) - Returns the largest integer not greater than `expr`.

返回不大于 expr 的最大整数。

Examples:

例子:

```
> SELECT floor(-0.1);
 -1
> SELECT floor(5);
 5

```

Since: 1.4.0

自: 1.4.0

forall 为了所有

forall(expr, pred) - Tests whether a predicate holds for all elements in the array.

Forall (expr，pred)-测试谓词是否适用于数组中的所有元素。

Examples:

例子:

```
> SELECT forall(array(1, 2, 3), x -> x % 2 == 0);
 false
> SELECT forall(array(2, 4, 8), x -> x % 2 == 0);
 true
> SELECT forall(array(1, null, 3), x -> x % 2 == 0);
 false
> SELECT forall(array(2, null, 8), x -> x % 2 == 0);
 NULL

```

Since: 3.0.0

自: 3.0.0

format_number 格式 _ 数

format_number(expr1, expr2) - Formats the number `expr1` like '#,###,###.##', rounded to `expr2` decimal places. If `expr2` is 0, the result has no decimal point or fractional part. `expr2` also accept a user specified format. This is supposed to function like MySQL's FORMAT.

Format _ number (expr1，expr2)-将数字 expr1格式设置为’＃，＃＃＃，＃＃＃。＃＃’四舍五入到小数点后2位。如果 expr2为0，则结果没有小数点或小数部分。Expr2还接受用户指定的格式。这应该像 MySQL 的格式一样运行。

Examples:

例子:

```
> SELECT format_number(12332.123456, 4);
 12,332.1235
> SELECT format_number(12332.123456, '##################.###');
 12332.123

```

Since: 1.5.0

自: 1.5.0

format_string 格式 _ 字符串

format_string(strfmt, obj, ...) - Returns a formatted string from printf-style format strings.

Format _ string (strfmt，obj，...)-从 printf 样式的格式字符串返回格式化的字符串。

Examples:

例子:

```
> SELECT format_string("Hello World %d %s", 100, "days");
 Hello World 100 days

```

Since: 1.5.0

自: 1.5.0

from_csv 来自 csv

from_csv(csvStr, schema[, options]) - Returns a struct value with the given `csvStr` and `schema`.

From _ csv (csvStr，schema [，options])-返回具有给定 csvStr 和模式的结构体值。

Examples:

例子:

```
> SELECT from_csv('1, 0.8', 'a INT, b DOUBLE');
 {"a":1,"b":0.8}
> SELECT from_csv('26/08/2015', 'time Timestamp', map('timestampFormat', 'dd/MM/yyyy'));
 {"time":2015-08-26 00:00:00}
```

```
5
```

Since: 3.0.0

自: 3.0.0

from_json 来自杰森

from_json(jsonStr, schema[, options]) - Returns a struct value with the given `jsonStr` and `schema`.

From _ json (jsonStr，schema [， options])-返回具有给定 jsonStr 和模式的结构值。

Examples:

例子:

```
> SELECT from_json('{"a":1, "b":0.8}', 'a INT, b DOUBLE');
 {"a":1,"b":0.8}
> SELECT from_json('{"time":"26/08/2015"}', 'time Timestamp', map('timestampFormat', 'dd/MM/yyyy'));
 {"time":2015-08-26 00:00:00}

```

Since: 2.2.0

自: 2.2.0

from_unixtime 来自 unixtime

from_unixtime(unix_time[, fmt]) - Returns `unix_time` in the specified `fmt`.

从 _ unixtime (unix _ time [， fmt])返回指定 fmt 中的 unix _ time。

Arguments:

论点:

- unix_time - UNIX Timestamp to be converted to the provided format.
- UNIX _ time-UNIX Timestamp 转换为所提供的格式。
- fmt - Date/time format pattern to follow. See Datetime Patterns for valid date and time format patterns. The 'yyyy-MM-dd HH:mm:ss' pattern is used if omitted.
- Fmt-要遵循的日期/时间格式模式。有关有效的日期和时间格式模式，请参见日期时间模式。如果省略，则使用“ yyyy-MM-dd HH: mm: ss”模式。

Examples:

例子:

```
> SELECT from_unixtime(0, 'yyyy-MM-dd HH:mm:ss');
```

```
 1969-12-31 16:00:00

> SELECT from_unixtime(0);
 1969-12-31 16:00:00

```

Since: 1.5.0
自: 1.5.0

from_utc_timestamp 时间戳

from_utc_timestamp(timestamp, timezone) - Given a timestamp like '2017-07-14 02:40:00.0', interprets it as a time in UTC, and renders that time as a timestamp in the given time zone. For example, 'GMT+1' would yield '2017-07-14 03:40:00.0'.
From _ UTC _ timestamp (timestamp，timezone)——给定一个像“2017-07-1402:40:00.0”这样的时间戳，将其解释为 UTC 时间，并将其作为给定时区的时间戳。例如，“ GMT + 1”将产生“2017-07-1403:40:00.0”。
Examples:
例子:

```
> SELECT from_utc_timestamp('2016-08-31', 'Asia/Seoul');
 2016-08-31 09:00:00

```

Since: 1.5.0
自: 1.5.0

get_json_object 获取 json 对象

get_json_object(json_txt, path) - Extracts a json object from `path`.
Get _ json _ object (json _ txt，path)-从 path 中提取 json 对象。
Examples:
例子:

```
> SELECT get_json_object('{"a":"b"}', '$.a');
 b

```

Since: 1.5.0
自: 1.5.0

getbit 获得位

getbit(expr, pos) - Returns the value of the bit (0 or 1) at the specified position. The positions are numbered from right to left, starting at zero. The position argument cannot be negative.
Getbit (expr，pos)-返回指定位置的位(0或1)的值。位置从右到左编号，从零开始。位置参数不能是否定的。

Examples:

例子:

```
> SELECT getbit(11, 0);
 1
> SELECT getbit(11, 2);
 0

```

Since: 3.2.0

3.2.0

greatest 最伟大的

greatest(expr, ...) - Returns the greatest value of all parameters, skipping null values.
返回所有参数的最大值，跳过空值。

Examples:

例子:

```
> SELECT greatest(10, 9, 2, 4, 3);
 10

```

Since: 1.5.0

自: 1.5.0

grouping 分组

grouping(col) - indicates whether a specified column in a GROUP BY is aggregated or not, returns 1 for aggregated or 0 for not aggregated in the result set.",
Grouping (col)——指示 GROUP BY 中的指定列是否聚合，返回聚合列1，返回结果集中未聚合列0。”，

Examples:

例子:

```
> SELECT name, grouping(name), sum(age) FROM VALUES (2, 'Alice'), (5, 'Bob') people(age, name) GROUP BY cube(name);
  Alice 0   2
  Bob   0   5
  NULL  1   7
```

Since: 2.0.0

自: 2.0.0

grouping_id 分组 _ id

grouping_id([col1[, col2 ..]]) - returns the level of grouping, equals to `(grouping(c1) << (n-1)) + (grouping(c2) << (n-2)) + ... + grouping(cn)`

Grouping _ id ([col1[， col2..]])- 返回分组水平，等于(分组(c1) < (n-1)) + (分组(c2) < (n-2)) + ... + 分组(cn)

Examples:

例子:

```
> SELECT name, grouping_id(), sum(age), avg(height) FROM VALUES (2, 'Alice', 165), (5, 'Bob', 180) people(age, name, height) GROUP BY cube(name, height);
  Alice 0   2   165.0
  Alice 1   2   165.0
  NULL  3   7   172.5
  Bob   0   5   180.0
  Bob   1   5   180.0
  NULL  2   2   165.0
  NULL  2   5   180.0
```

Note:

注意:

Input columns should match with grouping columns exactly, or empty (means all the grouping columns).

输入列应该与分组列完全匹配，或者为空(表示所有的分组列)。

Since: 2.0.0

自: 2.0.0

hash 大麻

hash(expr1, expr2, ...) - Returns a hash value of the arguments.

散列(expr1，expr2，...)-返回参数的散列值。

Examples:

例子:

```
> SELECT hash('Spark', array(123), 2);
 -1321691492

```

Since: 2.0.0

自: 2.0.0

hex 魔法

hex(expr) - Converts `expr` to hexadecimal.

十六进制(expr)-将 expr 转换为十六进制。

Examples:

例子:

```
> SELECT hex(17);
 11
> SELECT hex('Spark SQL');
 537061726B2053514C

```

Since: 1.5.0

自: 1.5.0

hour 一小时

hour(timestamp) - Returns the hour component of the string/timestamp.

Hour (timestamp)-返回字符串/时间戳的 hour 组件。

Examples:

例子:

```
> SELECT hour('2009-07-30 12:58:59');
```

```
 12
```

Since: 1.5.0

自: 1.5.0

hypot 抵押

hypot(expr1, expr2) - Returns sqrt(`expr1`**2 +** **`expr2`**2).

Hypot (expr1，expr2)-Returns sqrt (expr12 + expr22).

Examples:

例子:

```
> SELECT hypot(3, 4);
 5.0
```

Since: 1.4.0

自: 1.4.0

if 如果

if(expr1, expr2, expr3) - If `expr1` evaluates to true, then returns `expr2`; otherwise returns `expr3`.

如果(expr1，expr2，expr3)-如果 expr1计算为 true，则返回 expr2; 否则返回 expr3。

Examples:

例子:

```
> SELECT if(1 < 2, 'a', 'b');
 a
```

Since: 1.0.0

自: 1.0.0

ifnull

ifnull(expr1, expr2) - Returns `expr2` if `expr1` is null, or `expr1` otherwise.

Ifnull (expr1，expr2)-如果 expr1为 null，返回 expr2，否则返回 expr1。

Examples:

例子:

```
> SELECT ifnull(NULL, array('2'));
 ["2"]

```

Since: 2.0.0

自: 2.0.0

in 在

expr1 in(expr2, expr3, ...) - Returns true if `expr` equals to any valN.

Expr1 in (expr2，expr3，...)-如果 expr 等于任何 valN，返回 true。

Arguments:

论点:

- expr1, expr2, expr3, ... - the arguments must be same type.
- Expr1，expr2，expr3，...-参数必须是相同的类型。

Examples:

例子:

```
> SELECT 1 in(1, 2, 3);
 true
> SELECT 1 in(2, 3, 4);
 false
> SELECT named_struct('a', 1, 'b', 2) in(named_struct('a', 1, 'b', 1), named_struct('a', 1, 'b', 3));
 false
> SELECT named_struct('a', 1, 'b', 2) in(named_struct('a', 1, 'b', 2), named_struct('a', 1, 'b', 3));
 true

```

Since: 1.0.0

自: 1.0.0

initcap 开始

initcap(str) - Returns `str` with the first letter of each word in uppercase. All other letters are in lowercase. Words are delimited by white space.

Initcap (str)——返回大写的每个单词的第一个字母。其他字母都是小写的。单词由空格分隔。

Examples:
例子:

```
> SELECT initcap('sPark sql');
 Spark Sql

```

Since: 1.5.0
自: 1.5.0

inline 内嵌的

inline(expr) - Explodes an array of structs into a table. Uses column names col1, col2, etc. by default unless specified otherwise.
内联(扩展)-将一个结构体数组爆炸为一个表。默认情况下使用列名 col1、 col2等，除非另有指定。
Examples:
例子:

```
> SELECT inline(array(struct(1, 'a'), struct(2, 'b')));
 1  a
 2  b

```

Since: 2.0.0
自: 2.0.0

inline_outer Inline _ outer

inline_outer(expr) - Explodes an array of structs into a table. Uses column names col1, col2, etc. by default unless specified otherwise.
Inline _ outer (expr)-将一个结构体数组爆炸为一个表。默认情况下使用列名 col1、 col2等，除非另有指定。
Examples:
例子:

```
> SELECT inline_outer(array(struct(1, 'a'), struct(2, 'b')));
 1  a
 2  b

```

Since: 2.0.0
自: 2.0.0

input_file_block_length 输入文件 _ block _ length

input_file_block_length() - Returns the length of the block being read, or -1 if not available.
Input _ file _ block _ length ()-返回正在读取的块的长度，如果不可用，返回 -1。
Examples:
例子:

```
> SELECT input_file_block_length();
 -1

```

Since: 2.2.0
自: 2.2.0

input_file_block_start 输入文件块开始

input_file_block_start() - Returns the start offset of the block being read, or -1 if not available.
Input _ file _ block _ start ()-返回正在读取的块的起始偏移量，如果不可用，返回 -1。
Examples:
例子:

```
> SELECT input_file_block_start();
 -1

```

Since: 2.2.0
自: 2.2.0

input_file_name 输入 _ file _ name

input_file_name() - Returns the name of the file being read, or empty string if not available.
Input _ file _ name ()-返回正在读取的文件的名称，如果不可用，则返回空字符串。
Examples:
例子:

```
> SELECT input_file_name();
```

```
2
```

Since: 1.5.0

自: 1.5.0

instr 装置

instr(str, substr) - Returns the (1-based) index of the first occurrence of `substr` in `str`.

Instr (str，substr)-返回在 str 中第一个出现的 substr 的索引(基于1)。

Examples:

例子:

```
> SELECT instr('SparkSQL', 'SQL');
 6

```

Since: 1.5.0

自: 1.5.0

int

int(expr) - Casts the value `expr` to the target data type `int`.

Int (expr)-将值 expr 强制转换为目标数据类型 int。

Since: 2.0.1

自: 2.0.1

isnan 伊斯南

isnan(expr) - Returns true if `expr` is NaN, or false otherwise.

Isnan (expr)-如果 expr 是 NaN，返回 true，否则返回 false。

Examples:

例子:

```
> SELECT isnan(cast('NaN' as double));
 true

```

Since: 1.5.0

自: 1.5.0

isnotnull 不等于零

isnotnull(expr) - Returns true if `expr` is not null, or false otherwise.

如果 expr 不为 null，返回 true，否则返回 false。

Examples:

例子:

```
> SELECT isnotnull(1);
 true

```

Since: 1.0.0

自: 1.0.0

isnull 等值线

isnull(expr) - Returns true if `expr` is null, or false otherwise.

如果 expr 为 null，返回 true，否则返回 false。

Examples:

例子:

```
> SELECT isnull(1);
 false

```

Since: 1.0.0

自: 1.0.0

java_method 方法

java_method(class, method[, arg1[, arg2 ..]]) - Calls a method with reflection.

Java _ method (class，method [，arg1[，arg2. .])-调用具有反射的方法。

Examples:

例子:

```
> SELECT java_method('java.util.UUID', 'randomUUID');
 c33fb387-8500-4bfa-81d2-6e0e3e930df2
> SELECT java_method('java.util.UUID', 'fromString', 'a5cf6c42-0c85-418f-af6c-3e4e5b1328f2');
```

```
a5cf6c42-0c85-418f-af6c-3e4e5b1328f2
```

Since: 2.0.0

自: 2.0.0

json_array_length 数组长度

json_array_length(jsonArray) - Returns the number of elements in the outermost JSON array.

JSON _ array _ length (jsonArray)-返回最外层 JSON 数组中的元素数。

Arguments:

论点:

- jsonArray - A JSON array. NULL is returned in case of any other valid JSON string, NULL or an invalid JSON.
- JSON 阵列—— JSON 阵列。如果有任何其他有效的 JSON 字符串、 NULL 或无效的 JSON，则返回 NULL。

Examples:

例子:

```
> SELECT json_array_length('[1,2,3,4]');
  4
> SELECT json_array_length('[1,2,3,{"f1":1,"f2":[5,6]},4]');
  5
> SELECT json_array_length('[1,2');
  NULL
```

Since: 3.1.0

3.1.0

json_object_keys 对象键

json_object_keys(json_object) - Returns all the keys of the outermost JSON object as an array.

JSON _ object _ keys (JSON _ object)-以数组的形式返回最外层 JSON 对象的所有键。

Arguments:

论点:

- json_object - A JSON object. If a valid JSON object is given, all the keys of the outermost object will be returned as an array. If it is any other valid JSON string, an invalid JSON string or an empty string, the function returns null.
- JSON _ object-一个 JSON 对象。如果给定一个有效的 JSON 对象，则最外层对象的所有键都将作为数组返回。如果它是任何其他有效的 JSON 字符串、无效的 JSON 字符串或空字符串，则函数返回 null。

Examples:

例子:

```
> SELECT json_object_keys('{}');
  []
> SELECT json_object_keys('{"key": "value"}');
  ["key"]
> SELECT json_object_keys('{"f1":"abc","f2":{"f3":"a", "f4":"b"}}');
  ["f1","f2"]
```

Since: 3.1.0

3.1.0

json_tuple 这是我第一次来这里

json_tuple(jsonStr, p1, p2, ..., pn) - Returns a tuple like the function get_json_object, but it takes multiple names. All the input parameters and output column types are string.

Json _ tuple (jsonStr，p1，p2，... ，pn)——返回一个类似函数 get _ json _ object 的元组，但它有多个名称。所有输入参数和输出列类型都是字符串。

Examples:

例子:

```
> SELECT json_tuple('{"a":1, "b":2}', 'a', 'b');
 1  2
```

Since: 1.6.0

自: 1.6.0

kurtosis 峰度

kurtosis(expr) - Returns the kurtosis value calculated from values of a group.

峭度(expr)-返回从组的值计算出来的峭度值。

Examples:

例子:

```
> SELECT kurtosis(col) FROM VALUES (-10), (-20), (100), (1000) AS tab(col);
 -0.7014368047529627
> SELECT kurtosis(col) FROM VALUES (1), (10), (100), (10), (1) as tab(col);
```

```
0.19432323191699075
```

Since: 1.6.0
自: 1.6.0

lag 滞后

lag(input[, offset[, default]]) - Returns the value of `input` at the `offset`th row before the current row in the window. The default value of `offset` is 1 and the default value of `default` is null. If the value of `input` at the `offset`th row is null, null is returned. If there is no such offset row (e.g., when the offset is 1, the first row of the window does not have any previous row), `default` is returned.
延迟(input [， offset [， default])-返回窗口中当前行之前的偏移量行的输入值。偏移量的默认值为1，默认值为 null。如果位于偏移量行的输入值为 null，则返回 null。如果没有这样的偏移行(例如，当偏移量为1时，窗口的第一行没有以前的任何行) ，则返回默认值。
Arguments:
论点:

- input - a string expression to evaluate offset rows before the current row.
- Input-在当前行之前计算偏移量行的字符串表达式。
- offset - an int expression which is rows to jump back in the partition.
- 偏移量-一个 int 表达式，它是要跳回到分区中的行。
- default - a string expression which is to use when the offset row does not exist.
- Default-当偏移行不存在时使用的字符串表达式。

Examples:
例子:

```
> SELECT a, b, lag(b) OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
A1 1   NULL
A1 1   1
A1 2   1
A2 3   NULL
```

Since: 2.0.0
自: 2.0.0

last 最后

last(expr[, isIgnoreNull]) - Returns the last value of `expr` for a group of rows. If `isIgnoreNull` is true, returns only non-null values

Last (expr [， isIgnoreNull])-返回一组行的 expr 的最后一个值。如果 isIgnoreNull 为 true，则只返回非空值

Examples:

例子:

```
> SELECT last(col) FROM VALUES (10), (5), (20) AS tab(col);
 20
> SELECT last(col) FROM VALUES (10), (5), (NULL) AS tab(col);
 NULL
> SELECT last(col, true) FROM VALUES (10), (5), (NULL) AS tab(col);
 5

```

Note:

注意:

The function is non-deterministic because its results depends on the order of the rows which may be non-deterministic after a shuffle.

该函数是不确定的，因为它的结果取决于洗牌后可能不确定的行的顺序。

Since: 2.0.0

自: 2.0.0

last_day 最后一天

last_day(date) - Returns the last day of the month which the date belongs to.

Last _ day (date)-返回该日期所属的月份的最后一天。

Examples:

例子:

```
> SELECT last_day('2009-01-12');
 2009-01-31

```

Since: 1.5.0

自: 1.5.0

last_value 最后值

last_value(expr[, isIgnoreNull]) - Returns the last value of `expr` for a group of rows.

If `isIgnoreNull` is true, returns only non-null values

Last _ value (expr [， isIgnoreNull])-返回一组行的 expr 的最后一个值。如果 isIgnoreNull 为 true，则只返回非空值

Examples:

例子:

```
> SELECT last_value(col) FROM VALUES (10), (5), (20) AS tab(col);
 20
> SELECT last_value(col) FROM VALUES (10), (5), (NULL) AS tab(col);
 NULL
> SELECT last_value(col, true) FROM VALUES (10), (5), (NULL) AS tab(col);
 5

```

Note:

注意:

The function is non-deterministic because its results depends on the order of the rows which may be non-deterministic after a shuffle.

该函数是不确定的，因为它的结果取决于洗牌后可能不确定的行的顺序。

Since: 2.0.0

自: 2.0.0

lcase

lcase(str) - Returns `str` with all characters changed to lowercase.

Lcase (str)-返回所有字符都更改为小写的 str。

Examples:

例子:

```
> SELECT lcase('SparkSql');
 sparksql

```

Since: 1.0.1

自: 1.0.1

lead 铅

lead(input[, offset[, default]]) - Returns the value of `input` at the `offset`th row after the current row in the window. The default value of `offset` is 1 and the default value of `default` is null. If the value of `input` at the `offset`th row is null, null is returned. If there is no such an offset row (e.g., when the offset is 1, the last row of the window does not have any subsequent row), `default` is returned.

Lead (input [， offset [， default])-返回窗口中当前行之后的偏移量行的输入值。偏移量的默认值为1，默认值为 null。如果位于偏移量行的输入值为 null，则返回 null。如果没有这样的偏移行(例如，当偏移量为1时，窗口的最后一行没有任何后续行) ，则返回默认值。

Arguments:

论点:

- input - a string expression to evaluate offset rows after the current row.
- Input-用于计算当前行之后的偏移量行的字符串表达式。
- offset - an int expression which is rows to jump ahead in the partition.
- 偏移量-一个 int 表达式，它是在分区中向前跳转的行。
- default - a string expression which is to use when the offset is larger than the window. The default value is null.
- Default-当偏移量大于窗口时使用的字符串表达式。默认值为空。

Examples:

例子:

```
> SELECT a, b, lead(b) OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   1
 A1 1   2
 A1 2   NULL
 A2 3   NULL

```

Since: 2.0.0

自: 2.0.0

least 最少

least(expr, ...) - Returns the least value of all parameters, skipping null values.

返回所有参数的最小值，跳过空值。

Examples:

例子:

```
> SELECT least(10, 9, 2, 4, 3);
 2

```

Since: 1.5.0

自: 1.5.0

left 左边

left(str, len) - Returns the leftmost `len`(`len` can be string type) characters from the string `str`,if `len` is less or equal than 0 the result is an empty string.

Left (str，len)-从字符串 str 返回最左边的 len (len 可以是字符串类型)字符，如果 len 小于或等于0，则结果为空字符串。

Examples:

例子:

```
> SELECT left('Spark SQL', 3);
 Spa

```

Since: 2.3.0

自: 2.3.0

length 长度

length(expr) - Returns the character length of string data or number of bytes of binary data. The length of string data includes the trailing spaces. The length of binary data includes binary zeros.

返回字符串数据的字符长度或二进制数据的字节数。字符串数据的长度包括尾随空格。二进制数据的长度包括二进制零。

Examples:

例子:

```
> SELECT length('Spark SQL ');
 10
> SELECT CHAR_LENGTH('Spark SQL ');
 10
> SELECT CHARACTER_LENGTH('Spark SQL ');
 10
```

```
7
```

Since: 1.5.0
自: 1.5.0

levenshtein 莱文什坦

levenshtein(str1, str2) - Returns the Levenshtein distance between the two given strings.
返回给定字符串之间的莱文斯坦距离。
Examples:
例子:

```
> SELECT levenshtein('kitten', 'sitting');
 3
```

Since: 1.5.0
自: 1.5.0

like 喜欢

str like pattern[ESCAPE escape] - Returns true if str matches `pattern` with `escape`, null if any arguments are null, false otherwise.
如果 str 与 ESCAPE 匹配 pattern，返回 true; 如果有任何参数为 null，返回 null; 否则返回 false。
Arguments:
论点:

- str - a string expression Str-a 字符串表达式
- pattern - a string expression. The pattern is a string which is matched literally, with exception to the following special symbols:Pattern-a 字符串表达式。这个模式是一个字符串，字面上匹配，除了下面的特殊符号:_ matches any one character in the input (similar to . in posix regular expressions)_ 匹配输入中的任何一个字符(类似于 posix 正则表达式中的.)% matches zero or more characters in the input (similar to .* in posix regular expressions)% 匹配输入中的零个或多个字符(类似于 posix 正则表达式中的. *)Since Spark 2.0, string literals are unescaped in our SQL parser. For example, in order to match "\abc", the pattern should be "\abc".自 Spark 2.0以来，在我们的 SQL 解析器中字符串是非转义的。例如，为了匹配“ abc”，模式应该是“ abc”。When SQL config 'spark.sql.parser.escapedStringLiterals' is enabled, it falls back to Spark 1.6 behavior regarding string literal parsing. For example, if the config is enabled, the pattern to match "\abc" should be "\abc". * escape - an character added since Spark 3.0. The default escape character is the '\'. If an escape character precedes a special symbol or another escape character, the following character is matched literally. It is invalid to escape

any other character.当 SQL 配置‘ Spark.SQL.parser.escapedstringliterals’被启用时，它又回到了与字符串文字解析相关的 Spark 1.6行为。例如，如果启用了配置，匹配“ abc”的模式应该是“ abc”。* escape ——自 Spark 3.0以来添加的字符。默认的转义字符是”。如果转义字符位于特殊符号或其他转义字符之前，则按字面匹配以下字符。转义任何其他字符都是无效的。

Examples:

例子:

```
> SELECT like('Spark', '_park');
true
> SET spark.sql.parser.escapedStringLiterals=true;
spark.sql.parser.escapedStringLiterals  true
> SELECT '%SystemDrive%\Users\John' like '\%SystemDrive\%\\Users%';
true
> SET spark.sql.parser.escapedStringLiterals=false;
spark.sql.parser.escapedStringLiterals  false
> SELECT '%SystemDrive%\\Users\\John' like '\%SystemDrive\%\\\\Users%';
true
> SELECT '%SystemDrive%/Users/John' like '/%SystemDrive/%//Users%' ESCAPE '/';
true

```

Note:

注意:

Use RLIKE to match with standard regular expressions.

使用 RLIKE 来匹配标准正则表达式。

Since: 1.0.0

自: 1.0.0

ln 进

ln(expr) - Returns the natural logarithm (base e) of `expr`.

返回 expr 的自然对数(base e)。

Examples:

例子:

```
> SELECT ln(1);
 0.0

```

Since: 1.4.0

自: 1.4.0

locate 定位

locate(substr, str[, pos]) - Returns the position of the first occurrence of `substr` in `str` after position `pos`. The given `pos` and return value are 1-based.

Locate (substr，str [， pos])-返回位置 pos 后在 str 中第一次出现的位置。给定的 pos 和返回值是基于1的。

Examples:

例子:

```
> SELECT locate('bar', 'foobarbar');
 4
> SELECT locate('bar', 'foobarbar', 5);
 7
> SELECT POSITION('bar' IN 'foobarbar');
 4

```

Since: 1.5.0

自: 1.5.0

log 原木

log(base, expr) - Returns the logarithm of `expr` with `base`.

Log (base，expr)-返回 expr 的带底对数。

Examples:

例子:

```
> SELECT log(10, 100);
 2.0

```

Since: 1.5.0

自: 1.5.0

log10

log10(expr) - Returns the logarithm of `expr` with base 10.

Log10(expr)-返回以10为底的 expr 的对数。

Examples:

例子:

```
> SELECT log10(10);
 1.0

```

Since: 1.4.0

自: 1.4.0

log1p 对数

log1p(expr) - Returns log(1 + `expr`).

Log1p (expr)-Returns log (1 + expr).

Examples:

例子:

```
> SELECT log1p(0);
 0.0

```

Since: 1.4.0

自: 1.4.0

log2

log2(expr) - Returns the logarithm of `expr` with base 2.

Log2(expr)-返回以2为底的 expr 的对数。

Examples:

例子:

```
> SELECT log2(2);
 1.0

```

Since: 1.4.0

自: 1.4.0

lower 下

lower(str) - Returns `str` with all characters changed to lowercase.

Lower (str)-返回所有字符都改为小写的 str。

Examples:

例子:

```
> SELECT lower('SparkSql');
 sparksql

```

Since: 1.0.1

自: 1.0.1

lpad 发射台

lpad(str, len[, pad]) - Returns `str`, left-padded with `pad` to a length of `len`. If `str` is longer than `len`, the return value is shortened to `len` characters. If `pad` is not specified, `str` will be padded to the left with space characters.

Lpad (str，len [，pad])-返回 str，左填充垫到一个长度的 len。如果 str 长于 len，则返回值缩短为 len 字符。如果没有指定填充区，str 将用空格字符填充到左侧。

Examples:

例子:

```
> SELECT lpad('hi', 5, '??');
 ???hi
> SELECT lpad('hi', 1, '??');
 h
> SELECT lpad('hi', 5);
    hi

```

Since: 1.5.0

自: 1.5.0

ltrim

ltrim(str) - Removes the leading space characters from `str`.

Ltrim (str)-从 str 中删除前导空格字符。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- trimStr - the trim string characters to trim, the default value is a single space
- trimStr ——要修剪的修剪字符串字符，默认值是一个空格

Examples:

例子:

```
> SELECT ltrim('    SparkSQL   ');
 SparkSQL

```

Since: 1.5.0

自: 1.5.0

make_date 约会

make_date(year, month, day) - Create date from year, month and day fields.

Make _ date (year，month，day)-从年、月和日字段创建 date。

Arguments:

论点:

- year - the year to represent, from 1 to 9999
- 年份-代表的年份，从1到9999
- month - the month-of-year to represent, from 1 (January) to 12 (December)
- 1月1日至12月12日
- day - the day-of-month to represent, from 1 to 31
- 每月的第一天到第三十一天

Examples:

例子:

```
> SELECT make_date(2013, 7, 15);
 2013-07-15
> SELECT make_date(2019, 13, 1);
 NULL
> SELECT make_date(2019, 7, NULL);
 NULL
> SELECT make_date(2019, 2, 30);
 NULL

```

Since: 3.0.0
自: 3.0.0

make_dt_interval 使 _ dt _ interval

make_dt_interval([days[, hours[, mins[, secs]]]]) - Make DayTimeIntervalType duration from days, hours, mins and secs.
Make _ dt _ interval ([days [， hours [， mins [， secs]]])-将 DayTimeIntervalType 持续时间由天、小时、分钟和秒组成。
Arguments:
论点:

- days - the number of days, positive or negative
- 天数-天数，正数或负数
- hours - the number of hours, positive or negative
- 小时数-小时数，正数或负数
- mins - the number of minutes, positive or negative
- 分钟-分钟数，正的还是负的
- secs - the number of seconds with the fractional part in microsecond precision.
- 秒-秒数，以微秒精度表示小数部分。

Examples:
例子:

```
> SELECT make_dt_interval(1, 12, 30, 01.001001);
 1 12:30:01.001001000
> SELECT make_dt_interval(2);
 2 00:00:00.000000000
> SELECT make_dt_interval(100, null, 3);
 NULL

```

Since: 3.2.0
3.2.0

make_interval 制造间隔

make_interval([years[, months[, weeks[, days[, hours[, mins[, secs]]]]]]]) - Make interval from years, months, weeks, days, hours, mins and secs.

间隔([years [， months [， weeks [， days [， hours [， mins [， secs]]]])-间隔由年、月、周、日、小时、分钟和秒构成。

Arguments:

论点:

- years - the number of years, positive or negative
- 年数-年数，正数或负数
- months - the number of months, positive or negative
- 月份-月数，正数或负数
- weeks - the number of weeks, positive or negative
- 周数-周数，正数还是负数
- days - the number of days, positive or negative
- 天数-天数，正数或负数
- hours - the number of hours, positive or negative
- 小时数-小时数，正数或负数
- mins - the number of minutes, positive or negative
- 分钟-分钟数，正的还是负的
- secs - the number of seconds with the fractional part in microsecond precision.
- 秒-秒数，以微秒精度表示小数部分。

Examples:

例子:

```
> SELECT make_interval(100, 11, 1, 1, 12, 30, 01.001001);
 100 years 11 months 8 days 12 hours 30 minutes 1.001001 seconds
> SELECT make_interval(100, null, 3);
 NULL
> SELECT make_interval(0, 1, 0, 1, 0, 0, 100.000001);
 1 months 1 days 1 minutes 40.000001 seconds

```

Since: 3.0.0

自: 3.0.0

make_timestamp 时间戳

make_timestamp(year, month, day, hour, min, sec[, timezone]) - Create timestamp from year, month, day, hour, min, sec and timezone fields. The result data type is consistent with the value of configuration `spark.sql.timestampType`

Make _ timestamp (year，month，day，hour，min，sec [， timezone])-根据年、月、日、小时、 min、 sec 和 timezone 字段创建时间戳。结果数据类型与配置 spark.sql.timestampType 的值一致

Arguments:

论点:

- year - the year to represent, from 1 to 9999
- 年份-代表的年份，从1到9999
- month - the month-of-year to represent, from 1 (January) to 12 (December)
- 1月1日至12月12日
- day - the day-of-month to represent, from 1 to 31
- 每月的第一天到第三十一天
- hour - the hour-of-day to represent, from 0 to 23
- 一小时，一天中的一小时，从0到23
- min - the minute-of-hour to represent, from 0 to 59
- 分钟-小时的分钟，从0到59
- sec - the second-of-minute and its micro-fraction to represent, from 0 to 60. The value can be either an integer like 13 , or a fraction like 13.123. If the sec argument equals to 60, the seconds field is set to 0 and 1 minute is added to the final timestamp.
- 秒-秒和它的微分表示，从0到60。这个值可以是像13这样的整数，也可以是像13.123这样的分数。如果 sec 参数等于60，则 seconds 字段设置为0，并将1分钟添加到最终时间戳。
- timezone - the time zone identifier. For example, CET, UTC and etc.
- 时区-时区标识符。例如，CET，UTC 等等。

Examples:

例子:

```
> SELECT make_timestamp(2014, 12, 28, 6, 30, 45.887);
 2014-12-28 06:30:45.887
> SELECT make_timestamp(2014, 12, 28, 6, 30, 45.887, 'CET');
 2014-12-27 21:30:45.887
> SELECT make_timestamp(2019, 6, 30, 23, 59, 60);
 2019-07-01 00:00:00
> SELECT make_timestamp(2019, 6, 30, 23, 59, 1);
 2019-06-30 23:59:01
> SELECT make_timestamp(2019, 13, 1, 10, 11, 12, 'PST');
 NULL
> SELECT make_timestamp(null, 7, 22, 15, 30, 0);
 NULL

```

Since: 3.0.0
自: 3.0.0

make_ym_interval 使 ym 间隔

make_ym_interval([years[, months]]) - Make year-month interval from years, months.
Make _ ym _ interval ([years [， months])-使年月间隔从年月开始。
Arguments:
论点:

- years - the number of years, positive or negative
- 年数-年数，正数或负数
- months - the number of months, positive or negative
- 月份-月数，正数或负数

Examples:
例子:

```
> SELECT make_ym_interval(1, 2);
 1-2
> SELECT make_ym_interval(1, 0);
 1-0
> SELECT make_ym_interval(-1, 1);
 -0-11
> SELECT make_ym_interval(2);
 2-0

```

Since: 3.2.0
3.2.0

map 地图

map(key0, value0, key1, value1, ...) - Creates a map with the given key/value pairs.
Map (key0，value0，key1，value1，...)-使用给定的键/值对创建映射。
Examples:
例子:

```
> SELECT map(1.0, '2', 3.0, '4');
 {1.0:"2",3.0:"4"}

```

Since: 2.0.0
自: 2.0.0

map_concat 地图集合

map_concat(map, ...) - Returns the union of all the given maps
Map _ concat (map，...)-返回所有给定映射的联合
Examples:
例子:

```
> SELECT map_concat(map(1, 'a', 2, 'b'), map(3, 'c'));
 {1:"a",2:"b",3:"c"}

```

Since: 2.4.0
自: 2.4.0

map_entries 地图条目

map_entries(map) - Returns an unordered array of all entries in the given map.
Map _ entries (map)-返回给定 map 中所有条目的无序数组。
Examples:
例子:

```
> SELECT map_entries(map(1, 'a', 2, 'b'));
 [{"key":1,"value":"a"},{"key":2,"value":"b"}]

```

Since: 3.0.0
自: 3.0.0

map_filter 过滤器

map_filter(expr, func) - Filters entries in a map using the function.
Map _ filter (expr，func)-使用函数在 map 中筛选条目。
Examples:
例子:

```
> SELECT map_filter(map(1, 0, 2, 2, 3, -1), (k, v) -> k > v);
```

```
{1:0,3:-1}
```

Since: 3.0.0
自: 3.0.0

map_from_arrays 数组映射

map_from_arrays(keys, values) - Creates a map with a pair of the given key/value arrays. All elements in keys should not be null
Map _ from _ arrays (key，values)——用一对给定的 key/value 数组创建一个 map。键中的所有元素不应为空
Examples:
例子:

```
> SELECT map_from_arrays(array(1.0, 3.0), array('2', '4'));
 {1.0:"2",3.0:"4"}
```

Since: 2.4.0
自: 2.4.0

map_from_entries 从条目中获取地图

map_from_entries(arrayOfEntries) - Returns a map created from the given array of entries.
Map _ from _ entries (arrayOfEntries)——返回从给定的条目数组创建的映射。
Examples:
例子:

```
> SELECT map_from_entries(array(struct(1, 'a'), struct(2, 'b')));
 {1:"a",2:"b"}
```

Since: 2.4.0
自: 2.4.0

map_keys 地图键

map_keys(map) - Returns an unordered array containing the keys of the map.

Map _ keys (map)-返回一个包含 map 键的无序数组。

Examples:

例子:

```
> SELECT map_keys(map(1, 'a', 2, 'b'));
 [1,2]

```

Since: 2.0.0

自: 2.0.0

map_values 地图值

map_values(map) - Returns an unordered array containing the values of the map.

Map _ values (map)-返回一个包含 map 值的无序数组。

Examples:

例子:

```
> SELECT map_values(map(1, 'a', 2, 'b'));
 ["a","b"]

```

Since: 2.0.0

自: 2.0.0

map_zip_with 地图邮编

map_zip_with(map1, map2, function) - Merges two given maps into a single map by applying function to the pair of values with the same key. For keys only presented in one map, NULL will be passed as the value for the missing key. If an input map contains duplicated keys, only the first entry of the duplicated key is passed into the lambda function.

Map _ zip with (map1，map2，function)-通过对具有相同键的值对应用函数，将两个给定映射合并为一个映射。对于只在一个映射中显示的键，NULL 将作为缺少的键的值传递。如果输入映射包含重复键，则只有重复键的第一个条目被传递到 lambda 函数中。

Examples:

例子:

```
> SELECT map_zip_with(map(1, 'a', 2, 'b'), map(1, 'x', 2, 'y'), (k, v1, v2) ->
concat(v1, v2));
 {1:"ax",2:"by"}
```

```
3
```

Since: 3.0.0

自: 3.0.0

max 麦克斯

max(expr) - Returns the maximum value of `expr`.

Max (expr)-返回 expr 的最大值。

Examples:

例子:

```
> SELECT max(col) FROM VALUES (10), (50), (20) AS tab(col);
 50

```

Since: 1.0.0

自: 1.0.0

max_by 麦克斯比

max_by(x, y) - Returns the value of `x` associated with the maximum value of `y`.

Max _ by (x，y)-返回与 y 的最大值关联的 x 的值。

Examples:

例子:

```
> SELECT max_by(x, y) FROM VALUES (('a', 10)), (('b', 50)), (('c', 20)) AS tab(x, y);
 b

```

Since: 3.0.0

自: 3.0.0

md5

md5(expr) - Returns an MD5 128-bit checksum as a hex string of `expr`.

MD5(expr)-以 expr 的十六进制字符串形式返回 md5128位校验和。

Examples:

例子:

```
> SELECT md5('Spark');
 8cde774d6f7333752ed72cacddb05126

```

Since: 1.5.0

自: 1.5.0

mean 平均

mean(expr) - Returns the mean calculated from values of a group.

Mean (expr)-返回从组的值计算出的平均值。

Examples:

例子:

```
> SELECT mean(col) FROM VALUES (1), (2), (3) AS tab(col);
 2.0
> SELECT mean(col) FROM VALUES (1), (2), (NULL) AS tab(col);
 1.5

```

Since: 1.0.0

自: 1.0.0

min 分钟

min(expr) - Returns the minimum value of `expr`.

Min (expr)-返回 expr 的最小值。

Examples:

例子:

```
> SELECT min(col) FROM VALUES (10), (-1), (20) AS tab(col);
 -1

```

Since: 1.0.0

自: 1.0.0

min_by 明比

min_by(x, y) - Returns the value of `x` associated with the minimum value of `y`.
Min _ by (x，y)-返回与 y 的最小值关联的 x 的值。
Examples:
例子:

```
> SELECT min_by(x, y) FROM VALUES (('a', 10)), (('b', 50)), (('c', 20)) AS tab(x, y);
 a

```

Since: 3.0.0
自: 3.0.0

minute 一分钟

minute(timestamp) - Returns the minute component of the string/timestamp.
Minute (时间戳)-返回字符串/时间戳的 minute 组件。
Examples:
例子:

```
> SELECT minute('2009-07-30 12:58:59');
 58

```

Since: 1.5.0
自: 1.5.0

mod 国防部

expr1 mod expr2 - Returns the remainder after `expr1`/`expr2`.
Expr1/expr2后返回余数。
Examples:
例子:

```
> SELECT 2 % 1.8;
 0.2
> SELECT MOD(2, 1.8);
 0.2

```

Since: 1.0.0

自: 1.0.0

monotonically_increasing_id 单调递增 id

monotonically_increasing_id() - Returns monotonically increasing 64-bit integers. The generated ID is guaranteed to be monotonically increasing and unique, but not consecutive. The current implementation puts the partition ID in the upper 31 bits, and the lower 33 bits represent the record number within each partition. The assumption is that the data frame has less than 1 billion partitions, and each partition has less than 8 billion records. The function is non-deterministic because its result depends on partition IDs.

_ id ()-返回64位整数的单调递增。生成的 ID 保证是单调递增且唯一的，但不是连续的。当前的实现将分区 ID 放在上面的31位中，下面的33位表示每个分区中的记录编号。假设数据帧的分区少于10亿个，每个分区的记录少于80亿条。该函数是不确定的，因为其结果依赖于分区 id。

Examples:

例子:

```
> SELECT monotonically_increasing_id();
 0

```

Since: 1.4.0

自: 1.4.0

month 月

month(date) - Returns the month component of the date/timestamp.

返回日期/时间戳的月份部分。

Examples:

例子:

```
> SELECT month('2016-07-30');
 7

```

Since: 1.5.0

自: 1.5.0

months_between 两个月

months_between(timestamp1, timestamp2[, roundOff]) - If `timestamp1` is later than `timestamp2`, then the result is positive. If `timestamp1` and `timestamp2` are on the same day of month, or both are the last day of month, time of day will be ignored. Otherwise, the difference is calculated based on 31 days per month, and rounded to 8 digits unless roundOff=false.
如果 timestamp1比 timestamp2晚，那么结果是阳性的。如果 timestamp1和 timestamp2在一个月的同一天，或者两者都是一个月的最后一天，那么一天的时间将被忽略。否则，差额将根据每月31天计算，四舍五入到8位数，除非 roundOff = false。

Examples:

例子:

```
> SELECT months_between('1997-02-28 10:30:00', '1996-10-30');
 3.94959677
> SELECT months_between('1997-02-28 10:30:00', '1996-10-30', false);
 3.9495967741935485

```

Since: 1.5.0

自: 1.5.0

named_struct 命名为 _ struct

named_struct(name1, val1, name2, val2, ...) - Creates a struct with the given field names and values.

Named _ struct (name1，val1，name2，val2，...)-创建具有给定字段名和值的结构。

Examples:

例子:

```
> SELECT named_struct("a", 1, "b", 2, "c", 3);
 {"a":1,"b":2,"c":3}

```

Since: 1.5.0

自: 1.5.0

nanvl 纳瓦尔

nanvl(expr1, expr2) - Returns `expr1` if it's not NaN, or `expr2` otherwise.

Nanvl (expr1，expr2)-如果不是 NaN 则返回 expr1，否则返回 expr2。

Examples:

例子:

```
> SELECT nanvl(cast('NaN' as double), 123);
 123.0

```

Since: 1.5.0

自: 1.5.0

negative 否定的

negative(expr) - Returns the negated value of `expr`.

Negative (expr)-返回 expr 的负值。

Examples:

例子:

```
> SELECT negative(1);
 -1

```

Since: 1.0.0

自: 1.0.0

next_day 第二天

next_day(start_date, day_of_week) - Returns the first date which is later than `start_date` and named as indicated. The function returns NULL if at least one of the input parameters is NULL. When both of the input parameters are not NULL and day_of_week is an invalid input, the function throws IllegalArgumentException if `spark.sql.ansi.enabled` is set to true, otherwise NULL.

Next _ day (start _ date，day _ of _ week)-返回比 start _ date 晚的第一个日期，并按指示命名。如果至少有一个输入参数为 NULL，则函数返回 NULL。当两个输入参数都不是 NULL，并且 day _ of _ week 是无效的输入时，如果 spark.sql.ansi.enabled 设置为 true，则函数抛出 IllegalArgumentException，否则为 NULL。

Examples:

例子:

```
> SELECT next_day('2015-01-14', 'TU');
 2015-01-20
```

```
3
```

Since: 1.5.0

自: 1.5.0

not 不

not expr - Logical not.

没有扩展-逻辑上没有。

Examples:

例子:

```
> SELECT not true;
 false
> SELECT not false;
 true
> SELECT not NULL;
 NULL
```

Since: 1.0.0

自: 1.0.0

now 现在

now() - Returns the current timestamp at the start of query evaluation.

Now ()-返回查询计算开始时的当前时间戳。

Examples:

例子:

```
> SELECT now();
 2020-04-25 15:49:11.914
```

Since: 1.6.0

自: 1.6.0

nth_value N 值

nth_value(input[, offset]) - Returns the value of `input` at the row that is the `offset`th row from beginning of the window frame. Offset starts at 1. If ignoreNulls=true, we will skip nulls when finding the `offset`th row. Otherwise, every row counts for the `offset`. If there is no such an `offset`th row (e.g., when the offset is 10, size of the window frame is less than 10), null is returned.

Nth _ value (input [， offset])-返回输入行的值，该行是窗口框架开始处的 offsetth 行。偏移量从1开始。如果 ignoreNulls = true，则在查找 offsetth 行时将跳过 nulls。否则，每一行都计算偏移量。如果没有这样的偏移量行(例如，当偏移量为10时，窗口框架的大小小于10) ，则返回 null。

Arguments:

论点:

- input - the target column or expression that the function operates on.
- Input ——函数操作的目标列或表达式。
- offset - a positive int literal to indicate the offset in the window frame. It starts with 1.
- Offset-一个正整型数字，用于指示窗口框架中的偏移量。
- ignoreNulls - an optional specification that indicates the NthValue should skip null values in the determination of which row to use.
- Ignorrenulls ——一个可选的规范，指出在确定使用哪一行时 NthValue 应跳过空值。

Examples:

例子:

```
> SELECT a, b, nth_value(b, 2) OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   1
 A1 1   1
 A1 2   1
 A2 3   NULL

```

Since: 3.1.0

3.1.0

ntile

ntile(n) - Divides the rows for each window partition into `n` buckets ranging from 1 to at most `n`.

Ntile (n)-将每个窗口分区的行分成 n 个桶，范围从1到最多 n。

Arguments:

论点:

- buckets - an int expression which is number of buckets to divide the rows in. Default value is 1.
- Bucket-一个 int 表达式，它是用于除行的 bucket 数。默认值为1。

Examples:

例子:

```
> SELECT a, b, ntile(2) OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   1
 A1 1   1
 A1 2   2
 A2 3   1

```

Since: 2.0.0

自: 2.0.0

nullif 废弃物

nullif(expr1, expr2) - Returns null if `expr1` equals to `expr2`, or `expr1` otherwise.

Nullf1(expr1，expr2)-如果 expr1等于 expr2，返回 null，否则返回 expr1。

Examples:

例子:

```
> SELECT nullif(2, 2);
 NULL

```

Since: 2.0.0

自: 2.0.0

nvl 就在那个时候

nvl(expr1, expr2) - Returns `expr2` if `expr1` is null, or `expr1` otherwise.

Nvl (expr1，expr2)-如果 expr1为 null，返回 expr2，否则返回 expr1。

Examples:

例子:

```
> SELECT nvl(NULL, array('2'));
 ["2"]

```

Since: 2.0.0
自: 2.0.0

nvl2

nvl2(expr1, expr2, expr3) - Returns `expr2` if `expr1` is not null, or `expr3` otherwise.
Nvl2(expr1，expr2，expr3)-如果 expr1不为 null，返回 expr2，否则返回 expr3。
Examples:
例子:

```
> SELECT nvl2(NULL, 2, 1);
 1

```

Since: 2.0.0
自: 2.0.0

octet_length 八位组长度

octet_length(expr) - Returns the byte length of string data or number of bytes of binary data.
返回字符串数据的字节长度或二进制数据的字节数。
Examples:
例子:

```
> SELECT octet_length('Spark SQL');
 9

```

Since: 2.3.0
自: 2.3.0

or 或

expr1 or expr2 - Logical OR.
Expr1或 expr2- 逻辑或。
Examples:
例子:

```
> SELECT true or false;
```

```
 true
> SELECT false or false;
 false
> SELECT true or NULL;
 true
> SELECT false or NULL;
 NULL

```

Since: 1.0.0

自: 1.0.0

overlay 覆盖

overlay(input, replace, pos[, len]) - Replace `input` with `replace` that starts at `pos` and is of length `len`.

覆盖(输入，替换，pos [，len])-将输入替换为起始于 pos，长度为 len 的替换。

Examples:

例子:

```
> SELECT overlay('Spark SQL' PLACING '_' FROM 6);
 Spark_SQL
> SELECT overlay('Spark SQL' PLACING 'CORE' FROM 7);
 Spark CORE
> SELECT overlay('Spark SQL' PLACING 'ANSI ' FROM 7 FOR 0);
 Spark ANSI SQL
> SELECT overlay('Spark SQL' PLACING 'tructured' FROM 2 FOR 4);
 Structured SQL
> SELECT overlay(encode('Spark SQL', 'utf-8') PLACING encode('_', 'utf-8') FROM 6);
 Spark_SQL
> SELECT overlay(encode('Spark SQL', 'utf-8') PLACING encode('CORE', 'utf-8') FROM 7);
 Spark CORE
> SELECT overlay(encode('Spark SQL', 'utf-8') PLACING encode('ANSI ', 'utf-8') FROM 7
FOR 0);
 Spark ANSI SQL
> SELECT overlay(encode('Spark SQL', 'utf-8') PLACING encode('tructured', 'utf-8') FROM
2 FOR 4);
 Structured SQL

```

Since: 3.0.0

自: 3.0.0

parse_url 解析 url

parse_url(url, partToExtract[, key]) - Extracts a part from a URL.

Parse _ URL (URL，partToExtract [， key])-从 URL 中提取部分内容。

Examples:

例子:

```
> SELECT parse_url('http://spark.apache.org/path?query=1', 'HOST');
 spark.apache.org
> SELECT parse_url('http://spark.apache.org/path?query=1', 'QUERY');
 query=1
> SELECT parse_url('http://spark.apache.org/path?query=1', 'QUERY', 'query');
 1

```

Since: 2.0.0

自: 2.0.0

percent_rank 百分比等级

percent_rank() - Computes the percentage ranking of a value in a group of values.

% _ rank ()-计算一组值中某个值的百分比排序。

Arguments:

论点:

- children - this is to base the rank on; a change in the value of one the children will trigger a change in rank. This is an internal parameter and will be assigned by the Analyser.
- 子女——这是基于排名; 一个子女的值的变化将引发排名的变化。这是一个内部参数，将由分析器分配。

Examples:

例子:

```
> SELECT a, b, percent_rank(b) OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   0.0
```

```
A1 1   0.0
A1 2   1.0
A2 3   0.0
```

Since: 2.0.0
自: 2.0.0

percentile 百分位数

percentile(col, percentage [, frequency]) - Returns the exact percentile value of numeric column `col` at the given percentage. The value of percentage must be between 0.0 and 1.0. The value of frequency should be positive integral
百分位数(col，percentage [， frequency])——返回给定百分比的数字列 col 的精确百分位数值。百分比值必须介于0.0和1.0之间。频率的值应为正积分
percentile(col, array(percentage1 [, percentage2]...) [, frequency]) - Returns the exact percentile value array of numeric column `col` at the given percentage(s). Each value of the percentage array must be between 0.0 and 1.0. The value of frequency should be positive integral
Percentile (col，array (percentage1[， percentage2] ...)[， frequency])-返回给定百分比的数字列 col 的精确百分位值数组。百分比数组的每个值必须介于0.0和1.0之间。频率的值应为正积分
Examples:
例子:

```
> SELECT percentile(col, 0.3) FROM VALUES (0), (10) AS tab(col);
 3.0
> SELECT percentile(col, array(0.25, 0.75)) FROM VALUES (0), (10) AS tab(col);
 [2.5,7.5]
```

Since: 2.1.0
自: 2.1.0

percentile_approx 百分比约数

percentile_approx(col, percentage [, accuracy]) - Returns the approximate `percentile` of the numeric column `col` which is the smallest value in the ordered `col` values (sorted from least to greatest) such that no more than `percentage` of `col` values is less than the value or equal to that value. The value of percentage must be between 0.0 and 1.0. The `accuracy` parameter (default:

10000) is a positive numeric literal which controls approximation accuracy at the cost of memory. Higher value of `accuracy` yields better accuracy, `1.0/accuracy` is the relative error of the approximation. When `percentage` is an array, each value of the percentage array must be between 0.0 and 1.0. In this case, returns the approximate percentile array of column `col` at the given percentage array.

Percentile _ approx (col，percentage [， accuracy])——返回数字列 col 的近似百分位数，它是有序 col 值中的最小值(从最小到最大排序)，以便 col 值的百分比不小于该值或等于该值。百分比值必须介于0.0和1.0之间。精度参数(默认值: 10000)是一个正数值，它以内存为代价来控制近似精度。较高的精度值产生较好的精度，1.0/精度是近似值的相对误差。当 percentage 为数组时，百分比数组的每个值必须介于0.0和1.0之间。在本例中，返回给定百分比数组中列 col 的近似百分位数组。

Examples:

例子:

```
> SELECT percentile_approx(col, array(0.5, 0.4, 0.1), 100) FROM VALUES (0), (1), (2), (10) AS tab(col);
 [1,1,0]
> SELECT percentile_approx(col, 0.5, 100) FROM VALUES (0), (6), (7), (9), (10) AS tab(col);
 7
```

Since: 2.1.0

自: 2.1.0

pi 圆周率

pi() - Returns pi.

Π ()-返回 π。

Examples:

例子:

```
> SELECT pi();
 3.141592653589793
```

Since: 1.5.0

自: 1.5.0

pmod

pmod(expr1, expr2) - Returns the positive value of `expr1` mod `expr2`.
Pmod (expr1，expr2)-返回 expr1 mod expr2的正值。
Examples:
例子:

```
> SELECT pmod(10, 3);
 1
> SELECT pmod(-10, 3);
 2

```

Since: 1.5.0
自: 1.5.0

posexplode 后爆炸

posexplode(expr) - Separates the elements of array `expr` into multiple rows with positions, or the elements of map `expr` into multiple rows and columns with positions. Unless specified otherwise, uses the column name `pos` for position, `col` for elements of the array or `key` and `value` for elements of the map.
将阵列扩展的元素分隔成多个带有位置的行，或者将映射扩展的元素分隔成多个带有位置的行和列。除非另有说明，否则使用列名 pos 表示位置、 col 表示数组的元素或 key 和值表示映射的元素。
Examples:
例子:

```
> SELECT posexplode(array(10,20));
 0  10
 1  20

```

Since: 2.0.0
自: 2.0.0

posexplode_outer 外部爆炸

posexplode_outer(expr) - Separates the elements of array `expr` into multiple rows with positions, or the elements of map `expr` into multiple rows and columns with positions. Unless specified otherwise, uses the column name `pos` for position, `col` for elements of the array or `key` and `value` for elements of the map.

将阵列扩展的元素分成多个带有位置的行，或者将映射扩展的元素分成多个带有位置的行和列。除非另有说明，否则使用列名 pos 表示位置、 col 表示数组的元素或 key 和值表示映射的元素。

Examples:

例子:

```
> SELECT posexplode_outer(array(10,20));
 0  10
 1  20

```

Since: 2.0.0

自: 2.0.0

position 位置

position(substr, str[, pos]) - Returns the position of the first occurrence of `substr` in `str` after position `pos`. The given `pos` and return value are 1-based.

位置(substr，str [， pos])——返回位置 pos 后在 str 中第一次出现的位置。给定的 pos 和返回值是基于1的。

Examples:

例子:

```
> SELECT position('bar', 'foobarbar');
 4
> SELECT position('bar', 'foobarbar', 5);
 7
> SELECT POSITION('bar' IN 'foobarbar');
 4

```

Since: 1.5.0

自: 1.5.0

positive 肯定的

positive(expr) - Returns the value of `expr`.

Positive (expr)-返回 expr 的值。

Examples:

例子:

```
> SELECT positive(1);
 1

```

Since: 1.5.0

自: 1.5.0

pow 战俘

pow(expr1, expr2) - Raises `expr1` to the power of `expr2`.

Pow (expr1，expr2)-提高 expr1到 expr2的威力。

Examples:

例子:

```
> SELECT pow(2, 3);
 8.0

```

Since: 1.4.0

自: 1.4.0

power 权力

power(expr1, expr2) - Raises `expr1` to the power of `expr2`.

能量(expr1，expr2)-提高 expr1到 expr2的能量。

Examples:

例子:

```
> SELECT power(2, 3);
 8.0

```

Since: 1.4.0

自: 1.4.0

printf

printf(strfmt, obj, ...) - Returns a formatted string from printf-style format strings.

Printf (strfmt，obj，...)-从 printf 样式的格式字符串返回格式化的字符串。

Examples:

例子:

```
> SELECT printf("Hello World %d %s", 100, "days");
 Hello World 100 days

```

Since: 1.5.0

自: 1.5.0

quarter 季度

quarter(date) - Returns the quarter of the year for date, in the range 1 to 4.

季度(日期)-返回一年中的季度中的日期，在1到4之间。

Examples:

例子:

```
> SELECT quarter('2016-08-31');
 3

```

Since: 1.5.0

自: 1.5.0

radians 弧度

radians(expr) - Converts degrees to radians.

弧度(expr)-将度转换为弧度。

Arguments:

论点:

- expr - angle in degrees 以度为单位的膨胀角

Examples:

例子:

```
> SELECT radians(180);
 3.141592653589793

```

Since: 1.4.0

自: 1.4.0

raise_error 引起误差

raise_error(expr) - Throws an exception with `expr`.

使用 expr 抛出异常。

Examples:

例子:

```
> SELECT raise_error('custom error message');
 java.lang.RuntimeException
 custom error message

```

Since: 3.1.0

3.1.0

rand 兰德

rand([seed]) - Returns a random value with independent and identically distributed (i.i.d.) uniformly distributed values in [0, 1).

Rand ([seed])-在[0,1]中返回一个具有独立和同分布(i.i.i.d)均匀分布值的随机值。

Examples:

例子:

```
> SELECT rand();
 0.9629742951434543
> SELECT rand(0);
 0.7604953758285915
> SELECT rand(null);
 0.7604953758285915

```

Note:

注意:

The function is non-deterministic in general case.

函数在一般情况下是不确定的。

Since: 1.5.0

自: 1.5.0

randn 兰登

randn([seed]) - Returns a random value with independent and identically distributed (i.i.d.) values drawn from the standard normal distribution.

返回一个从标准正态分布中提取的具有独立和同分布(i.i.d)值的随机值。

Examples:

例子:

```
> SELECT randn();
 -0.3254147983080288
> SELECT randn(0);
 1.6034991609278433
> SELECT randn(null);
 1.6034991609278433

```

Note:

注意:

The function is non-deterministic in general case.

函数在一般情况下是不确定的。

Since: 1.5.0

自: 1.5.0

random 随机的

random([seed]) - Returns a random value with independent and identically distributed (i.i.d.) uniformly distributed values in [0, 1).

随机([seed])-返回一个随机值，在[0,1]中具有独立的、同分布的(i.i.i.d)均匀分布的值。

Examples:

例子:

```
> SELECT random();
 0.9629742951434543
> SELECT random(0);
 0.7604953758285915
> SELECT random(null);
 0.7604953758285915

```

Note:
注意:
The function is non-deterministic in general case.
函数在一般情况下是不确定的。
Since: 1.5.0
自: 1.5.0

rank 等级

rank() - Computes the rank of a value in a group of values. The result is one plus the number of rows preceding or equal to the current row in the ordering of the partition. The values will produce gaps in the sequence.
Rank ()——计算一组值中某个值的秩。结果是在分区排序中，前面的行数加上当前行数，或等于当前行数。这些值将在序列中产生间隔。
Arguments:
论点:

- children - this is to base the rank on; a change in the value of one the children will trigger a change in rank. This is an internal parameter and will be assigned by the Analyser.
- 子女——这是基于排名; 一个子女的值的变化将引发排名的变化。这是一个内部参数，将由分析器分配。

Examples:
例子:

```
> SELECT a, b, rank(b) OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   1
 A1 1   1
 A1 2   3
 A2 3   1

```

Since: 2.0.0
自: 2.0.0

reflect 反映

reflect(class, method[, arg1[, arg2 ..]]) - Calls a method with reflection.
Reflect (class，method [，arg1[，arg2. .])-调用带有反射的方法。
Examples:

例子:

```
> SELECT reflect('java.util.UUID', 'randomUUID');
 c33fb387-8500-4bfa-81d2-6e0e3e930df2
> SELECT reflect('java.util.UUID', 'fromString', 'a5cf6c42-0c85-418f-af6c-3e4e5b1328f2');
 a5cf6c42-0c85-418f-af6c-3e4e5b1328f2

```

Since: 2.0.0

自: 2.0.0

regexp 进出口

regexp(str, regexp) - Returns true if `str` matches `regexp`, or false otherwise.

Regexp (str，regexp)-如果 str 匹配 regexp，则返回 true，否则返回 false。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- regexp - a string expression. The regex string should be a Java regular expression.Regex 字符串应该是一个 Java 正则表达式。Since Spark 2.0, string literals (including regex patterns) are unescaped in our SQL parser. For example, to match "\abc", a regular expression for `regexp` can be "^\abc$".自从 Spark 2.0以来，字符串文本(包括 regex 模式)在我们的 SQL 解析器中是非转义的。例如，要匹配“ abc”，regexp 的正则表达式可以是“ ^ abc $”。There is a SQL config 'spark.sql.parser.escapedStringLiterals' that can be used to fallback to the Spark 1.6 behavior regarding string literal parsing. For example, if the config is enabled, the `regexp` that can match "\abc" is "^\abc$".有一个 SQL 配置“ Spark.SQL.parser.escapedstringliterals”，可用于回退到与字符串文本解析有关的 Spark 1.6行为。例如，如果启用了配置，则可以匹配“ abc”的 regexp 为“ ^ abc $”。

Examples:

例子:

```
> SET spark.sql.parser.escapedStringLiterals=true;
spark.sql.parser.escapedStringLiterals  true
> SELECT regexp('%SystemDrive%\Users\John', '%SystemDrive%\\Users.*');
true
> SET spark.sql.parser.escapedStringLiterals=false;
spark.sql.parser.escapedStringLiterals  false
> SELECT regexp('%SystemDrive%\\Users\\John', '%SystemDrive%\\\\Users.*');
true

```

Note:

注意:

Use LIKE to match with simple string pattern.

使用 LIKE 来匹配简单的字符串模式。

Since: 3.2.0

3.2.0

regexp_extract Regexp extract

regexp_extract(str, regexp[, idx]) - Extract the first string in the `str` that match the `regexp` expression and corresponding to the regex group index.

Regexp _ Extract (str，regexp [，idx])-提取 str 中匹配 regexp 表达式并对应于 regex 组索引的第一个字符串。

Arguments:

论点:

- str - a string expression. Str-a 字符串表达式
- regexp - a string representing a regular expression. The regex string should be a Java regular expression.Regexp-表示正则表达式的字符串。正则表达式应该是 Java 正则表达式。Since Spark 2.0, string literals (including regex patterns) are unescaped in our SQL parser. For example, to match "\abc", a regular expression for `regexp` can be "^\abc$".自从 Spark 2.0以来，字符串文本(包括 regex 模式)在我们的 SQL 解析器中是非转义的。例如，要匹配“ abc”，regexp 的正则表达式可以是“ ^ abc $”。There is a SQL config 'spark.sql.parser.escapedStringLiterals' that can be used to fallback to the Spark 1.6 behavior regarding string literal parsing. For example, if the config is enabled, the `regexp` that can match "\abc" is "^\abc$". * idx - an integer expression that representing the group index. The regex maybe contains multiple groups. `idx` indicates which regex group to extract. The group index should be non-negative. The minimum value of `idx` is 0, which means matching the entire regular expression. If `idx` is not specified, the default group index value is 1. The `idx` parameter is the Java regex Matcher group() method index.有一个 SQL 配置“ Spark.SQL.parser.escapedstringliterals”，可用于回退到与字符串文本解析有关的 Spark 1.6行为。例如，如果启用了配置，则可以匹配“ abc”的 regexp 为“ ^ abc $”。* idx-表示组索引的整数表达式。正则表达式可能包含多个组。Idx 表示要提取哪个正则表达式组。组指数应该是非负的。Idx 的最小值为0，这意味着匹配整个正则表达式。如果未指定 idx，则默认组索引值为1。Idx 参数是 javaregexmatcher 组()方法索引。

Examples:

例子:

```
> SELECT regexp_extract('100-200', '(\\d+)-(\\d+)', 1);
 100

```

Since: 1.5.0
自: 1.5.0

regexp_extract_all 提取所有

regexp_extract_all(str, regexp[, idx]) - Extract all strings in the `str` that match the `regexp` expression and corresponding to the regex group index.
Regexp _ Extract _ all (str，regexp [，idx])-提取 str 中匹配 regexp 表达式并对应于 regex 组索引的所有字符串。

Arguments:
论点:

- str - a string expression. Str-a 字符串表达式
- regexp - a string representing a regular expression. The regex string should be a Java regular expression.Regexp-表示正则表达式的字符串。正则表达式应该是 Java 正则表达式。Since Spark 2.0, string literals (including regex patterns) are unescaped in our SQL parser. For example, to match "\abc", a regular expression for `regexp` can be "^\abc$".自从 Spark 2.0以来，字符串文本(包括 regex 模式)在我们的 SQL 解析器中是非转义的。例如，要匹配“ abc”，regexp 的正则表达式可以是“ ^ abc $”。There is a SQL config 'spark.sql.parser.escapedStringLiterals' that can be used to fallback to the Spark 1.6 behavior regarding string literal parsing. For example, if the config is enabled, the `regexp` that can match "\abc" is "^\abc$". * idx - an integer expression that representing the group index. The regex may contains multiple groups. `idx` indicates which regex group to extract. The group index should be non-negative. The minimum value of `idx` is 0, which means matching the entire regular expression. If `idx` is not specified, the default group index value is 1. The `idx` parameter is the Java regex Matcher group() method index.有一个 SQL 配置“ Spark.SQL.parser.escapedstringliterals”，可用于回退到与字符串文本解析有关的 Spark 1.6行为。例如，如果启用了配置，则可以匹配“ abc”的 regexp 为“ ^ abc $”。* idx-表示组索引的整数表达式。正则表达式可以包含多个组。Idx 表示要提取哪个正则表达式组。组指数应该是非负的。Idx 的最小值为0，这意味着匹配整个正则表达式。如果未指定 idx，则默认组索引值为1。Idx 参数是 javaregexmatcher 组()方法索引。

Examples:
例子:

```
> SELECT regexp_extract_all('100-200, 300-400', '(\\d+)-(\\d+)', 1);
 ["100","300"]

```

Since: 3.1.0
3.1.0

regexp_like 类似于 regexp

regexp_like(str, regexp) - Returns true if `str` matches `regexp`, or false otherwise.

如果 str 匹配 regexp，则返回 true，否则返回 false。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- regexp - a string expression. The regex string should be a Java regular expression.Regex 字符串应该是一个 Java 正则表达式。Since Spark 2.0, string literals (including regex patterns) are unescaped in our SQL parser. For example, to match "\abc", a regular expression for `regexp` can be "^\abc$".自从 Spark 2.0以来，字符串文本(包括 regex 模式)在我们的 SQL 解析器中是非转义的。例如，要匹配“ abc”，regexp 的正则表达式可以是“ ^ abc $”。There is a SQL config 'spark.sql.parser.escapedStringLiterals' that can be used to fallback to the Spark 1.6 behavior regarding string literal parsing. For example, if the config is enabled, the `regexp` that can match "\abc" is "^\abc$".有一个 SQL 配置“ Spark.SQL.parser.escapedstringliterals”，可用于回退到与字符串文本解析有关的 Spark 1.6行为。例如，如果启用了配置，则可以匹配“ abc”的 regexp 为“ ^ abc $”。

Examples:

例子:

```
> SET spark.sql.parser.escapedStringLiterals=true;
spark.sql.parser.escapedStringLiterals  true
> SELECT regexp_like('%SystemDrive%\Users\John', '%SystemDrive%\\Users.*');
true
> SET spark.sql.parser.escapedStringLiterals=false;
spark.sql.parser.escapedStringLiterals  false
> SELECT regexp_like('%SystemDrive%\\Users\\John', '%SystemDrive%\\\\Users.*');
true

```

Note:

注意:

Use LIKE to match with simple string pattern.

使用 LIKE 来匹配简单的字符串模式。

Since: 3.2.0

3.2.0

regexp_replace 替换

regexp_replace(str, regexp, rep[, position]) - Replaces all substrings of `str` that match `regexp` with `rep`.

Regexp _ replace (str，regexp，rep [， position])-将所有匹配 regexp 的 str 子字符串替换为 rep。

Arguments:

论点:

- str - a string expression to search for a regular expression pattern match.
- Str-用于搜索正则表达式模式匹配的字符串表达式。
- regexp - a string representing a regular expression. The regex string should be a Java regular expression.Regexp-表示正则表达式的字符串。正则表达式应该是 Java 正则表达式。Since Spark 2.0, string literals (including regex patterns) are unescaped in our SQL parser. For example, to match "\abc", a regular expression for `regexp` can be "^\abc$".自从 Spark 2.0以来，字符串文本(包括 regex 模式)在我们的 SQL 解析器中是非转义的。例如，要匹配“ abc”，regexp 的正则表达式可以是“ ^ abc $”。There is a SQL config 'spark.sql.parser.escapedStringLiterals' that can be used to fallback to the Spark 1.6 behavior regarding string literal parsing. For example, if the config is enabled, the `regexp` that can match "\abc" is "^\abc$". * rep - a string expression to replace matched substrings. * position - a positive integer literal that indicates the position within `str` to begin searching. The default is 1. If position is greater than the number of characters in `str`, the result is `str`.有一个 SQL 配置“ Spark.SQL.parser.escapedstringliterals”，可用于回退到与字符串文本解析有关的 Spark 1.6行为。例如，如果启用了配置，则可以匹配“ abc”的 regexp 为“ ^ abc $”。Rep-用于替换匹配子字符串的字符串表达式。* position-一个正整数字面值，指示 str 内开始搜索的位置。默认值是1。如果位置大于 str 中的字符数，则结果为 str。

Examples:

例子:

```
> SELECT regexp_replace('100-200', '(\\d+)', 'num');
 num-num

```

Since: 1.5.0

自: 1.5.0

repeat 重复

repeat(str, n) - Returns the string which repeats the given string value n times.

Repeat (str，n)-返回重复给定字符串值 n 次的字符串。

Examples:

例子:

```
> SELECT repeat('123', 2);
 123123

```

Since: 1.5.0

自: 1.5.0

replace 替换

replace(str, search[, replace]) - Replaces all occurrences of `search` with `replace`.

Replace (str，search [，replace])-将所有出现的搜索替换为 replace。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- search - a string expression. If search is not found in str, str is returned unchanged.
- 搜索-一个字符串表达式。如果在 str 中没有找到搜索，str 将不变地返回。
- replace - a string expression. If replace is not specified or is an empty string, nothing replaces the string that is removed from str.
- Replace-a 字符串表达式。如果没有指定 replace 或者 replace 是一个空字符串，则不会替换从 str 中移除的字符串。

Examples:

例子:

```
> SELECT replace('ABCabc', 'abc', 'DEF');
 ABCDEF

```

Since: 2.3.0

自: 2.3.0

reverse 倒档

reverse(array) - Returns a reversed string or an array with reverse order of elements.

反向(数组)-返回一个反向字符串或数组与元素的反向顺序。

Examples:

例子:

```
> SELECT reverse('Spark SQL');
```

```
 LQS krapS
> SELECT reverse(array(2, 1, 4, 3));
 [3,4,1,2]

```

Note:

注意:

Reverse logic for arrays is available since 2.4.0.

从2.4.0开始，数组的反向逻辑就可以使用了。

Since: 1.5.0

自: 1.5.0

right 对

right(str, len) - Returns the rightmost `len`(`len` can be string type) characters from the string `str`,if `len` is less or equal than 0 the result is an empty string.

右(str，len)-返回字符串 str 的最右边的 len (len 可以是字符串类型)字符，如果 len 小于或等于0，结果是一个空字符串。

Examples:

例子:

```
> SELECT right('Spark SQL', 3);
 SQL

```

Since: 2.3.0

自: 2.3.0

rint 林特

rint(expr) - Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

Rint (expr)-返回在值上最接近参数并且等于数学整数的双精度值。

Examples:

例子:

```
> SELECT rint(12.3456);
 12.0

```

Since: 1.4.0

自: 1.4.0

rlike 同 “ r”

rlike(str, regexp) - Returns true if `str` matches `regexp`, or false otherwise.

Rlike (str，regexp)-如果 str 匹配 regexp，返回 true，否则返回 false。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- regexp - a string expression. The regex string should be a Java regular expression.Regex 字符串应该是一个 Java 正则表达式。Since Spark 2.0, string literals (including regex patterns) are unescaped in our SQL parser. For example, to match "\abc", a regular expression for `regexp` can be "^\abc$".自从 Spark 2.0以来，字符串文本(包括 regex 模式)在我们的 SQL 解析器中是非转义的。例如，要匹配“ abc”，regexp 的正则表达式可以是“ ^ abc $”。There is a SQL config 'spark.sql.parser.escapedStringLiterals' that can be used to fallback to the Spark 1.6 behavior regarding string literal parsing. For example, if the config is enabled, the `regexp` that can match "\abc" is "^\abc$".有一个 SQL 配置“ Spark.SQL.parser.escapedstringliterals”，可用于回退到与字符串文本解析有关的 Spark 1.6行为。例如，如果启用了配置，则可以匹配“ abc”的 regexp 为“ ^ abc $”。

Examples:

例子:

```
> SET spark.sql.parser.escapedStringLiterals=true;
spark.sql.parser.escapedStringLiterals  true
> SELECT rlike('%SystemDrive%\Users\John', '%SystemDrive%\\Users.*');
true
> SET spark.sql.parser.escapedStringLiterals=false;
spark.sql.parser.escapedStringLiterals  false
> SELECT rlike('%SystemDrive%\\Users\\John', '%SystemDrive%\\\\Users.*');
true

```

Note:

注意:

Use LIKE to match with simple string pattern.

使用 LIKE 来匹配简单的字符串模式。

Since: 1.0.0

自: 1.0.0

round 圆

round(expr, d) - Returns `expr` rounded to `d` decimal places using HALF_UP rounding mode.

舍入(expr，d)-使用 HALF _ up 舍入模式，返回舍入到小数点后 d 位的 expr。

Examples:

例子:

```
> SELECT round(2.5, 0);
 3

```

Since: 1.5.0

自: 1.5.0

row_number 行号

row_number() - Assigns a unique, sequential number to each row, starting with one, according to the ordering of rows within the window partition.

行号()-根据窗口分区内行的顺序，从一行开始，为每一行分配一个唯一的顺序编号。

Examples:

例子:

```
> SELECT a, b, row_number() OVER (PARTITION BY a ORDER BY b) FROM VALUES ('A1', 2), ('A1', 1), ('A2', 3), ('A1', 1) tab(a, b);
 A1 1   1
 A1 1   2
 A1 2   3
 A2 3   1

```

Since: 2.0.0

自: 2.0.0

rpad 无线电发射台

rpad(str, len[, pad]) - Returns `str`, right-padded with `pad` to a length of `len`. If `str` is longer than `len`, the return value is shortened to `len` characters. If `pad` is not specified, `str` will be padded to the right with space characters.

Rpad (str，len [， pad])-返回 str，右填充垫到一个长度的 len。如果 str 长于 len，则返回值缩短为 len 字符。如果没有指定填充区，str 将用空格字符填充到右侧。

Examples:

例子:

```
> SELECT rpad('hi', 5, '??');
 hi???
> SELECT rpad('hi', 1, '??');
 h
> SELECT rpad('hi', 5);
 hi

```

Since: 1.5.0

自: 1.5.0

rtrim 修剪

rtrim(str) - Removes the trailing space characters from `str`.

从 str 中移除尾随空格字符。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- trimStr - the trim string characters to trim, the default value is a single space
- trimStr ——要修剪的修剪字符串字符，默认值是一个空格

Examples:

例子:

```
> SELECT rtrim('    SparkSQL   ');
 SparkSQL

```

Since: 1.5.0

自: 1.5.0

schema_of_csv 的模式

schema_of_csv(csv[, options]) - Returns schema in the DDL format of CSV string.

Schema _ of _ CSV (CSV [， options])——返回 CSV 字符串的 DDL 格式的模式。

Examples:
例子:

```
> SELECT schema_of_csv('1,abc');
 STRUCT<`_c0`: INT, `_c1`: STRING>

```

Since: 3.0.0
自: 3.0.0

schema_of_json 的模式

schema_of_json(json[, options]) - Returns schema in the DDL format of JSON string.
Schema _ of _ JSON (JSON [， options])-返回 JSON 字符串的 DDL 格式的模式。
Examples:
例子:

```
> SELECT schema_of_json('[{"col":0}]');
 ARRAY<STRUCT<`col`: BIGINT>>
> SELECT schema_of_json('[{"col":01}]', map('allowNumericLeadingZeros', 'true'));
 ARRAY<STRUCT<`col`: BIGINT>>

```

Since: 2.4.0
自: 2.4.0

second 第二

second(timestamp) - Returns the second component of the string/timestamp.
Second (timestamp)-返回字符串/时间戳的第二个组件。
Examples:
例子:

```
> SELECT second('2009-07-30 12:58:59');
 59

```

Since: 1.5.0
自: 1.5.0

sentences 句子

sentences(str[, lang, country]) - Splits `str` into an array of array of words.

句子(str [， lang，country])-将 str 分成一系列单词。

Examples:

例子:

```
> SELECT sentences('Hi there! Good morning.');
 [["Hi","there"],["Good","morning"]]

```

Since: 2.0.0

自: 2.0.0

sequence 序列

sequence(start, stop, step) - Generates an array of elements from start to stop (inclusive), incrementing by step. The type of the returned elements is the same as the type of argument expressions.

序列(开始，停止，步骤)-生成一个从开始到停止的元素数组(包括) ，按步骤递增。返回元素的类型与参数表达式的类型相同。

Supported types are: byte, short, integer, long, date, timestamp.

支持的类型有: 字节、短、整数、长、日期、时间戳。

The start and stop expressions must resolve to the same type. If start and stop expressions resolve to the 'date' or 'timestamp' type then the step expression must resolve to the 'interval' or 'year-month interval' or 'day-time interval' type, otherwise to the same type as the start and stop expressions.

启动和停止表达式必须解析为相同的类型。如果启动和停止表达式解析为 “ date” 或 “ timestamp” 类型，那么步骤表达式必须解析为 “ interval” 或 “ year-month interval” 或 “ day-time interval” 类型，否则解析为与启动和停止表达式相同的类型。

Arguments:

论点:

- start - an expression. The start of the range.
- 开始-一个表达式。范围的开始。
- stop - an expression. The end the range (inclusive).
- Stop-an 表达式。结束范围(包含)。
- step - an optional expression. The step of the range. By default step is 1 if start is less than or equal to stop, otherwise -1. For the temporal sequences it's 1 day and -1 day respectively. If start is greater than stop then the

step must be negative, and vice versa.

- 步骤-可选表达式。范围的步骤。如果 start 小于或等于 stop，默认步骤为1，否则为 -1。时间序列分别为1天和-1天。如果开始大于停止，那么步骤必须是负的，反之亦然。

Examples:

例子:

```
> SELECT sequence(1, 5);
 [1,2,3,4,5]
> SELECT sequence(5, 1);
 [5,4,3,2,1]
> SELECT sequence(to_date('2018-01-01'), to_date('2018-03-01'), interval 1 month);
 [2018-01-01,2018-02-01,2018-03-01]
> SELECT sequence(to_date('2018-01-01'), to_date('2018-03-01'), interval '0-1' year to month);
 [2018-01-01,2018-02-01,2018-03-01]

```

Since: 2.4.0

自: 2.4.0

session_window 会话窗口

session_window(time_column, gap_duration) - Generates session window given a timestamp specifying column and gap duration. See 'Types of time windows' in Structured Streaming guide doc for detailed explanation and examples.

Session _ window (time _ column，gap _ duration)——给定一个指定列和间隔持续时间的时间戳，生成会话窗口。详细的解释和例子请参阅结构化串流指南文档中的“时间窗口类型”。

Arguments:

论点:

- time_column - The column or the expression to use as the timestamp for windowing by time. The time column must be of TimestampType.
- Time _ column ——按时间作为窗口时间戳的列或表达式。时间列必须是 TimestampType。
- gap_duration - A string specifying the timeout of the session represented as "interval value" (See Interval Literal for more details.) for the fixed gap duration, or an expression which is applied for each input and evaluated to the "interval value" for the dynamic gap duration.
- Gap _ duration ——指定会话的超时的字符串，表示为“ Interval value”(更多详细信息请参见 Interval Literal)对于固定间隙持续时间，或对于每个输入应用并对动态间隙持续时间的“间隔值”求值的表达式。

Examples:

例子:

```
> SELECT a, session_window.start, session_window.end, count(*) as cnt FROM VALUES ('A1', '2021-01-01 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:10:00'), ('A2', '2021-01-01 00:01:00') AS tab(a, b) GROUP by a, session_window(b, '5 minutes') ORDER BY a, start;
  A1    2021-01-01 00:00:00 2021-01-01 00:09:30 2
  A1    2021-01-01 00:10:00 2021-01-01 00:15:00 1
  A2    2021-01-01 00:01:00 2021-01-01 00:06:00 1
> SELECT a, session_window.start, session_window.end, count(*) as cnt FROM VALUES ('A1', '2021-01-01 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:10:00'), ('A2', '2021-01-01 00:01:00'), ('A2', '2021-01-01 00:04:30') AS tab(a, b) GROUP by a, session_window(b, CASE WHEN a = 'A1' THEN '5 minutes' WHEN a = 'A2' THEN '1 minute' ELSE '10 minutes' END) ORDER BY a, start;
  A1    2021-01-01 00:00:00 2021-01-01 00:09:30 2
  A1    2021-01-01 00:10:00 2021-01-01 00:15:00 1
  A2    2021-01-01 00:01:00 2021-01-01 00:02:00 1
  A2    2021-01-01 00:04:30 2021-01-01 00:05:30 1

```

Since: 3.2.0

3.2.0

sha 女名女子名

sha(expr) - Returns a sha1 hash value as a hex string of the `expr`.

返回一个 sha1散列值作为 expr 的十六进制字符串。

Examples:

例子:

```
> SELECT sha('Spark');
 85f5955f4b27a9a4c2aab6ffe5d7189fc298b92c

```

Since: 1.5.0

自: 1.5.0

sha1

sha1(expr) - Returns a sha1 hash value as a hex string of the `expr`.

返回一个 sha1散列值作为 expr 的十六进制字符串。

Examples:

例子:

```
> SELECT sha1('Spark');
 85f5955f4b27a9a4c2aab6ffe5d7189fc298b92c

```

Since: 1.5.0

自: 1.5.0

sha2

sha2(expr, bitLength) - Returns a checksum of SHA-2 family as a hex string of `expr`. SHA-224, SHA-256, SHA-384, and SHA-512 are supported. Bit length of 0 is equivalent to 256.

Sha2(expr，bitLength)-以 expr 的十六进制字符串形式返回 sha-2家族的校验和。支持 SHA-224、SHA-256、 sha-384和 SHA-512。位长度为0等于256。

Examples:

例子:

```
> SELECT sha2('Spark', 256);
 529bc3b07127ecb7e53a4dcf1991d9152c24537d919178022b2c42657f79a26b

```

Since: 1.5.0

自: 1.5.0

shiftleft 左转

shiftleft(base, expr) - Bitwise left shift.

移左(基地，扩展)-位左移。

Examples:

例子:

```
> SELECT shiftleft(2, 1);
 4

```

Since: 1.5.0

自: 1.5.0

shiftright 交通灯

shiftright(base, expr) - Bitwise (signed) right shift.

移位(基，扩展)-按位(符号)右移。

Examples:

例子:

```
> SELECT shiftright(4, 1);
 2

```

Since: 1.5.0

自: 1.5.0

shiftrightunsigned 史夫特里顿

shiftrightunsigned(base, expr) - Bitwise unsigned right shift.

Shiftrightunsigned (base，expr)-Bitwise unsigned 右移。

Examples:

例子:

```
> SELECT shiftrightunsigned(4, 1);
 2

```

Since: 1.5.0

自: 1.5.0

shuffle 洗牌

shuffle(array) - Returns a random permutation of the given array.

洗牌(数组)-返回给定数组的随机排列。

Examples:

例子:

```
> SELECT shuffle(array(1, 20, 3, 5));
 [3,1,5,20]
> SELECT shuffle(array(1, 20, null, 3));
 [20,null,3,1]

```

Note:

注意:

The function is non-deterministic.

这个函数是不确定的。

Since: 2.4.0

自: 2.4.0

sign 符号

sign(expr) - Returns -1.0, 0.0 or 1.0 as `expr` is negative, 0 or positive.

符号(expr)-return-1.0,0.0或1.0，因为 expr 是负数，0或正数。

Examples:

例子:

```
> SELECT sign(40);
 1.0

```

Since: 1.4.0

自: 1.4.0

signum 符号

signum(expr) - Returns -1.0, 0.0 or 1.0 as `expr` is negative, 0 or positive.

Signum (expr)-return-1.0,0.0或1.0 as expr 为负，0或正。

Examples:

例子:

```
> SELECT signum(40);
 1.0

```

Since: 1.4.0

自: 1.4.0

sin 罪

sin(expr) - Returns the sine of `expr`, as if computed by `java.lang.Math.sin`.

Sin (expr)-返回 expr 的 sine，就像是由 java.lang.Math.sin 计算一样。

Arguments:

论点:

- expr - angle in radians 弧度扩展角

Examples:

例子:

```
> SELECT sin(0);
 0.0

```

Since: 1.4.0

自: 1.4.0

sinh 没错

sinh(expr) - Returns hyperbolic sine of `expr`, as if computed by `java.lang.Math.sinh`.

返回 expr 的双曲正弦值，如同由 java.lang.Math.sinh 计算一样。

Arguments:

论点:

- expr - hyperbolic angle 双曲角

Examples:

例子:

```
> SELECT sinh(0);
 0.0

```

Since: 1.4.0

自: 1.4.0

size 大小

size(expr) - Returns the size of an array or a map. The function returns null for null input if spark.sql.legacy.sizeOfNull is set to false or spark.sql.ansi.enabled is set to true. Otherwise, the function returns -1 for null input. With the default settings, the function returns -1 for null input.

Size (expr)-返回数组或映射的大小。如果 spark.sql.legacy.sizeOfNull 设置为 false 或 spark.sql.ansi.enabled 设置为 true，那么该函数将对 null 输入返回 null。否则，函数返回 -1表示空输入。使用默认设置，函数返回 -1表示空输入。

Examples:

例子:

```
> SELECT size(array('b', 'd', 'c', 'a'));
 4
> SELECT size(map('a', 1, 'b', 2));
 2
> SELECT size(NULL);
 -1
```

Since: 1.5.0
自: 1.5.0

skewness 斜度

skewness(expr) - Returns the skewness value calculated from values of a group.
Skewness (expr)-返回从组的值计算出的 skewness 值。
Examples:
例子:

```
> SELECT skewness(col) FROM VALUES (-10), (-20), (100), (1000) AS tab(col);
 1.1135657469022011
> SELECT skewness(col) FROM VALUES (-1000), (-100), (10), (20) AS tab(col);
 -1.1135657469022011
```

Since: 1.6.0
自: 1.6.0

slice 切片

slice(x, start, length) - Subsets array x starting from index start (array indices start at 1, or starting from the end if start is negative) with the specified length.
Slice (x，start，length)-子集数组 x 从指定的长度开始(数组索引从1开始，如果开始为负数则从末尾开始)。
Examples:
例子:

```
> SELECT slice(array(1, 2, 3, 4), 2, 2);
 [2,3]
```

```
> SELECT slice(array(1, 2, 3, 4), -2, 2);
 [3,4]

```

Since: 2.4.0

自: 2.4.0

smallint

smallint(expr) - Casts the value `expr` to the target data type `smallint`.

Smallint (expr)-将值 expr 强制转换为目标数据类型 smallint。

Since: 2.0.1

自: 2.0.1

some 一些

some(expr) - Returns true if at least one value of `expr` is true.

如果至少有一个 expr 值为真，则返回 true。

Examples:

例子:

```
> SELECT some(col) FROM VALUES (true), (false), (false) AS tab(col);
 true
> SELECT some(col) FROM VALUES (NULL), (true), (false) AS tab(col);
 true
> SELECT some(col) FROM VALUES (false), (false), (NULL) AS tab(col);
 false

```

Since: 3.0.0

自: 3.0.0

sort_array 排序数组

sort_array(array[, ascendingOrder]) - Sorts the input array in ascending or descending order according to the natural ordering of the array elements. NaN is greater than any non-NaN elements for double/float type. Null elements will be placed at the beginning of the returned array in ascending order or at the end of the returned array in descending order.

Sort _ array (array [， ascendingOrder])-根据数组元素的自然顺序按升序或降序对输入数组进行排序。对于 double/float 类型，NaN 大于任何非 NaN 元素。空元素将按升序放置在返回数组的开头，或按降序放置在返回数组的结尾。

Examples:

例子:

```
> SELECT sort_array(array('b', 'd', null, 'c', 'a'), true);
 [null,"a","b","c","d"]

```

Since: 1.5.0

自: 1.5.0

soundex 探测器

soundex(str) - Returns Soundex code of the string.

Soundex (str)-返回字符串的 Soundex 代码。

Examples:

例子:

```
> SELECT soundex('Miller');
 M460

```

Since: 1.5.0

自: 1.5.0

space 空间

space(n) - Returns a string consisting of `n` spaces.

Space (n)-返回一个由 n 个空格组成的字符串。

Examples:

例子:

```
> SELECT concat(space(2), '1');
   1

```

Since: 1.5.0

自: 1.5.0

spark_partition_id 火花分区 id

spark_partition_id() - Returns the current partition id.

Spark _ partition _ id ()-返回当前分区 id。

Examples:

例子:

```
> SELECT spark_partition_id();
 0

```

Since: 1.4.0

自: 1.4.0

split 分裂

split(str, regex, limit) - Splits `str` around occurrences that match `regex` and returns an array with a length of at most `limit`

Split (str，regex，limit)-对匹配 regex 的事件进行分割，并返回最多长度为的数组

Arguments:

论点:

- str - a string expression to split.
- 要拆分的字符串表达式。
- regex - a string representing a regular expression. The regex string should be a Java regular expression.
- 表示正则表达式的字符串。正则表达式应该是 Java 正则表达式。
- limit - an integer expression which controls the number of times the regex is applied. Limit ——一个整数表达式，它控制应用正则表达式的次数
  - limit > 0: The resulting array's length will not be more than limit, and the resulting array's last entry will contain all input beyond the last matched regex.
  - Limit > 0: 结果数组的长度不会超过 limit，结果数组的最后一个条目将包含超过最后一个匹配正则表达式的所有输入。
  - limit <= 0: regex will be applied as many times as possible, and the resulting array can be of any size.
  - Limit < = 0: regex 将被应用尽可能多的次数，得到的数组可以是任意大小的。

Examples:

例子:

```
> SELECT split('oneAtwoBthreeC', '[ABC]');
```

```
 ["one","two","three",""]
> SELECT split('oneAtwoBthreeC', '[ABC]', -1);
 ["one","two","three",""]
> SELECT split('oneAtwoBthreeC', '[ABC]', 2);
 ["one","twoBthreeC"]

```

Since: 1.5.0

自: 1.5.0

sqrt 平方英尺

sqrt(expr) - Returns the square root of `expr`.

Sqrt (expr)-返回 expr 的平方根。

Examples:

例子:

```
> SELECT sqrt(4);
 2.0

```

Since: 1.1.1

1.1.1

stack 堆栈

stack(n, expr1, ..., exprk) - Separates `expr1`, ..., `exprk` into `n` rows. Uses column names col0, col1, etc. by default unless specified otherwise.

(n，expr1，...，exprk)-将 expr1，...，exprk 分成 n 行。默认情况下使用列名 col0、 col1等，除非另有指定。

Examples:

例子:

```
> SELECT stack(2, 1, 2, 3);
 1  2
 3  NULL

```

Since: 2.0.0

自: 2.0.0

std 性病

std(expr) - Returns the sample standard deviation calculated from values of a group.

Std (expr)-返回由组值计算的样本标准差。

Examples:

例子:

```
> SELECT std(col) FROM VALUES (1), (2), (3) AS tab(col);
 1.0

```

Since: 1.6.0

自: 1.6.0

stddev

stddev(expr) - Returns the sample standard deviation calculated from values of a group.

返回从组的值计算出来的样例标准差。

Examples:

例子:

```
> SELECT stddev(col) FROM VALUES (1), (2), (3) AS tab(col);
 1.0

```

Since: 1.6.0

自: 1.6.0

stddev_pop 这是一个很好的例子

stddev_pop(expr) - Returns the population standard deviation calculated from values of a group.

返回从一个组的值计算出来的填充标准差。

Examples:

例子:

```
> SELECT stddev_pop(col) FROM VALUES (1), (2), (3) AS tab(col);
 0.816496580927726
```

```
3
```

Since: 1.6.0

自: 1.6.0

stddev_samp 山姆

stddev_samp(expr) - Returns the sample standard deviation calculated from values of a group.

返回从组的值计算出来的样例标准差。

Examples:

例子:

```
> SELECT stddev_samp(col) FROM VALUES (1), (2), (3) AS tab(col);
 1.0

```

Since: 1.6.0

自: 1.6.0

str_to_map 的地图

str_to_map(text[, pairDelim[, keyValueDelim]]) - Creates a map after splitting the text into key/value pairs using delimiters. Default delimiters are ',' for `pairDelim` and ':' for `keyValueDelim`. Both `pairDelim` and `keyValueDelim` are treated as regular expressions.

Str _ to _ map (text [， pairDelim [， keyValueDelim]])-在使用分隔符将文本分割为键/值对之后创建映射。默认分隔符是 ‘,’ 表示 pairDelim， ‘:’ 表示 keyValueDelim。pairDelim 和 keyValueDelim 都被视为正则表达式。

Examples:

例子:

```
> SELECT str_to_map('a:1,b:2,c:3', ',', ':');
 {"a":"1","b":"2","c":"3"}
> SELECT str_to_map('a');
 {"a":null}

```

Since: 2.0.1

自: 2.0.1

string 字符串

string(expr) - Casts the value `expr` to the target data type `string`.

String (expr)-将值 expr 强制转换为目标数据类型 string。

Since: 2.0.1

自: 2.0.1

struct 结构

struct(col1, col2, col3, ...) - Creates a struct with the given field values.

Struct (col1，col2，col3，...)-使用给定的字段值创建结构。

Examples:

例子:

```
> SELECT struct(1, 2, 3);
 {"col1":1,"col2":2,"col3":3}

```

Since: 1.4.0

自: 1.4.0

substr 底座

substr(str, pos[, len]) - Returns the substring of `str` that starts at `pos` and is of length `len`, or the slice of byte array that starts at `pos` and is of length `len`.

Substr (str，pos [，len])-返回以 pos 开始的长度 len 的 str 子字符串，或者以 pos 开始的长度 len 的字节数组片断。

substr(str FROM pos[FOR len]]) - Returns the substring of `str` that starts at `pos` and is of length `len`, or the slice of byte array that starts at `pos` and is of length `len`.

Substr (str FROM pos [FOR len])-返回以 pos 开始的长度为 len 的 str 子字符串，或者以 pos 开始的长度为 len 的字节数组切片。

Examples:

例子:

```
> SELECT substr('Spark SQL', 5);
 k SQL
> SELECT substr('Spark SQL', -3);
 SQL
```

```
> SELECT substr('Spark SQL', 5, 1);
 k
> SELECT substr('Spark SQL' FROM 5);
 k SQL
> SELECT substr('Spark SQL' FROM -3);
 SQL
> SELECT substr('Spark SQL' FROM 5 FOR 1);
 k

```

Since: 1.5.0
自: 1.5.0

substring 子串

substring(str, pos[, len]) - Returns the substring of `str` that starts at `pos` and is of length `len`, or the slice of byte array that starts at `pos` and is of length `len`.
Substring (str，pos [，len])-返回以 pos 开始的长度 len 的 str 子字符串，或者以 pos 开始的长度 len 的字节数组片断。
substring(str FROM pos[FOR len]]) - Returns the substring of `str` that starts at `pos` and is of length `len`, or the slice of byte array that starts at `pos` and is of length `len`.
Substring (str FROM pos [FOR len])-返回以 pos 开始的长度 len 的子字符串，或者以 pos 开始的长度 len 的字节数组片断。
Examples:
例子:

```
> SELECT substring('Spark SQL', 5);
 k SQL
> SELECT substring('Spark SQL', -3);
 SQL
> SELECT substring('Spark SQL', 5, 1);
 k
> SELECT substring('Spark SQL' FROM 5);
 k SQL
> SELECT substring('Spark SQL' FROM -3);
 SQL
> SELECT substring('Spark SQL' FROM 5 FOR 1);
 k

```

Since: 1.5.0
自: 1.5.0

substring_index 子字符串 _ 索引

substring_index(str, delim, count) - Returns the substring from `str` before `count` occurrences of the delimiter `delim`. If `count` is positive, everything to the left of the final delimiter (counting from the left) is returned. If `count` is negative, everything to the right of the final delimiter (counting from the right) is returned. The function substring_index performs a case-sensitive match when searching for `delim`.
Substring _ index (str，delim，count)-返回分隔符 delim 在计数之前的子字符串。如果计数为正，则返回最终分隔符(从左开始计数)左边的所有内容。如果计数为负，则返回最终分隔符(从右边开始计数)右边的所有内容。子字符串 _ index 函数在搜索 delim 时执行区分大小写的匹配。
Examples:
例子:

```
> SELECT substring_index('www.apache.org', '.', 2);
 www.apache

```

Since: 1.5.0
自: 1.5.0

sum 和

sum(expr) - Returns the sum calculated from values of a group.
Sum (expr)-返回从组的值计算出来的总和。
Examples:
例子:

```
> SELECT sum(col) FROM VALUES (5), (10), (15) AS tab(col);
 30
> SELECT sum(col) FROM VALUES (NULL), (10), (15) AS tab(col);
 25
> SELECT sum(col) FROM VALUES (NULL), (NULL) AS tab(col);
 NULL

```

Since: 1.0.0

自: 1.0.0

tan 古铜色

tan(expr) - Returns the tangent of `expr`, as if computed by `java.lang.Math.tan`.

返回 expr 的切线，就像是由 java.lang.Math.tan 计算一样。

Arguments:

论点:

- expr - angle in radians 弧度扩展角

Examples:

例子:

```
> SELECT tan(0);
 0.0

```

Since: 1.4.0

自: 1.4.0

tanh 坦

tanh(expr) - Returns the hyperbolic tangent of `expr`, as if computed by `java.lang.Math.tanh`.

返回 expr 的双曲正切，如同由 java.lang.Math.tanh 计算一样。

Arguments:

论点:

- expr - hyperbolic angle 双曲角

Examples:

例子:

```
> SELECT tanh(0);
 0.0

```

Since: 1.4.0

自: 1.4.0

timestamp 时间戳

timestamp(expr) - Casts the value `expr` to the target data type `timestamp`.

Timestamp (expr)-将值 expr 强制转换为目标数据类型 timestamp。

Since: 2.0.1

自: 2.0.1

timestamp_micros 时间戳

timestamp_micros(microseconds) - Creates timestamp from the number of microseconds since UTC epoch.

Timestamp _ micros (微秒)——根据 UTC 纪元以来的微秒数创建时间戳。

Examples:

例子:

```
> SELECT timestamp_micros(1230219000123123);
 2008-12-25 07:30:00.123123

```

Since: 3.1.0

3.1.0

timestamp_millis 时间戳 millis

timestamp_millis(milliseconds) - Creates timestamp from the number of milliseconds since UTC epoch.

Timestamp _ millis (milliseconds)——根据 UTC 纪元以来的毫秒数创建时间戳。

Examples:

例子:

```
> SELECT timestamp_millis(1230219000123);
 2008-12-25 07:30:00.123

```

Since: 3.1.0

3.1.0

timestamp_seconds 时间戳秒

timestamp_seconds(seconds) - Creates timestamp from the number of seconds (can be fractional) since UTC epoch.

时间戳 _ 秒(秒)——根据 UTC 纪元以来的秒数(可以是小数)创建时间戳。

Examples:
例子:

```
> SELECT timestamp_seconds(1230219000);
 2008-12-25 07:30:00
> SELECT timestamp_seconds(1230219000.123);
 2008-12-25 07:30:00.123

```

Since: 3.1.0
3.1.0

tinyint 小巧玲珑

tinyint(expr) - Casts the value `expr` to the target data type `tinyint`.
Tinyint (expr)-将值 expr 强制转换为目标数据类型 tinyint。
Since: 2.0.1
自: 2.0.1

to_csv 致 csv

to_csv(expr[, options]) - Returns a CSV string with a given struct value
To _ CSV (expr [， options])-返回带有给定结构值的 CSV 字符串
Examples:
例子:

```
> SELECT to_csv(named_struct('a', 1, 'b', 2));
 1,2
> SELECT to_csv(named_struct('time', to_timestamp('2015-08-26', 'yyyy-MM-dd')),
map('timestampFormat', 'dd/MM/yyyy'));
 26/08/2015

```

Since: 3.0.0
自: 3.0.0

to_date 到目前为止

to_date(date_str[, fmt]) - Parses the `date_str` expression with the `fmt` expression to a date. Returns null with invalid input. By default, it follows casting rules to a date if the `fmt` is omitted.

To _ date (date _ str [， fmt])-用 fmt 表达式解析 date _ str 表达式为 date。返回无效输入的 null。默认情况下，如果忽略 fmt，则按照强制转换规则转换为日期。

Arguments:

论点:

- date_str - A string to be parsed to date.
- Date _ str-要解析到日期的字符串。
- fmt - Date format pattern to follow. See Datetime Patterns for valid date and time format patterns.
- 要遵循的日期格式模式。有关有效的日期和时间格式模式，请参见日期时间模式。

Examples:

例子:

```
> SELECT to_date('2009-07-30 04:17:52');
 2009-07-30
> SELECT to_date('2016-12-31', 'yyyy-MM-dd');
 2016-12-31

```

Since: 1.5.0

自: 1.5.0

to_json 敬杰森

to_json(expr[, options]) - Returns a JSON string with a given struct value

To _ JSON (expr [， options])-返回一个带有给定结构值的 JSON 字符串

Examples:

例子:

```
> SELECT to_json(named_struct('a', 1, 'b', 2));
 {"a":1,"b":2}
> SELECT to_json(named_struct('time', to_timestamp('2015-08-26', 'yyyy-MM-dd')), map('timestampFormat', 'dd/MM/yyyy'));
 {"time":"26/08/2015"}
> SELECT to_json(array(named_struct('a', 1, 'b', 2)));
 [{"a":1,"b":2}]
> SELECT to_json(map('a', named_struct('b', 1)));
 {"a":{"b":1}}
```

```
> SELECT to_json(map(named_struct('a', 1),named_struct('b', 2)));
 {"[1]":{"b":2}}
> SELECT to_json(map('a', 1));
 {"a":1}
> SELECT to_json(array((map('a', 1))));
 [{"a":1}]

```

Since: 2.2.0

自: 2.2.0

to_timestamp 时间戳

to_timestamp(timestamp_str[, fmt]) - Parses the `timestamp_str` expression with the `fmt` expression to a timestamp. Returns null with invalid input. By default, it follows casting rules to a timestamp if the `fmt` is omitted. The result data type is consistent with the value of configuration `spark.sql.timestampType`.

To _ timestamp (timestamp _ str [， fmt])-使用 fmt 表达式将 timestamp _ str 表达式解析为时间戳。返回无效输入的 null。默认情况下，如果忽略 fmt，它将遵循将规则转换为时间戳的方式。结果数据类型与配置 spark.sql.timestampType 的值一致。

Arguments:

论点:

- timestamp_str - A string to be parsed to timestamp.
- Timestamp _ str-要解析为 timestamp 的字符串。
- fmt - Timestamp format pattern to follow. See Datetime Patterns for valid date and time format patterns.
- 要遵循的时间戳格式模式。有关有效的日期和时间格式模式，请参见 Datetime Patterns。

Examples:

例子:

```
> SELECT to_timestamp('2016-12-31 00:12:00');
 2016-12-31 00:12:00
> SELECT to_timestamp('2016-12-31', 'yyyy-MM-dd');
 2016-12-31 00:00:00

```

Since: 2.2.0

自: 2.2.0

to_unix_timestamp 时间戳

to_unix_timestamp(timeExp[, fmt]) - Returns the UNIX timestamp of the given time.

To _ UNIX _ timestamp (timeExp [， fmt])-返回给定时间的 UNIX 时间戳。

Arguments:

论点:

- timeExp - A date/timestamp or string which is returned as a UNIX timestamp.
- timeExp-作为 UNIX 时间戳返回的日期/时间戳或字符串。
- fmt - Date/time format pattern to follow. Ignored if timeExp is not a string. Default value is "yyyy-MM-dd HH:mm:ss". See Datetime Patterns for valid date and time format patterns.
- Fmt-要遵循的日期/时间格式模式。如果 timeExp 不是字符串，则忽略。默认值是“ yyyy-MM-dd HH: mm: ss”。有关有效的日期和时间格式模式，请参见日期时间模式。

Examples:

例子:

```
> SELECT to_unix_timestamp('2016-04-08', 'yyyy-MM-dd');
 1460098800

```

Since: 1.6.0

自: 1.6.0

to_utc_timestamp 时间戳

to_utc_timestamp(timestamp, timezone) - Given a timestamp like '2017-07-14 02:40:00.0', interprets it as a time in the given time zone, and renders that time as a timestamp in UTC. For example, 'GMT+1' would yield '2017-07-14 01:40:00.0'.

To _ UTC _ timestamp (timestamp，timezone)——给定一个类似于“2017-07-1402:40:00.0”的时间戳，将其解释为给定时区中的一个时间，并将该时间呈现为 UTC 中的一个时间戳。例如，“ GMT + 1”将产生“2017-07-1401:40:00.0”。

Examples:

例子:

```
> SELECT to_utc_timestamp('2016-08-31', 'Asia/Seoul');
 2016-08-30 15:00:00

```

Since: 1.5.0

自: 1.5.0

transform 变换

transform(expr, func) - Transforms elements in an array using the function.

Transform (expr，func)-使用函数在数组中转换元素。

Examples:

例子:

```
> SELECT transform(array(1, 2, 3), x -> x + 1);
 [2,3,4]
> SELECT transform(array(1, 2, 3), (x, i) -> x + i);
 [1,3,5]

```

Since: 2.4.0

自: 2.4.0

transform_keys 变换键

transform_keys(expr, func) - Transforms elements in a map using the function.

Transform _ keys (expr，func)-使用函数在映射中转换元素。

Examples:

例子:

```
> SELECT transform_keys(map_from_arrays(array(1, 2, 3), array(1, 2, 3)), (k, v) -> k + 1);
 {2:1,3:2,4:3}
> SELECT transform_keys(map_from_arrays(array(1, 2, 3), array(1, 2, 3)), (k, v) -> k + v);
 {2:1,4:2,6:3}

```

Since: 3.0.0

自: 3.0.0

transform_values 变换值

transform_values(expr, func) - Transforms values in the map using the function.

Transform _ values (expr，func)-使用函数在映射中转换值。

Examples:

例子:

```
> SELECT transform_values(map_from_arrays(array(1, 2, 3), array(1, 2, 3)), (k, v) -> v + 1);
 {1:2,2:3,3:4}
> SELECT transform_values(map_from_arrays(array(1, 2, 3), array(1, 2, 3)), (k, v) -> k + v);
 {1:2,2:4,3:6}

```

Since: 3.0.0

自: 3.0.0

translate 翻译

translate(input, from, to) - Translates the `input` string by replacing the characters present in the `from` string with the corresponding characters in the `to` string.

Translate (input，from，to)-通过将 from 字符串中的字符替换为 to 字符串中的相应字符来翻译输入字符串。

Examples:

例子:

```
> SELECT translate('AaBbCc', 'abc', '123');
 A1B2C3

```

Since: 1.5.0

自: 1.5.0

trim 修剪

trim(str) - Removes the leading and trailing space characters from `str`.

修剪(str)-从 str 中删除前导和尾随空格字符。

trim(BOTH FROM str) - Removes the leading and trailing space characters from `str`.

删除 str 的前导和尾随空格字符。

trim(LEADING FROM str) - Removes the leading space characters from `str`.

修剪(从 str 引导)-从 str 移除前导空格字符。

trim(TRAILING FROM str) - Removes the trailing space characters from `str`.

删除 str 的尾随空格字符。

trim(trimStr FROM str) - Remove the leading and trailing `trimStr` characters from `str`.

修剪(trimStr FROM str)-从 str 中删除前导和尾随的 trimStr 字符。
trim(BOTH trimStr FROM str) - Remove the leading and trailing `trimStr` characters from `str`.
修剪(包括 trimStr FROM str)-从 str 中删除前导和尾随的 trimStr 字符。
trim(LEADING trimStr FROM str) - Remove the leading `trimStr` characters from `str`.
修剪(龙头 trimStr FROM str)-从 str 中删除龙头 trimStr 字符。
trim(TRAILING trimStr FROM str) - Remove the trailing `trimStr` characters from `str`.
修剪(TRAILING trimStr FROM str)-从 str 删除 TRAILING trimStr 字符。

Arguments:

论点:

- str - a string expression Str-a 字符串表达式
- trimStr - the trim string characters to trim, the default value is a single space
- trimStr ——要修剪的修剪字符串字符，默认值是一个空格
- BOTH, FROM - these are keywords to specify trimming string characters from both ends of the string
- FROM-这些关键字用于指定从字符串的两端修剪字符串字符
- LEADING, FROM - these are keywords to specify trimming string characters from the left end of the string
- LEADING，FROM-这些关键字用于指定从字符串左端修剪字符串字符
- TRAILING, FROM - these are keywords to specify trimming string characters from the right end of the string
- TRAILING，FROM-这些关键字用于指定从字符串的右端修剪字符串

Examples:

例子:

```
> SELECT trim('    SparkSQL   ');
 SparkSQL
> SELECT trim(BOTH FROM '    SparkSQL   ');
 SparkSQL
> SELECT trim(LEADING FROM '    SparkSQL   ');
 SparkSQL
> SELECT trim(TRAILING FROM '    SparkSQL   ');
     SparkSQL
> SELECT trim('SL' FROM 'SSparkSQLS');
 parkSQ
> SELECT trim(BOTH 'SL' FROM 'SSparkSQLS');
 parkSQ
> SELECT trim(LEADING 'SL' FROM 'SSparkSQLS');
 parkSQLS
> SELECT trim(TRAILING 'SL' FROM 'SSparkSQLS');
 SSparkSQ

```

Since: 1.5.0
自: 1.5.0

trunc 中继线

trunc(date, fmt) - Returns `date` with the time portion of the day truncated to the unit specified by the format model `fmt`.
Trunk (date，fmt)-返回日期，日期的时间部分被截断为格式模型 fmt 指定的单元。
Arguments:
论点:

- date - date value or valid date string
- 日期日期值或有效日期字符串
- fmt - the format representing the unit to be truncated to Fmt-表示要截断的单元的格式
  - "YEAR", "YYYY", "YY" - truncate to the first date of the year that the date falls in
  - “ YEAR”、“ YYYY”、“广州欢聚时代”-截短到日期所在年份的第一个日期
  - "QUARTER" - truncate to the first date of the quarter that the date falls in
  - “ QUARTER”-截断到该日期所在季度的第一个日期
  - "MONTH", "MM", "MON" - truncate to the first date of the month that the date falls in
  - “ MONTH”、“ MM”、“ MON”-截断到该日期所在月份的第一个日期
  - "WEEK" - truncate to the Monday of the week that the date falls in
  - “ WEEK”-截断到该日期所在的星期一

Examples:
例子:

```
> SELECT trunc('2019-08-04', 'week');
 2019-07-29
> SELECT trunc('2019-08-04', 'quarter');
 2019-07-01
> SELECT trunc('2009-02-12', 'MM');
 2009-02-01
> SELECT trunc('2015-10-27', 'YEAR');
 2015-01-01

```

Since: 1.5.0
自: 1.5.0

try_add 试试 add

try_add(expr1, expr2) - Returns the sum of `expr1`and `expr2` and the result is null on overflow. The acceptable input types are the same with the `+` operator.

尝试 _ add (expr1，expr2)-返回 expr1和 expr2的和，溢出时结果为 null。可接受的输入类型与 + 运算符相同。

Examples:

例子:

```
> SELECT try_add(1, 2);
 3
> SELECT try_add(2147483647, 1);
 NULL
> SELECT try_add(date'2021-01-01', 1);
 2021-01-02
> SELECT try_add(date'2021-01-01', interval 1 year);
 2022-01-01
> SELECT try_add(timestamp'2021-01-01 00:00:00', interval 1 day);
 2021-01-02 00:00:00
> SELECT try_add(interval 1 year, interval 2 year);
 3-0

```

Since: 3.2.0

3.2.0

try_divide 试试分开

try_divide(dividend, divisor) - Returns `dividend`/`divisor`. It always performs floating point division. Its result is always null if `expr2` is 0. `dividend` must be a numeric or an interval. `divisor` must be a numeric.

尝试 _ divide (divism，divism)-Returns dividend/divism。它总是执行浮点除法。如果 expr2为0，则其结果始终为 null。股息必须是一个数字或一个间隔。除数一定是个数字。

Examples:

例子:

```
> SELECT try_divide(3, 2);
 1.5
> SELECT try_divide(2L, 2L);
```

```
 1.0
> SELECT try_divide(1, 0);
 NULL
> SELECT try_divide(interval 2 month, 2);
 0-1
> SELECT try_divide(interval 2 month, 0);
 NULL

```

Since: 3.2.0

3.2.0

typeof 类型

typeof(expr) - Return DDL-formatted type string for the data type of the input.

Typeof (expr)-为输入的数据类型返回 ddl 格式的类型字符串。

Examples:

例子:

```
> SELECT typeof(1);
 int
> SELECT typeof(array(1));
 array<int>

```

Since: 3.0.0

自: 3.0.0

ucase 尿素酶

ucase(str) - Returns `str` with all characters changed to uppercase.

返回所有字符都改为大写的 str。

Examples:

例子:

```
> SELECT ucase('SparkSql');
 SPARKSQL

```

Since: 1.0.1

自: 1.0.1

unbase64

unbase64(str) - Converts the argument from a base 64 string `str` to a binary.

Unbase64(str)-将参数从基数64个字符串转换为二进制。

Examples:

例子:

```
> SELECT unbase64('U3BhcmsgU1FM');
 Spark SQL

```

Since: 1.5.0

自: 1.5.0

unhex 诅咒

unhex(expr) - Converts hexadecimal `expr` to binary.

Unhex (expr)-将十六进制 expr 转换为二进制。

Examples:

例子:

```
> SELECT decode(unhex('537061726B2053514C'), 'UTF-8');
 Spark SQL

```

Since: 1.5.0

自: 1.5.0

unix_date Unix 日期

unix_date(date) - Returns the number of days since 1970-01-01.

Unix _ date (date)-返回自1970-01-01以来的天数。

Examples:

例子:

```
> SELECT unix_date(DATE("1970-01-02"));
 1
```

```
3
```

Since: 3.1.0

3.1.0

unix_micros Unix micros

unix_micros(timestamp) - Returns the number of microseconds since 1970-01-01 00:00:00 UTC.

Unix _ micros (timestamp)-返回自1970-01-0100:00:00 UTC 以来的微秒数。

Examples:

例子:

```
> SELECT unix_micros(TIMESTAMP('1970-01-01 00:00:01Z'));
 1000000

```

Since: 3.1.0

3.1.0

unix_millis Unix millis

unix_millis(timestamp) - Returns the number of milliseconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.

Unix _ millis (timestamp)-返回自1970-01-0100:00:00 UTC 以来的毫秒数。

Examples:

例子:

```
> SELECT unix_millis(TIMESTAMP('1970-01-01 00:00:01Z'));
 1000

```

Since: 3.1.0

3.1.0

unix_seconds Unix 秒

unix_seconds(timestamp) - Returns the number of seconds since 1970-01-01 00:00:00 UTC. Truncates higher levels of precision.

Unix _ seconds (timestamp)-返回自1970-01-0100:00:00 UTC 以来的秒数。截断更高级别的精度。

Examples:

例子:

```
> SELECT unix_seconds(TIMESTAMP('1970-01-01 00:00:01Z'));
 1

```

Since: 3.1.0

3.1.0

unix_timestamp 时间戳

unix_timestamp([timeExp[, fmt]]) - Returns the UNIX timestamp of current or specified time.

UNIX _ timestamp ([timeExp [， fmt]])-返回当前或指定时间的 UNIX 时间戳。

Arguments:

论点:

- timeExp - A date/timestamp or string. If not provided, this defaults to current time.
- timeExp-日期/时间戳或字符串。如果没有提供，默认为当前时间。
- fmt - Date/time format pattern to follow. Ignored if timeExp is not a string. Default value is "yyyy-MM-dd HH:mm:ss". See Datetime Patterns for valid date and time format patterns.
- Fmt-要遵循的日期/时间格式模式。如果 timeExp 不是字符串，则忽略。默认值是“ yyyy-MM-dd HH: mm: ss”。有关有效的日期和时间格式模式，请参见日期时间模式。

Examples:

例子:

```
> SELECT unix_timestamp();
 1476884637
> SELECT unix_timestamp('2016-04-08', 'yyyy-MM-dd');
 1460041200

```

Since: 1.5.0

自: 1.5.0

upper 上

upper(str) - Returns `str` with all characters changed to uppercase.

返回所有字符都改为大写的字符串。

Examples:
例子:

```
> SELECT upper('SparkSql');
 SPARKSQL

```

Since: 1.0.1
自: 1.0.1

uuid 尿路感染

uuid() - Returns an universally unique identifier (UUID) string. The value is returned as a canonical UUID 36-character string.
UUID ()-返回一个 UUID 字符串(UUID)， 该值作为一个规范的 UUID 36字符串返回。
Examples:
例子:

```
> SELECT uuid();
 46707d92-02f4-4817-8116-a4c3b23e6266

```

Note:
注意:
The function is non-deterministic.
这个函数是不确定的。
Since: 2.3.0
自: 2.3.0

var_pop 我不知道你在说什么

var_pop(expr) - Returns the population variance calculated from values of a group.
返回从组的值计算出的总体方差。
Examples:
例子:

```
> SELECT var_pop(col) FROM VALUES (1), (2), (3) AS tab(col);
 0.6666666666666666

```

Since: 1.6.0
自: 1.6.0

var_samp 瓦萨普

var_samp(expr) - Returns the sample variance calculated from values of a group.
Var _ samp (expr)-返回从组的值计算出的样本方差。
Examples:
例子:

```
> SELECT var_samp(col) FROM VALUES (1), (2), (3) AS tab(col);
 1.0

```

Since: 1.6.0
自: 1.6.0

variance 方差

variance(expr) - Returns the sample variance calculated from values of a group.
方差(expr)-返回从组的值计算出的样本方差。
Examples:
例子:

```
> SELECT variance(col) FROM VALUES (1), (2), (3) AS tab(col);
 1.0

```

Since: 1.6.0
自: 1.6.0

version 版本

version() - Returns the Spark version. The string contains 2 fields, the first being a release version and the second being a git revision.
Version ()-返回 Spark 版本。这个字符串包含两个字段，第一个是版本，第二个是 git 版本。
Examples:
例子:

```
> SELECT version();
 3.1.0 a6d6ea3efedbad14d99c24143834cd4e2e52fb40

```

Since: 3.0.0

自: 3.0.0

weekday 工作日

weekday(date) - Returns the day of the week for date/timestamp (0 = Monday, 1 = Tuesday, ..., 6 = Sunday).

返回日期/时间戳的星期几(0 = 星期一，1 = 星期二，... ，6 = 星期日)。

Examples:

例子:

```
> SELECT weekday('2009-07-30');
 3

```

Since: 2.4.0

自: 2.4.0

weekofyear 每年一周

weekofyear(date) - Returns the week of the year of the given date. A week is considered to start on a Monday and week 1 is the first week with >3 days.

年的周(日期)-返回给定日期的年的周。一周被认为是从星期一开始，第一周是第一个超过3天的星期。

Examples:

例子:

```
> SELECT weekofyear('2008-02-20');
 8

```

Since: 1.5.0

自: 1.5.0

when 当

CASE WHEN expr1 THEN expr2 [WHEN expr3 THEN expr4]* [ELSE expr5] END - When `expr1` = true, returns `expr2`; else when `expr3` = true, returns `expr4`; else returns `expr5`.
当 expr1 THEN expr2[WHEN expr3 THEN expr4] * [ELSE expr5] END-当 expr1 = true 时，返回 expr2; 当 expr3 = true 时，返回 expr4; ELSE 返回 expr5。

Arguments:

论点:

- expr1, expr3 - the branch condition expressions should all be boolean type.
- Expr1，expr3-分支条件表达式都应该是布尔类型。
- expr2, expr4, expr5 - the branch value expressions and else value expression should all be same type or coercible to a common type.
- Expr2，expr4，expr5-分支值表达式和 else 值表达式应该是相同的类型或者可以强制到一个公共类型。

Examples:

例子:

```
> SELECT CASE WHEN 1 > 0 THEN 1 WHEN 2 > 0 THEN 2.0 ELSE 1.2 END;
 1.0
> SELECT CASE WHEN 1 < 0 THEN 1 WHEN 2 > 0 THEN 2.0 ELSE 1.2 END;
 2.0
> SELECT CASE WHEN 1 < 0 THEN 1 WHEN 2 < 0 THEN 2.0 END;
 NULL

```

Since: 1.0.1

自: 1.0.1

width_bucket 宽度桶

width_bucket(value, min_value, max_value, num_bucket) - Returns the bucket number to which `value` would be assigned in an equiwidth histogram with `num_bucket` buckets, in the range `min_value` to `max_value`."

Width _ bucket (value，min _ value，max _ value，num _ bucket)——返回桶号，该值将在 num _ bucket 的等宽直方图中分配，范围为 min _ value 到 max _ value

Examples:

例子:

```
> SELECT width_bucket(5.3, 0.2, 10.6, 5);
 3
> SELECT width_bucket(-2.1, 1.3, 3.4, 3);
 0
```

```
> SELECT width_bucket(8.1, 0.0, 5.7, 4);
 5
> SELECT width_bucket(-0.9, 5.2, 0.5, 2);
 3

```

Since: 3.1.0

3.1.0

window 窗户

window(time_column, window_duration[, slide_duration[, start_time]]) - Bucketize rows into one or more time windows given a timestamp specifying column. Window starts are inclusive but the window ends are exclusive, e.g. 12:05 will be in the window [12:05,12:10) but not in [12:00,12:05). Windows can support microsecond precision. Windows in the order of months are not supported. See 'Window Operations on Event Time' in Structured Streaming guide doc for detailed explanation and examples.

Window (time _ column，window _ duration [， slide _ duration [， start _ time])-在指定列的时间戳的情况下，将 Bucketize 行分成一个或多个时间窗口。窗口的开始时间是包含在内的，但窗口的结束时间是独占的，例如12:05在窗口[12:05,12:10] ，而不在[12:00,12:05]。Windows 可以支持微秒级的精度。不支持按月排列的 Windows。参见“事件时间窗口操作”的结构化流指南文档中的详细解释和示例。

Arguments:

论点:

- time_column - The column or the expression to use as the timestamp for windowing by time. The time column must be of TimestampType.
- Time _ column ——按时间作为窗口时间戳的列或表达式。时间列必须是 TimestampType。
- window_duration - A string specifying the width of the window represented as "interval value". (See Interval Literal for more details.) Note that the duration is a fixed length of time, and does not vary over time according to a calendar.
- Window _ duration-指定窗口宽度的字符串，表示为“ interval value”。(详见 Interval Literal。)请注意，持续时间是一个固定的时间长度，并且不会随着时间的推移而变化。
- slide_duration - A string specifying the sliding interval of the window represented as "interval value". A new window will be generated every slide_duration. Must be less than or equal to the window_duration. This duration is likewise absolute, and does not vary according to a calendar.
- Slide _ duration-指定窗口的滑动间隔的字符串，表示为“ interval value”。每个幻灯片持续时间将生成一个新窗口。必须小于或等于窗口 _ 持续时间。这种持续时间同样是绝对的，不会随着日历的变化而变化。
- start_time - The offset with respect to 1970-01-01 00:00:00 UTC with which to start window intervals. For

example, in order to have hourly tumbling windows that start 15 minutes past the hour, e.g. 12:15-13:15, 13:15-14:15... provide start_time as 15 minutes.

- 开始 _ 时间——与1970-01-0100:00:00 UTC 相关的偏移量，用于开始窗口间隔。例如，为了让每小时一次的翻滚窗口在一小时后15分钟开始，例如12:15-13:15,13:15-14:15... 提供开始时间为15分钟。

Examples:

例子:

```
> SELECT a, window.start, window.end, count(*) as cnt FROM VALUES ('A1', '2021-01-01 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:06:00'), ('A2', '2021-01-01 00:01:00') AS tab(a, b) GROUP by a, window(b, '5 minutes') ORDER BY a, start;
  A1    2021-01-01 00:00:00 2021-01-01 00:05:00 2
  A1    2021-01-01 00:05:00 2021-01-01 00:10:00 1
  A2    2021-01-01 00:00:00 2021-01-01 00:05:00 1
> SELECT a, window.start, window.end, count(*) as cnt FROM VALUES ('A1', '2021-01-01 00:00:00'), ('A1', '2021-01-01 00:04:30'), ('A1', '2021-01-01 00:06:00'), ('A2', '2021-01-01 00:01:00') AS tab(a, b) GROUP by a, window(b, '10 minutes', '5 minutes') ORDER BY a, start;
  A1    2020-12-31 23:55:00 2021-01-01 00:05:00 2
  A1    2021-01-01 00:00:00 2021-01-01 00:10:00 3
  A1    2021-01-01 00:05:00 2021-01-01 00:15:00 1
  A2    2020-12-31 23:55:00 2021-01-01 00:05:00 1
  A2    2021-01-01 00:00:00 2021-01-01 00:10:00 1

```

Since: 2.0.0

自: 2.0.0

xpath

xpath(xml, xpath) - Returns a string array of values within the nodes of xml that match the XPath expression.

XPath (xml，XPath)——返回 xml 节点中与 XPath 表达式匹配的值的字符串数组。

Examples:

例子:

```
> SELECT xpath('<a><b>b1</b><b>b2</b><b>b3</b><c>c1</c><c>c2</c></a>','a/b/text()');
 ["b1","b2","b3"]

```

Since: 2.0.0

自: 2.0.0

xpath_boolean 布尔值

xpath_boolean(xml, xpath) - Returns true if the XPath expression evaluates to true, or if a matching node is found.

XPath _ boolean (xml，XPath)-如果 XPath 表达式的计算结果为 true，或者找到匹配的节点，则返回 true。

Examples:

例子:

```
> SELECT xpath_boolean('<a><b>1</b></a>','a/b');
 true

```

Since: 2.0.0

自: 2.0.0

xpath_double 双倍

xpath_double(xml, xpath) - Returns a double value, the value zero if no match is found, or NaN if a match is found but the value is non-numeric.

Xpath _ double (xml，xpath)-返回一个双值，如果没有找到匹配，返回值0，如果找到匹配，返回值为非数值，返回值 NaN。

Examples:

例子:

```
> SELECT xpath_double('<a><b>1</b><b>2</b></a>', 'sum(a/b)');
 3.0

```

Since: 2.0.0

自: 2.0.0

xpath_float 2010年10月11日

xpath_float(xml, xpath) - Returns a float value, the value zero if no match is found, or NaN if a match is found but the value is non-numeric.

Xpath _ float (xml，xpath)-返回一个 float 值，如果没有找到匹配，返回值0，如果找到匹配，返回值为非数值，返回 NaN。

Examples:
例子:

```
> SELECT xpath_float('<a><b>1</b><b>2</b></a>', 'sum(a/b)');
 3.0

```

Since: 2.0.0
自: 2.0.0

xpath_int 2010年10月11日

xpath_int(xml, xpath) - Returns an integer value, or the value zero if no match is found, or a match is found but the value is non-numeric.
Xpath _ int (xml，xpath)-返回一个整数值，如果没有找到匹配项，则返回值0，或者找到一个匹配项，但该值是非数值。
Examples:
例子:

```
> SELECT xpath_int('<a><b>1</b><b>2</b></a>', 'sum(a/b)');
 3

```

Since: 2.0.0
自: 2.0.0

xpath_long 长度

xpath_long(xml, xpath) - Returns a long integer value, or the value zero if no match is found, or a match is found but the value is non-numeric.
Xpath _ long (xml，xpath)-返回一个长整数值，如果没有找到匹配项，则返回值0，或者找到一个匹配项，但该值是非数值。
Examples:
例子:

```
> SELECT xpath_long('<a><b>1</b><b>2</b></a>', 'sum(a/b)');
 3

```

Since: 2.0.0

自: 2.0.0

xpath_number 数字

xpath_number(xml, xpath) - Returns a double value, the value zero if no match is found, or NaN if a match is found but the value is non-numeric.

Xpath _ number (xml，xpath)-返回一个双值，如果没有找到匹配，返回值0，如果找到匹配，返回值为非数值，返回值 NaN。

Examples:

例子:

```
> SELECT xpath_number('<a><b>1</b><b>2</b></a>', 'sum(a/b)');
 3.0

```

Since: 2.0.0

自: 2.0.0

xpath_short 短路

xpath_short(xml, xpath) - Returns a short integer value, or the value zero if no match is found, or a match is found but the value is non-numeric.

Xpath _ short (xml，xpath)-返回一个简短的整数值，如果没有找到匹配项，则返回值0，或者找到一个匹配项，但该值是非数值。

Examples:

例子:

```
> SELECT xpath_short('<a><b>1</b><b>2</b></a>', 'sum(a/b)');
 3

```

Since: 2.0.0

自: 2.0.0

xpath_string 字符串

xpath_string(xml, xpath) - Returns the text contents of the first xml node that matches the XPath expression.

XPath _ string (xml，XPath)-返回与 XPath 表达式匹配的第一个 xml 节点的文本内容。

Examples:
例子:

```
> SELECT xpath_string('<a><b>b</b><c>cc</c></a>','a/c');
 cc

```

Since: 2.0.0
自: 2.0.0

xxhash64 64号

xxhash64(expr1, expr2, ...) - Returns a 64-bit hash value of the arguments.
Xxhash64(expr1，expr2，...)-返回参数的64位散列值。
Examples:
例子:

```
> SELECT xxhash64('Spark', array(123), 2);
 5602566077635097486

```

Since: 3.0.0
自: 3.0.0

year 年

year(date) - Returns the year component of the date/timestamp.
返回日期/时间戳的年份部分。
Examples:
例子:

```
> SELECT year('2016-07-30');
 2016

```

Since: 1.5.0
自: 1.5.0

zip_with 压缩

zip_with(left, right, func) - Merges the two given arrays, element-wise, into a single array using function. If one array is shorter, nulls are appended at the end to match the length of the longer array, before applying function.
Zip _ with (left，right，func)-使用函数将两个给定的数组按元素合并到单个数组中。如果一个数组较短，则在应用函数之前，在末尾追加空值以匹配较长数组的长度。

Examples:

例子:

```
> SELECT zip_with(array(1, 2, 3), array('a', 'b', 'c'), (x, y) -> (y, x));
 [{"y":"a","x":1},{"y":"b","x":2},{"y":"c","x":3}]
> SELECT zip_with(array(1, 2), array(3, 4), (x, y) -> x + y);
 [4,6]
> SELECT zip_with(array('a', 'b', 'c'), array('d', 'e', 'f'), (x, y) -> concat(x, y));
 ["ad","be","cf"]

```

Since: 2.4.0

自: 2.4.0

|

expr1 | expr2 - Returns the result of bitwise OR of `expr1` and `expr2`.

返回 expr1和 expr2的按位 OR 结果。

Examples:

例子:

```
> SELECT 3 | 5;
 7

```

Since: 1.4.0

自: 1.4.0

||

expr1 || expr2 - Returns the concatenation of `expr1` and `expr2`.

Expr1 | | expr2-返回 expr1和 expr2的连接。

Examples:

例子:

```
> SELECT 'Spark' || 'SQL';
 SparkSQL
> SELECT array(1, 2, 3) || array(4, 5) || array(6);
 [1,2,3,4,5,6]

```

Note:

注意:

|| for arrays is available since 2.4.0.

从2.4.0开始就可以使用数组。

Since: 2.3.0

自: 2.3.0

~

~ expr - Returns the result of bitwise NOT of `expr`.

返回 expr 的按位 NOT 结果。

Examples:

例子:

```
> SELECT ~ 0;
 -1

```

Since: 1.4.0

自: 1.4.0