Date & time data types

Snowflake supports data types for managing dates, times, and timestamps (combined date + time). Snowflake also supports formats for string constants used in manipulating dates, times, and timestamps.

Data types

Snowflake supports the following date and time data types:

- DATE
- DATETIME
- TIME
- TIMESTAMP_LTZ , TIMESTAMP_NTZ , TIMESTAMP_TZ

Note
For DATE and TIMESTAMP data, Snowflake recommends using years between 1582 and 9999. Snowflake accepts some years outside this range, but years prior to 1582 should be avoided due to limitations on the Gregorian Calendar.

Snowflake supports a single DATE data type for storing dates (with no time elements).

DATE accepts dates in the most common forms (`YYYY-MM-DD`, `DD-MON-YYYY`, and so on).

In addition, all accepted TIMESTAMP values are valid inputs for dates, but the TIME information is truncated.

DATETIME

DATETIME is an alias for TIMESTAMP_NTZ.

TIME

Snowflake supports a single TIME data type for storing times in the form of `HH:MI:SS`.

TIME supports an optional precision parameter for fractional seconds (for example, `TIME(3)`). Time precision can range from 0 (seconds) to 9 (nanoseconds). The default precision is 9.

All TIME values must be between `00:00:00` and `23:59:59.999999999`. TIME internally stores “wallclock” time, and all operations on TIME values are performed without taking any time zone into consideration.

TIMESTAMP_LTZ , TIMESTAMP_NTZ , TIMESTAMP_TZ

Snowflake supports three variations of timestamp.

TIMESTAMP_LTZ TIMESTAMP_LTZ internally stores UTC time with a specified precision. However, all operations are performed in the current session's time zone, controlled by the TIMEZONE session parameter.

Aliases for TIMESTAMP_LTZ:

- TIMESTAMPLTZ
- TIMESTAMP WITH LOCAL TIME ZONE

TIMESTAMP_NTZ TIMESTAMP_NTZ internally stores "wallclock" time with a specified precision. All operations are performed without taking any time zone into account.

If the output format contains a time zone, the UTC indicator (Z) is displayed.

TIMESTAMP_NTZ is the default for TIMESTAMP.

Aliases for TIMESTAMP_NTZ:

- TIMESTAMPNTZ
- TIMESTAMP WITHOUT TIME ZONE
- DATETIME

TIMESTAMP_TZ TIMESTAMP_TZ internally stores UTC time together with an associated *time zone offset*. When a time zone isn't provided, the session time zone offset is used. All operations are performed with the time zone offset specific to each record.

Aliases for TIMESTAMP_TZ:

- TIMESTAMP WITH TIME ZONE

TIMESTAMP_TZ values are compared based on their times in UTC. For example, the following comparison between different times in different timezones returns TRUE because the two values have equivalent times in UTC.

```
SELECT '2024-01-01 00:00:00 +0000'::TIMESTAMP_TZ = '2024-01-01 01:00:00 +0100'::TIME
```

Attention

TIMESTAMP_TZ currently only stores the *offset* of a given time zone, not the actual *time zone*, at the moment of creation for a given value. This is especially important for daylight saving time, which is not utilized by UTC.

For example, with the TIMEZONE parameter set to `"America/Los_Angeles"`, converting a value to TIMESTAMP_TZ in January of a given year stores the time zone offset of `-0800`. If six months are later added to the value, the `-0800` offset is retained, even though in July the offset for Los Angeles is `-0700`. This is because, after the value is created, the actual time zone information (`"America/Los_Angeles"`) is no longer available. The following code sample illustrates this behavior:

```
SELECT '2024-01-01 12:00:00'::TIMESTAMP_TZ;
```

```
+-----------------------------------+
| '2024-01-01 12:00:00'::TIMESTAMP_TZ |
```

```
SELECT DATEADD(MONTH, 6, '2024-01-01 12:00:00'::TIMESTAMP_TZ);
```

```
+-----------------------------------------------------------+
| DATEADD(MONTH, 6, '2024-01-01 12:00:00'::TIMESTAMP_TZ) |
|-----------------------------------------------------------|
| 2024-07-01 12:00:00.000 -0800                             |
+-----------------------------------------------------------+
```

TIMESTAMP

TIMESTAMP in Snowflake is a user-specified alias associated with one of the TIMESTAMP_* variations. In all operations where TIMESTAMP is used, the associated TIMESTAMP_* variation is automatically used. The TIMESTAMP data type is never stored in tables.

The TIMESTAMP_* variation associated with TIMESTAMP is specified by the TIMESTAMP_TYPE_MAPPING session parameter. The default is TIMESTAMP_NTZ.

All timestamp variations, as well as the TIMESTAMP alias, support an optional precision parameter for fractional seconds (for example, `TIMESTAMP(3)`). Timestamp precision can range from 0 (seconds) to 9 (nanoseconds). The default precision is 9.

First, create a table with a TIMESTAMP column (mapped to TIMESTAMP_NTZ):

```
ALTER SESSION SET TIMESTAMP_TYPE_MAPPING = TIMESTAMP_NTZ;

CREATE OR REPLACE TABLE ts_test(ts TIMESTAMP);

DESC TABLE ts_test;
```

```
+------+------------------+--------+-------+---------+-------------+------------+-------+------------+
| name | type             | kind   | null? | default | primary key | unique key | check | expression |
|------+------------------+--------+-------+---------+-------------+------------+-------+------------|
| TS   | TIMESTAMP_NTZ(9) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       |
+------+------------------+--------+-------+---------+-------------+------------+-------+------------+
```

Next, explicitly use one of the TIMESTAMP variations (TIMESTAMP_LTZ):

```
CREATE OR REPLACE TABLE ts_test(ts TIMESTAMP_LTZ);

DESC TABLE ts_test;
```

```
+------+------------------+--------+-------+---------+-------------+------------+-------+------------+
| name | type             | kind   | null? | default | primary key | unique key | check | expression |
|------+------------------+--------+-------+---------+-------------+------------+-------+------------|
| TS   | TIMESTAMP_LTZ(9) | COLUMN | Y     | NULL    | N           | N          | NULL  | NULL       |
```

Use TIMESTAMP_LTZ with different time zones:

```
CREATE OR REPLACE TABLE ts_test(ts TIMESTAMP_LTZ);

ALTER SESSION SET TIMEZONE = 'America/Los_Angeles';

INSERT INTO ts_test VALUES('2024-01-01 16:00:00');
INSERT INTO ts_test VALUES('2024-01-02 16:00:00 +00:00');
```

This query shows that the time for January 2nd is 08:00 in Los Angeles (which is 16:00 in UTC):

```
SELECT ts, HOUR(ts) FROM ts_test;
```

```
+-------------------------------+-----------+
| TS                            | HOUR(TS)  |
|-------------------------------+-----------|
| 2024-01-01 16:00:00.000 -0800 |        16 |
| 2024-01-02 08:00:00.000 -0800 |         8 |
+-------------------------------+-----------+
```

Next, note that the times change with a different time zone:

```
ALTER SESSION SET TIMEZONE = 'America/New_York';
```

```
+-------------------------------+----------+
| TS                            | HOUR(TS) |
|-------------------------------+----------|
| 2024-01-01 19:00:00.000 -0500 |       19 |
| 2024-01-02 11:00:00.000 -0500 |       11 |
+-------------------------------+----------+
```

Create a table and use TIMESTAMP_NTZ:

```
CREATE OR REPLACE TABLE ts_test(ts TIMESTAMP_NTZ);

ALTER SESSION SET TIMEZONE = 'America/Los_Angeles';

INSERT INTO ts_test VALUES('2024-01-01 16:00:00');
INSERT INTO ts_test VALUES('2024-01-02 16:00:00 +00:00');
```

Note that both times from different time zones are converted to the same “wallclock” time:

```
SELECT ts, HOUR(ts) FROM ts_test;
```

```
+-------------------------+----------+
| TS                      | HOUR(TS) |
|-------------------------+----------|
| 2024-01-01 16:00:00.000 |       16 |
| 2024-01-02 16:00:00.000 |       16 |
```

Next, note that changing the session time zone doesn't affect the results:

```
ALTER SESSION SET TIMEZONE = 'America/New_York';

SELECT ts, HOUR(ts) FROM ts_test;
```

```
+-------------------------+----------+
| TS                      | HOUR(TS) |
|-------------------------+----------|
| 2024-01-01 16:00:00.000 |       16 |
| 2024-01-02 16:00:00.000 |       16 |
+-------------------------+----------+
```

Create a table and use TIMESTAMP_TZ:

```
CREATE OR REPLACE TABLE ts_test(ts TIMESTAMP_TZ);

ALTER SESSION SET TIMEZONE = 'America/Los_Angeles';

INSERT INTO ts_test VALUES('2024-01-01 16:00:00');
INSERT INTO ts_test VALUES('2024-01-02 16:00:00 +00:00');
```

Note that the January 1st record inherited the session time zone, and `America/Los_Angeles` was converted to a numeric time zone offset:

```
+-------------------------------+----------+
| TS                            | HOUR(TS) |
|-------------------------------+----------|
| 2024-01-01 16:00:00.000 -0800 |       16 |
| 2024-01-02 16:00:00.000 +0000 |       16 |
+-------------------------------+----------+
```

Next, note that changing the session time zone doesn't affect the results:

```
ALTER SESSION SET TIMEZONE = 'America/New_York';

SELECT ts, HOUR(ts) FROM ts_test;
```

```
+-------------------------------+----------+
| TS                            | HOUR(TS) |
|-------------------------------+----------|
| 2024-01-01 16:00:00.000 -0800 |       16 |
| 2024-01-02 16:00:00.000 +0000 |       16 |
+-------------------------------+----------+
```

Supported calendar

Snowflake uses the Gregorian Calendar for all dates and timestamps. The Gregorian Calendar starts in the year 1582, but

Date and time formats

All of these data types accept most non-ambiguous date, time, or date + time formats. See Supported formats for AUTO detection for the formats that Snowflake recognizes when configured to detect the format automatically.

You can also specify the date and time format manually. When specifying the format, you can use the case-insensitive elements listed in the following table:

Format element	Description
`YYYY`	Four-digit year.
`YY`	Two-digit year, controlled by the TWO_DIGIT_CENTURY_START session parameter. For example, when set to `1980`, values of `79` and `80` are parsed as `2079` and `1980` respectively.
`MM`	Two-digit month (`01` = January, and so on).
`MON`	Full or abbreviated month name.
`MMMM`	Full month name.
`DD`	Two-digit day of month (`01` through `31`).
`DY`	Abbreviated day of week.
`HH24`	Two digits for hour (`00` through `23`). You ***must not*** specify `AM` / `PM`.
`HH12`	Two digits for hour (`01` through `12`). You can specify `AM` / `PM`.

Format element	Description
`MI`	Two digits for minute (`00` through `59`).
`SS`	Two digits for second (`00` through `59`).
`FF[0-9]`	Fractional seconds with precision `0` (seconds) to `9` (nanoseconds), e.g. `FF`, `FF0`, `FF3`, `FF9`. Specifying `FF` is equivalent to `FF9` (nanoseconds).
`TZH:TZM`, `TZHTZM`, `TZH`	Time zone hour and minute, offset from UTC. Can be prefixed by `+`/`-` for sign.
`UUUU`	Four-digit year in ISO format, which are negative for BCE years.

Note

- When a date-only format is used, the associated time is assumed to be midnight on that day.
- Anything in the format between double quotes or other than the above elements is parsed/formatted without being interpreted.
- For more details about valid ranges, number of digits, and best practices, see Additional information about using date, time, and timestamp formats.

Examples of using date and time formats

```
CREATE OR REPLACE TABLE timestamp_demo_table(
  tstmp TIMESTAMP,
  tstmp_tz TIMESTAMP_TZ,
  tstmp_ntz TIMESTAMP_NTZ,
  tstmp_ltz TIMESTAMP_LTZ);
INSERT INTO timestamp_demo_table (tstmp, tstmp_tz, tstmp_ntz, tstmp_ltz) VALUES (
  '2024-03-12 01:02:03.123456789',
  '2024-03-12 01:02:03.123456789',
  '2024-03-12 01:02:03.123456789',
  '2024-03-12 01:02:03.123456789');
```

```
ALTER SESSION SET TIMESTAMP_OUTPUT_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';
ALTER SESSION SET TIMESTAMP_TZ_OUTPUT_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';
ALTER SESSION SET TIMESTAMP_NTZ_OUTPUT_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';
ALTER SESSION SET TIMESTAMP_LTZ_OUTPUT_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';
```

```
SELECT tstmp, tstmp_tz, tstmp_ntz, tstmp_ltz
  FROM timestamp_demo_table;
```

```
+-------------------------------+-------------------------------+-------------------------------+-----
| TSTMP                         | TSTMP_TZ                      | TSTMP_NTZ                     | TSTM
|-------------------------------+-------------------------------+-------------------------------+-----
| 2024-03-12 01:02:03.123456789 | 2024-03-12 01:02:03.123456789 | 2024-03-12 01:02:03.123456789 | 2024
+-------------------------------+-------------------------------+-------------------------------+-----
```

Date and time constants

Constants (also known as *literals*) are fixed data values. Snowflake supports using string constants to specify fixed date, time, or timestamp values. String constants must always be enclosed between delimiter characters. Snowflake supports using single quotes to delimit string constants.

For example:

```
DATE '2024-08-14'
TIME '10:03:56'
TIMESTAMP '2024-08-15 10:59:43'
```

The string is parsed as a DATE, TIME, or TIMESTAMP value based on the input format for the data type, as set through the following parameters:

DATE	DATE_INPUT_FORMAT
TIME	TIME_INPUT_FORMAT
TIMESTAMP	TIMESTAMP_INPUT_FORMAT

For example, to insert a specific date into a column in a table:

```
CREATE TABLE t1 (d1 DATE);

INSERT INTO t1 (d1) VALUES (DATE '2024-08-15');
```

Interval constants

You can use interval constants to add or subtract a period of time to or from a date, time, or timestamp. Interval constants are implemented using the INTERVAL keyword, which has the following syntax:

```
{ + | - } INTERVAL '<integer> [ <date_time_part> ] [ , <integer> [ <date_time_part> ] ... ]'
```

As with all string constants, Snowflake requires single quotes to delimit interval constants.

The INTERVAL keyword supports one more integers and, optionally, one or more date or time parts. For example:

- `INTERVAL '1 year'` represents one year.
- `INTERVAL '4 years, 5 months, 3 hours'` represents four years, five months, and three hours.

If a date or time part isn't specified, the interval represents seconds (for example, `INTERVAL '2'` is the same as `INTERVAL '2 seconds'`). Note that this is different from the default unit of time for performing date arithmetic. For more details, see Simple arithmetic for dates.

For the list of supported date and time parts, see Supported Date and Time Parts for Intervals.

Note

- The order of interval increments is important. The increments are added or subtracted in the order listed. For example:
  - `INTERVAL '1 year, 1 day'` first adds or subtracts a year and then a day.

```
SELECT TO_DATE ('2019-02-28') + INTERVAL '1 day, 1 year';
```

```
+-----------------------------------------------------+
| TO_DATE ('2019-02-28') + INTERVAL '1 DAY, 1 YEAR' |
|-----------------------------------------------------|
| 2020-03-01                                          |
+-----------------------------------------------------+
```

```
SELECT TO_DATE ('2019-02-28') + INTERVAL '1 year, 1 day';
```

```
+-----------------------------------------------------+
| TO_DATE ('2019-02-28') + INTERVAL '1 YEAR, 1 DAY' |
|-----------------------------------------------------|
| 2020-02-29                                          |
+-----------------------------------------------------+
```

- INTERVAL is not a data type (that is, you can't define a table column to be of data type INTERVAL). Intervals can only be used in date, time, and timestamp arithmetic.
- You can't use an interval with a SQL variable. For example, the following query returns an error:

```
SET v1 = '1 year';

SELECT TO_DATE('2023-04-15') + INTERVAL $v1;
```

Supported date and time parts for intervals

The INTERVAL keyword supports the following date and time parts as arguments (case-insensitive):

Date or Time Part	Abbreviations / Variations
`year`	`y`, `yy`, `yyy`, `yyyy`, `yr`, `years`, `yrs`
`quarter`	`q`, `qtr`, `qtrs`, `quarters`
`month`	`mm`, `mon`, `mons`, `months`
`week`	`w`, `wk`, `weekofyear`, `woy`, `wy`, `weeks`
`day`	`d`, `dd`, `days`, `dayofmonth`
`hour`	`h`, `hh`, `hr`, `hours`, `hrs`
`minute`	`m`, `mi`, `min`, `minutes`, `mins`
`second`	`s`, `sec`, `seconds`, `secs`
`millisecond`	`ms`, `msec`, `milliseconds`
`microsecond`	`us`, `usec`, `microseconds`
`nanosecond`	`ns`, `nsec`, `nanosec`, `nsecond`, `nanoseconds`, `nanosecs`, `nseconds`

Interval examples

Add a year interval to a specific date:

```
SELECT TO_DATE('2023-04-15') + INTERVAL '1 year';
```

```
+-------------------------------------------+
| TO_DATE('2023-04-15') + INTERVAL '1 YEAR' |
|-------------------------------------------|
| 2024-04-15                                |
+-------------------------------------------+
```

Add an interval of 3 hours and 18 minutes to a specific time:

```
SELECT TO_TIME('04:15:29') + INTERVAL '3 hours, 18 minutes';
```

```
+-----------------------------------------------------+
| TO_TIME('04:15:29') + INTERVAL '3 HOURS, 18 MINUTES' |
|-----------------------------------------------------|
| 07:33:29                                            |
+-----------------------------------------------------+
```

Add a complex interval to the output of the CURRENT_TIMESTAMP function:

```
SELECT CURRENT_TIMESTAMP + INTERVAL
    '1 year, 3 quarters, 4 months, 5 weeks, 6 days, 7 minutes, 8 seconds,
    1000 milliseconds, 4000000 microseconds, 5000000001 nanoseconds'
  AS complex_interval1;
```

The following is sample output. The output is different when the current timestamp is different.

```
+-------------------------------+
| COMPLEX_INTERVAL1             |
|-------------------------------|
| 2026-11-07 18:07:19.875000001 |
+-------------------------------+
```

Add a complex interval with abbreviated date/time part notation to a specific date:

```
SELECT TO_DATE('2025-01-17') + INTERVAL
    '1 y, 3 q, 4 mm, 5 w, 6 d, 7 h, 9 m, 8 s,
    1000 ms, 445343232 us, 898498273498 ns'
  AS complex_interval2;
```

```
+-------------------------------+
| COMPLEX_INTERVAL2             |
|-------------------------------|
| 2027-03-30 07:31:32.841505498 |
```

Query a table of employee information and return the names of employees who were hired within the past two years and three months:

```
SELECT name, hire_date
  FROM employees
  WHERE hire_date > CURRENT_DATE - INTERVAL '2 y, 3 month';
```

Filter a TIMESTAMP column named `ts` from a table named `t1` and add four seconds to each returned value:

```
SELECT ts + INTERVAL '4 seconds'
  FROM t1
  WHERE ts > TO_TIMESTAMP('2024-04-05 01:02:03');
```

Simple arithmetic for dates

In addition to using interval constants to add to and subtract from dates, times, and timestamps, you can also add and subtract days to and from DATE values, in the form of `{ + | - }` `<integer>`, where `<integer>` specifies the number of days to add or subtract.

Note

TIME and TIMESTAMP values don't yet support simple arithmetic.

Add one day to a specific date:

```
SELECT TO_DATE('2024-04-15') + 1;
```

```
+----------------------------+
| TO_DATE('2024-04-15') + 1 |
|----------------------------|
| 2024-04-16                 |
+----------------------------+
```

Subtract four days from a specific date:

```
SELECT TO_DATE('2024-04-15') - 4;
```

```
+----------------------------+
| TO_DATE('2024-04-15') - 4 |
|----------------------------|
| 2024-04-11                 |
+----------------------------+
```

Query a table named `employees` and return the names of people who left the company, but were employed more than 365 days:

```
    WHERE end_date > start_date + 365;
```