SQream DB v1.9.6 Cheat Sheet - STRINGS

Types

VARCHAR(n) – ASCII string of maximum length n
NVARCHAR(n) – UTF-8 string of maximum length n
NVARCHAR can't be aggregated in GROUP BY or used as a join key in this SQream DB version

Conversions

```
CAST(9 as varchar(2))
 → '9'
true :: varchar(5)
 → 'true'
CAST('1997-01-01' as datetime)
 → 1997-01-01 00:00:00.000
2015 :: varchar(5)
 → '2015'
CAST('2015' as varchar(2))
```

SQream DB can't convert between NVARCHAR and VARCHAR with casts in this version.

String literals

- Strings literals must be surrounded by a single quote (')
- Strings containing single quotes must be escaped by writing two adjacent single quotes:

```
SELECT 'This''ll do'
 → This'll do
```

Avoid confusion!
A double quote is a way to qualify an identifier:

```
SELECT "public"."t"."firstname"
```

Measurements

```
LENGTH('SQream DB ') → 9
LENGTH('Flüßigkeit') → 10
OCTET_LENGTH('SQream DB') → 9
OCTET_LENGTH('Flüßigkeit') → 12
```

Modifications

Concatenation

```
'SQream      ' || 'DB'
 → 'SQreamDB'
'SQream' || '  DB'
 → 'SQream  DB'
'Value:' || (42 :: varchar(1))
 → 'Value*'
'Value:' || (42 :: varchar(2))
 → 'Value:42'
```

Various

```
SUBSTRING(column,start position,length)
```

Examples:

```
SUBSTRING('SQream DB',1,3)
 → 'SQr'
SUBSTRING('SQream DB',0,3)
 → 'SQ'
UPPER('SQream DB') → 'SQREAM DB'
LOWER('SQream DB') → 'sqream db'
REVERSE('Meow') → 'Woem'
RTRIM(' Quack ') → '  Quack'
LTRIM(' Quack ') → 'Quack'
```

Where x is 'Flüßigkeit' (NVARCHAR):

```
LEFT(x,6) → 'Flüßig'
RIGHT(x,4) → 'keit'
```

LEFT and RIGHT only work on NVARCHAR columns, not literals.

Gotcha!
Trailing whitespace on the right is automatically clipped

Search and Matching

Where x is 'SQream DB v1.9.6' (VARCHAR)
and y is 'DB' (VARCHAR)

```
ISPREFIXOF(needle,haystack)
```

Returns 1/0 (= true/false) if x is prefix of y.
Example:

```
ISPREFIXOF(x,y)
→ 0
```

Search and Matching (cont.)

Where x is 'SQream DB v1.9.6' (VARCHAR)
and y is 'DB' (VARCHAR)

```
CHARINDEX(needle,haystack[, start index])
```

Returns the first character index, or 0 if not found
Example:

```
CHARINDEX(y,x,1)
→ 0
```

Count patterns matching regex. Syntax:

```
REGEXP_COUNT(column, regex [,start index])
```

Example:

```
REGEXP_COUNT(x,'[0-9]')
→ 3
REGEXP_COUNT(x,'[0-9]',13)
→ 2
```

Find first occurrence of regex pattern. Syntax:

```
REGEXP_INSTR(column, regex [,start index])
```

Example:

```
REGEXP_INSTR(x,'[0-9]')
→ 12
```

Extract substring of regex pattern match or '' if not found. Syntax:

```
REGEXP_SUBSTR(column, regex [,start index])
```

Example:

```
REGEXP_SUBSTR(x,'[0-9].[0-9].[0-9]')
→ '1.9.6'
```

Returns the first occurrence index of the pattern. Syntax (SQL Server standard, see Pattern matching in the next column):

```
PATINDEX(pattern, column)
```

Example:

```
PATINDEX('%[0-9]%',x)
→ '1'
```

Like
Used in a WHERE clause to search for a specified pattern in a column
Example:

```
SELECT x FROM t WHERE x LIKE '%DB%'
→ 'SQream DB v1.9.6'
  'DB'
```

Patterns

Pattern matching

Syntax	Description
%	match zero or more characters
_	match exactly one character
[A-Z]	match any character between A and Z inclusive
[^A-Z]	match any character not between A and Z
[abS]	match any one of a b and S
[^de]	match any character that isn't one of d and e
[abC-F]	match a b or between C and F

Regex pattern matching

Syntax	Description
^	Match beginning of a string
$	Match end of a string
.	Match any character
*	Match the previous pattern **0+** times
+	Match the previous pattern **1+** times
?	Match the previous pattern **0 or 1** times
de|abc	Match either 'de' or 'abc'
(abc)*	Match **0+** instances of the sequence 'abc'
{2}	Match the prev. pattern 2 times
{2,4}	Match the previous pattern between two and four times
[a-dX], [^a-dX]	Matches any character that is (or isn't - if ^ is used) either a,b,c,d or X. A dash (-) character between two other characters forms a range. Example: [0-9] matches any decimal digit. To include a literal] character, it must immediately follow the opening bracket [. To include a literal dash (-) character, it must appear first or last. Any character that does not have a defined special meaning inside a [] pair matches only itself.