Date Functions

+ and - binary operations

There also exist algebraic operators (+, -) to add or subtract dates.

toDate(String dateTime, String format)

Creates a date time from a string and a format. Example: `toDate("2019/03/28 14:06:42", "yyyy/MM/dd HH:mm:ss")`

toDate(long milliseconds)

Creates a new date with Unix timestamp (milliseconds since January 1, 1970 UTC)

Year(<Date d>|<String s, Pattern p>)

Extracts the year part of the date

Example

```
SELECT YEAR("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 2013
```

Month(<Date d>|<String s, Pattern p>)

Extracts the month part of the date

Example

```
SELECT MONTH("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 12
```

Week(<Date d>|<String s, Pattern p>)

Returns the week number of the date

Example

```
SELECT WEEK("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 52
```

Day(<Date d>|<String s, Pattern p>)

Extracts the day part of the date

Example

```
SELECT DAY("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 24
```

Hour(<Date d>|<String s, Pattern p>)

Returns the hour of the date

Example

```
SELECT HOUR("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 5
```

Minute(<Date d>|<String s, Pattern p>)

Returns the minute of the date

Example

```
SELECT MINUTE("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 30
```

Second(<Date d>|<String s, Pattern p>)

Returns the second of the date

Example

```
SELECT SECOND("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 10
```

Millisecond(<Date d>|<String s, Pattern p>)

Returns the millisecond of the date

Example

```
SELECT MILLISECOND("12/24/13 5:30:10.970","M/d/y h:m:s.SSS") FROM Stream
=> 970
```

WeekDay(<Date d>|<String s, Pattern p>)

Returns the week day index of the date (1 = Sunday, 2 = Monday, ..., 7 = Saturday)

Example

```
SELECT WEEKDAY("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 3
```

DayOfMonth(<Date d>|<String s, Pattern p>)

Synonym for day().

Example

```
SELECT DAYOFMONTH("12/24/13 5:30:10","M/d/y h:m:s") FROM Stream
=> 24
```

Years(Date d1, Date d2)

Computes the number of years between the two dates.

Months(Date d1, Date d2)

Computes the number of months between the two dates.

Days(Date d1, Date d2)

Computes the number of days between the two dates.

Hours(Date d1, Date d2)

Computes the number of hours between the two dates.

Minutes(Date d1, Date d2)

Computes the number of minutes between the two dates.

Seconds(Date d1, Date d2)

Computes the number of seconds between the two dates.

Milliseconds(Date d1, Date d2)

Computes the number of milliseconds between the two dates.

BusinessDays(Date from, Date to)

Computes the number of business days between the two dates.

CurDate()

Return the current system time specific date

DateInMillis()

Returns the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC

DateInMillis(Date d)

Returns the difference, measured in milliseconds, between the given date and midnight, January 1, 1970 UTC

toLong(Date d)

Same as DateInMillis(d)

NanoTime()

Returns the current value of the time source, in nanoseconds.

StreamDate()

Returns the current stream time specific date

ToString(Date d, Pattern p)

Converts the given date to a *string* value according to the given pattern.

ToString(Date d, Pattern p, Timezone ts)

Converts the given date to a *string* value according to the given pattern in the given timezone. (Using the functions from SimpleDateFormat).

format(Date d, String pattern)

Same as ToString, but evaluates Pattern only the first time (i.e. first pattern will be used for every data object)

setter Functions

You can set all values in a date individually with the setter functions:

setYear(Date date, int year), setMonth(Date date, int month), setDayOfMonth(Date date, int day), setHourOfDay(Date date, int hour), setMinute(Date date, int minute), setSecond(Date date, int second), setMilliSecond(Date date, int millisecond) .

Example:

```
setYear = MAP({
                expressions = [
                  ['setYear(someDate, 2042)','inFuture'],
                  ['setMinute(someDate, 42)','changedMinute']
                ],
                keepinput = true
              },
              plusHour
            )
```

Periods

If you want to add or substract a period to or from a date, you can work with Periods. Internally, they work with the Java Periods, hence, you can have a look at that class: https://docs.oracle.com/javase/8/docs/api/java/time/Period.html

toTimePeriod(int years, int months, int days)

Creates a time period

Date date + Period period

Adds a time period to a date. Note: you can have negative time periods, hence, you can also substract but don't need a substract sign.

Example:

```
baseInfo = MAP({
                expressions = [
                  ['toDate("2019/03/28 14:06:42", "yyyy/MM/dd HH:mm:ss")',
'now'],
                  ['toTimePeriod(-1,0,0)', 'oneYear']
                ]
              },
              input
            )

minusYear = MAP({
                expressions = [
                  ['now + oneYear','oneYearAgo']
                ],
                KEEPINPUT = true
              },
              baseInfo
            )
```

Durations

If you want to add or substract a duration from a date, you can work with Durations. This is for adding days, hours, minutes, seconds, milliseconds and nanoseconds. If you want to take a closer look, see https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html

toTimeDuration(long days, long hours, long minutes, long seconds, long milliseconds, long nanoseconds)

Creates a duration by summing all the given values.

Date date + Duration duration

Adds a duration to a date. Note: you can have negative time durations, hence, you can also substract but don't need a substract sign.

Example:

```
baseInfo = MAP({
                expressions = [
                  ['toDate("2019/03/28 14:06:42", "yyyy/MM/dd HH:mm:ss")',
'now'],
                  ['toTimePeriod(-1,0,0)', 'oneYear']
                ]
              },
              input
            )

minusYear = MAP({
                expressions = [
                  ['now + oneYear','oneYearAgo'],
                  ['toTimeDuration(0,1,0,0,0,0)', 'oneHour']
                ],
                KEEPINPUT = true
              },
              baseInfo
            )

plusHour = MAP({
                expressions = [
                  ['oneYearAgo + oneHour','oneYearAgoPlusOneHour']
                ],
                KEEPINPUT = true
              },
              minusYear
            )
```