String Operators

Operator CONTAINS

CONTAINS returns TRUE if the *SearchExpression* is present within the *SourceExpression*, otherwise it returns FALSE.

Syntax:

```
>>-CONTAINS--(--SourceExpression--, --SearchExpression--) -------><
```

Examples

```
CONTAINS('Hello World!', 'ello'); ---Returns TRUE
CONTAINS('Hello World!', 'daisy');---Returns FALSE
```

Operator ENDSWITH

ENDSWITH returns TRUE if *SourceExpression* ends with *SearchExpression*, otherwise it returns FALSE.

Syntax:

```
>>-ENDSWITH--(--SourceExpression--, --SearchExpression--) -------><
```

Examples:

ENDSWITH('Hello World!', 'World!'); ---Returns TRUE

ENDSWITH('Hello World!', 'World'); ---Returns FALSE

Operator STARTSWITH

STARTSWITH returns TRUE if *SourceExpression*starts with *SearchExpression*, otherwise it returns FALSE.

Syntax:

```
>>-STARTSWITH--(--SourceExpression--, --SearchExpression--) -------><
```

Examples:

STARTSWITH('Hello World!', Hello’); ---Returns TRUE

STARTSWITH('Hello World!', 'World'); ---Returns FALSE

Operator LEFT

It returns a string consisting of the source string truncated to the LEFT given by the length expression.

Syntax:

```
>>-LEFT--(--source_string--, --LengthIntegerExpression--)-------><
```

Examples:

LEFT ('Hello', 2) ---returns 'He'

LEFT ('12345',3) ---returns '123'

Operator RIGHT

It returns a string consisting of the source string truncated to the RIGHT given by the length expression.

Syntax:

```
>>-RIGHT--(--source_string--, --LengthIntegerExpression--)-------><
```

Examples:

RIGHT (‘Hello’, 2) ---returns ‘lo’

RIGHT (‘12345’,3) ---returns ‘345’

Operator LENGTH

The LENGTH calculates the length of a string and returns an integer value.

Syntax:

```
>>-LENGTH--(--source_string--)----------------------------------><
```

Examples:

LENGTH('Hello World!'); ---Returns 12

LENGTH(''); ---Returns 0

Operators LOWER and LCASE

These both are same and return a string in which all uppercase letters get converted into corresponding lowercase letters.

Syntax:

```
>>-+-LOWER-+--(--source_string--)------------------------------><
```

OR

```
>>-+-LCASE-+--(--source_string--)------------------------------><
```

Example:

LOWER('Mr Smith') ---Returns ‘mr smith’

LCASE('ABCD') --- Returns ‘abcd’

Operators UPPER and UCASE

These both are same and return a string in which all uppercase letters get converted into corresponding lowercase letters.

Syntax:

```
>>-+-UPPER-+--(--source_string--)------------------------------><
```

OR

```
>>-+-UCASE-+--(--source_string--)------------------------------><
```

Examples:

UPPER ('mr smith') ---Returns ‘MR SMITH’

UCASE(‘abcd’) --- Returns ‘ABCD

Operator LTRIM

Syntax:

```
>>-LTRIM--(--source_string--)----------------------------------><
```

```
Example:
```

LTRIM(‘ HELLO ‘); --- Returns ‘Hello ‘ (i.e. space is removed from left).

Operator RTRIM

Syntax:

```
>>-RTRIM--(--source_string--)----------------------------------><
```

Example:

RTRIM (‘ HELLO ‘); --- Returns ‘Hello‘ (i.e. space is removed from right).

Operator TRIM

Syntax:

```
>>-TRIM--(--+-----------------------------------------------+---->
            '-+-trim_singleton------------------+-- FROM -'
              | .-BOTH-----.                    |
              '-+-LEADING--+--+-----------------+-'
                '-TRAILING-' '-trim_singleton-'

>--source_string--)-------------------------------------------><
```

Examples:

TRIM(TRAILING 'b' FROM 'aaabBb') ---Returns ‘aaabB’

TRIM (LEADING FROM ‘ Hello ‘) ---Returns ‘Hello ‘;

Above example is same is LTRIM (‘ Hello ‘)

TRIM(‘ Hello ‘) ---Returns ‘Hello’ (i.e removes both the spaces)

TRIM('b' FROM 'bbbaaabbb') --returns ‘aaa’

Operator OVERLAY

It replaces part of a string with a substring.

Syntax:

```
>>-OVERLAY--(--source_string-- PLACING --source_string2--------->

>-- FROM --start_position--+----------------------+--)--------><
                           '- FOR --string_length-'
```

Example:

OVERLAY ('ABCDEFGHIJ' PLACING '1234' FROM 4 FOR 3) –Returns 'ABC1234GHIJ'

Meaning of above line is, it goes to the 4th position (‘D’) from start and replaces next 3 strings (‘DEF’) from 4th position with ‘1234’.

Operator POSITION

It returns the position of one string within another.

Syntax:

```
>>-POSITION--(--SearchExpression--IN--SourceExpression--+----------------------+-->
                                                        '-FROM--FromExpression-'

>--+--------------------------+--)----------------------------><
   '-REPEAT--RepeatExpression-'
```

Example:

```
POSITION('Village' IN 'HursleyVillage');   returns 9
POSITION('Town' IN 'HursleyVillage');   returns 0

        POSITION ('B' IN 'ABCABCABCABCABC');-> returns 2
        POSITION ('D' IN 'ABCABCABCABCABC');-> returns 0

        POSITION ('A' IN 'ABCABCABCABCABC' FROM 4);-> returns 4
        POSITION ('C' IN 'ABCABCABCABCABC' FROM 2);-> returns 3
```

POSITION ('B' IN 'ABCABCABCABCABC' REPEAT 2);-> returns 5
POSITION ('C' IN 'ABCABCABCABCABC' REPEAT 4);-> returns 12

POSITION ('A' IN 'ABCABCABCABCABC' FROM 4 REPEAT 2);-> returns 7
POSITION ('AB' IN 'ABCABCABCABCABC' FROM 2 REPEAT 3);-> returns 10

POSITION ('A' IN 'ABCABCABCABCABC' REPEAT -2);-> returns 10
POSITION ('BC' IN 'ABCABCABCABCABC' FROM 2 REPEAT -3);-> returns 5

Operator REPLACE

It replaces parts of a string with supplied substrings.

Syntax:

```
>>-REPLACE--(--SourceStringExpression--,--SearchStringExpression--+---------------------------+--)-><
                                                                  '-ReplaceStringExpression-'
```

Examples:

REPLACE('ABCDABCDABCDA', 'A', 'AA') --- Returns AABCDAABCDAABCDAA

REPLACE('AAAABCDEFGHAAAABCDEFGH', 'AA', 'A') ---Returns AABCDEFGHAABCDEFGH

REPLACE('AAAAABCDEFGHAAAABCDEFGH', 'AA', 'XYZ') ---Returns XYZXYZABCDEFGHXYZXYZBCDEFGH

Operator REPLICATE

It returns a string made up of multiple copies of a supplied string.

Syntax:

>>-REPLICATE--(--*PatternStringExpression*--,--*CountNumericExpression*--)-><

Example:

REPLICATE (‘a’,5) --- Returns ‘aaaaa’

Operator SUBSTRING

It extracts characters from a string to create another string.

Syntax:

```
>>-SUBSTRING--(--SourceExpression--+- FROM --StartPosition------+-->
                                   +- BEFORE --BeforeExpression-+
                                   '- AFTER --AfterExpression---'

>--+---------------------+--)--------------------------------><
   '- FOR --StringLength-'
```

Examples:

SUBSTRING('HelloWorld!' FROM 7 FOR 4) –Returns ‘Worl’

SUBSTRING('HelloWorld!' BEFORE 'World'); --Returns ‘Hello ‘

SUBSTRING('HelloWorld!' BEFORE 'World' FOR 3); --Returns ‘lo ‘

SUBSTRING('HelloWorld!' BEFORE 'e'); --Returns ‘H’

SUBSTRING('HelloWorld!' AFTER 'World'); -- Returns ‘!’

```
SUBSTRING('HelloWorld!' AFTER 'W' FOR 2); -- Returns ‘or’

SUBSTRING('HelloWorld!' AFTER 'P'); -- Returns ‘ ‘;
```