Working with date and time values

Date and time calculations are among the most widely used and most critical computations in analytics and data mining. This topic provides practical examples of common date and time queries and calculations.

Loading dates and timestamps

This section provides examples for loading date and timestamp values, and describes considerations related to time zones when loading these values.

Loading timestamps with no time zone attached

In the following example, the TIMESTAMP_TYPE_MAPPING parameter is set to `TIMESTAMP_LTZ` (local time zone). The TIMEZONE parameter is set to `America/Chicago` time. If some incoming timestamps don't have a specified time zone, then Snowflake loads those strings assuming the timestamps represent local time in the time zone set for the TIMEZONE parameter.

```
ALTER SESSION SET TIMESTAMP_TYPE_MAPPING = 'TIMESTAMP_LTZ';
ALTER SESSION SET TIMEZONE = 'America/Chicago';
```

```
SELECT * FROM time;
```

```
+-------------------------------+
| LTZ                           |
|-------------------------------|
| 2024-05-01 00:00:00.000 -0500 |
+-------------------------------+
```

Loading timestamps with a time zone attached

In the following example, the TIMESTAMP_TYPE_MAPPING parameter is set to `TIMESTAMP_LTZ` (local time zone). The TIMEZONE parameter is set to `America/Chicago` time. If some incoming timestamps have a different time zone specified, Snowflake loads the string in `America/Chicago` time.

```
ALTER SESSION SET TIMESTAMP_TYPE_MAPPING = 'TIMESTAMP_LTZ';
ALTER SESSION SET TIMEZONE = 'America/Chicago';

CREATE OR REPLACE TABLE time (ltz TIMESTAMP);
INSERT INTO time VALUES ('2024-04-30 19:00:00.000 -0800');

SELECT * FROM time;
```

```
+-------------------------------+
```

```
| 2024-04-30 22:00:00.000 -0500 |
+-------------------------------+
```

Converting timestamps to alternative time zones

In the following example, a set of timestamp values is stored with no time zone data. The timestamps are loaded in UTC time and converted to other time zones:

```
ALTER SESSION SET TIMEZONE = 'UTC';
ALTER SESSION SET TIMESTAMP_LTZ_OUTPUT_FORMAT = 'YYYY-MM-DD HH24:MI:SS TZH:TZM';

CREATE OR REPLACE TABLE utctime (ntz TIMESTAMP_NTZ);
INSERT INTO utctime VALUES ('2024-05-01 00:00:00.000');
```

```
SELECT * FROM utctime;
```

```
+-------------------------+
| NTZ                     |
|-------------------------|
| 2024-05-01 00:00:00.000 |
+-------------------------+
```

```
  FROM utctime;
```

```
+---------------------------+
| CHICAGOTIME               |
|---------------------------|
| 2024-04-30 19:00:00 +0000 |
+---------------------------+
```

```
SELECT CONVERT_TIMEZONE('UTC','America/Los_Angeles', ntz)::TIMESTAMP_LTZ AS LATime
  FROM utctime;
```

```
+---------------------------+
| LATIME                    |
|---------------------------|
| 2024-04-30 17:00:00 +0000 |
+---------------------------+
```

Inserting valid date strings into date columns in a table

This example inserts values into a DATE column.

```
CREATE OR REPLACE TABLE my_table(id INTEGER, date1 DATE);
```

```
SELECT id, date1
  FROM my_table
  ORDER BY id;
```

```
+----+------------+
| ID | DATE1      |
|----+------------|
|  1 | 2024-07-23 |
|  2 | NULL       |
+----+------------+
```

The TO_DATE function accepts TIMESTAMP values and strings in TIMESTAMP format, but discards the time information (hours, minutes, and so on).

```
INSERT INTO my_table(id, date1) VALUES
  (3, TO_DATE('2024.02.20 11:15:00', 'YYYY.MM.DD HH:MI:SS')),
  (4, TO_TIMESTAMP('2024.02.24 04:00:00', 'YYYY.MM.DD HH:MI:SS'));
```

```
SELECT id, date1
  FROM my_table
  WHERE id >= 3;
```

```
. . .
```

```
|  4 | 2024-02-24 |
+----+------------+
```

If you insert a DATE that was defined with only a time, then the default date is January 1, 1970.

```
INSERT INTO my_table(id, date1) VALUES
  (5, TO_DATE('11:20:30', 'hh:mi:ss'));
```

```
SELECT id, date1
  FROM my_table
  WHERE id = 5;
```

```
+----+------------+
| ID | DATE1      |
|----+------------|
|  5 | 1970-01-01 |
+----+------------+
```

When you retrieve DATE values, you can format them as TIMESTAMP values:

```
SELECT id,
       TO_VARCHAR(date1, 'dd-mon-yyyy hh:mi:ss') AS date1
  FROM my_table
  ORDER BY id;
```

```
+----+----------------------+
| ID | DATE1                |
|----+----------------------|
|  1 | 23-Jul-2024 00:00:00 |
|  2 | NULL                 |
|  3 | 20-Feb-2024 00:00:00 |
|  4 | 24-Feb-2024 00:00:00 |
|  5 | 01-Jan-1970 00:00:00 |
+----+----------------------+
```

Retrieving the current date and time

Get the current date as a DATE value:

```
SELECT CURRENT_DATE();
```

Get the current date and time as a TIMESTAMP value:

```
SELECT CURRENT_TIMESTAMP();
```

```
SELECT EXTRACT('dayofweek', CURRENT_DATE());
```

Note

- The `dayofweek_iso` part follows the ISO-8601 data elements and interchange formats standard. The function returns the day of the week as an integer value in the range 1-7, where 1 represents Monday.
- For compatibility with some other systems, the `dayofweek` part follows the UNIX standard. The function returns the day of the week as an integer value in the range 0-6, where 0 represents Sunday.

You can get the current day of the week as a string using the TO_VARCHAR or DECODE function.

Run a query that returns the short English name (for example, `Sun`, `Mon`, and so on) for the current date:

```
SELECT TO_VARCHAR(CURRENT_DATE(), 'dy');
```

Run a query that returns the explicitly-provided weekday names for the current date:

```
SELECT DECODE(EXTRACT('dayofweek_iso', CURRENT_DATE()),
  1, 'Monday',
  2, 'Tuesday',
  3, 'Wednesday',
  4, 'Thursday',
  5, 'Friday',
  6, 'Saturday',
```

Retrieving date and time parts

You can get various date and time parts for the current date and time using the DATE_PART function.

Query for the current day of the month:

```
SELECT DATE_PART(day, CURRENT_TIMESTAMP());
```

Query for the current year:

```
SELECT DATE_PART(year, CURRENT_TIMESTAMP());
```

Query for the current month:

```
SELECT DATE_PART(month, CURRENT_TIMESTAMP());
```

Query for the current hour:

```
SELECT DATE_PART(hour, CURRENT_TIMESTAMP());
```

Query for the current minute:

```
SELECT DATE_PART(minute, CURRENT_TIMESTAMP());
```

Query for the current second:

```
SELECT DATE_PART(second, CURRENT_TIMESTAMP());
```

You can also use the EXTRACT function to get various date and time parts for the current date and time.

Query for the current day of the month:

```
SELECT EXTRACT('day', CURRENT_TIMESTAMP());
```

Query for the current year:

```
SELECT EXTRACT('year', CURRENT_TIMESTAMP());
```

Query for the current month:

```
SELECT EXTRACT('month', CURRENT_TIMESTAMP());
```

```
SELECT EXTRACT('hour', CURRENT_TIMESTAMP());
```

Query for the current minute:

```
SELECT EXTRACT('minute', CURRENT_TIMESTAMP());
```

Query for the current second:

```
SELECT EXTRACT('second', CURRENT_TIMESTAMP());
```

This query returns tabular output with various date and time parts for the current date and time:

```
SELECT month(CURRENT_TIMESTAMP()) AS month,
       day(CURRENT_TIMESTAMP()) AS day,
       hour(CURRENT_TIMESTAMP()) AS hour,
       minute(CURRENT_TIMESTAMP()) AS minute,
       second(CURRENT_TIMESTAMP()) AS second;
```

```
+-------+-----+------+--------+--------+
| MONTH | DAY | HOUR | MINUTE | SECOND |
|-------+-----+------+--------+--------|
|     8 |  28 |    7 |     59 |     28 |
```

Calculating business calendar dates and times

Get the first day of the month as a DATE value using the DATE_TRUNC function. For example, get the first day of the current month:

```
SELECT DATE_TRUNC('month', CURRENT_DATE());
```

Get the last day of the current month as a DATE value using the DATEADD and DATE_TRUNC functions:

```
SELECT DATEADD('day',
               -1,
               DATE_TRUNC('month', DATEADD(day, 31, DATE_TRUNC('month',CURRENT_DATE()))));
```

For an alternative option, the following example uses DATE_TRUNC to retrieve the beginning of the current month, adds one month to retrieve the beginning of the next month, and then subtracts one day to determine the last day of the current month.

```
SELECT DATEADD('day',
               -1,
               DATEADD('month', 1, DATE_TRUNC('month', CURRENT_DATE())));
```

Get the last day of the previous month as a DATE value:

```
                DATE_TRUNC('month', CURRENT_DATE()));
```

Get the short English name (for example, `Jan`, `Dec`, and so on) for the current month:

```
SELECT TO_VARCHAR(CURRENT_DATE(), 'Mon');
```

Get the current month name using explicitly-provided month names:

```
SELECT DECODE(EXTRACT('month', CURRENT_DATE()),
         1, 'January',
         2, 'February',
         3, 'March',
         4, 'April',
         5, 'May',
         6, 'June',
         7, 'July',
         8, 'August',
         9, 'September',
         10, 'October',
         11, 'November',
         12, 'December') AS current_month;
```

Get the date for Monday in the current week:

```
SELECT DATEADD('day',
               (EXTRACT('dayofweek_iso', CURRENT_DATE()) * -1) + 1,
```

Get the date for Friday in the current week:

```
SELECT DATEADD('day',
               (5 - EXTRACT('dayofweek_iso', CURRENT_DATE())),
               CURRENT_DATE());
```

Get the date for the first Monday in the current month using the DATE_PART function:

```
SELECT DATEADD(day,
               MOD( 7 + 1 - DATE_PART('dayofweek_iso', DATE_TRUNC('month', CURRENT_DATE())), 7),
               DATE_TRUNC('month', CURRENT_DATE()));
```

Note

In the above query, the `1` value in `7 + 1` translates to Monday. To retrieve the date for the first Tuesday, Wednesday, and so on, substitute `2`, `3`, and so on, respectively, through `7` for `Sunday`.

Get the first day of the current year as a DATE value:

```
SELECT DATE_TRUNC('year', CURRENT_DATE());
```

Get the last day of the current year as a DATE value:

```
                DATEADD('year', 1, DATE_TRUNC('year', CURRENT_DATE())));
```

Get the last day of the previous year as a DATE value:

```
SELECT DATEADD('day',
               -1,
               DATE_TRUNC('year', CURRENT_DATE()));
```

Get the first day of the current quarter as a DATE value:

```
SELECT DATE_TRUNC('quarter', CURRENT_DATE());
```

Get the last day of the current quarter as a DATE value:

```
SELECT DATEADD('day',
               -1,
               DATEADD('month', 3, DATE_TRUNC('quarter', CURRENT_DATE())));
```

Get the date and timestamp for midnight in the current day:

```
SELECT DATE_TRUNC('day', CURRENT_TIMESTAMP());
```

```
| Wed, 07 Sep 2016 00:00:00 -0700         |
+-----------------------------------------+
```

Incrementing date and time values

Use the DATEADD function to increment date and time values.

Add two years to the current date:

```
SELECT DATEADD(year, 2, CURRENT_DATE());
```

Add two days to the current date:

```
SELECT DATEADD(day, 2, CURRENT_DATE());
```

Add two hours to the current date and time:

```
SELECT DATEADD(hour, 2, CURRENT_TIMESTAMP());
```

Add two minutes to the current date and time:

Add two seconds to the current date and time:

```
SELECT DATEADD(second, 2, CURRENT_TIMESTAMP());
```

Converting valid character strings to dates, times, or timestamps

In most use cases, Snowflake correctly handles date and timestamp values formatted as strings. In certain cases, such as string-based comparisons or when a result depends on a different timestamp format from the format set in the session parameters, we recommend explicitly converting values to the desired format to avoid unexpected results.

For example, without explicit casting, comparing string values produces string-based results:

```
CREATE OR REPLACE TABLE timestamps(timestamp1 STRING);

INSERT INTO timestamps VALUES
  ('Fri, 05 Apr 2013 00:00:00 -0700'),
  ('Sat, 06 Apr 2013 00:00:00 -0700'),
  ('Sat, 01 Jan 2000 00:00:00 -0800'),
  ('Wed, 01 Jan 2020 00:00:00 -0800');
```

The following query performs a comparison without explicit casting:

```
+-------------+
| TIMESTAMP1  |
|-------------|
+-------------+
```

The following query performs a comparison with explicit casting to DATE:

```
SELECT * FROM timestamps WHERE timestamp1 < '2014-01-01'::DATE;
```

```
+---------------------------------+
| DATE1                           |
|---------------------------------|
| Fri, 05 Apr 2013 00:00:00 -0700 |
| Sat, 06 Apr 2013 00:00:00 -0700 |
| Sat, 01 Jan 2000 00:00:00 -0800 |
+---------------------------------+
```

For more information about conversion functions, see Date and time formats in conversion functions.

Applying date arithmetic to date strings

Add five days to the date expressed in a string:

```
                TO_TIMESTAMP('12-jan-2024 00:00:00','dd-mon-yyyy hh:mi:ss'))
  AS add_five_days;
```

```
+-------------------------+
| ADD_FIVE_DAYS           |
|-------------------------|
| 2024-01-17 00:00:00.000 |
+-------------------------+
```

You can calculate the difference in days between the current date and the date expressed in a string using the DATEDIFF function.

Calculate the difference in days using the TO_TIMESTAMP function:

```
SELECT DATEDIFF('day',
                TO_TIMESTAMP ('12-jan-2024 00:00:00','dd-mon-yyyy hh:mi:ss'),
                CURRENT_DATE())
  AS to_timestamp_difference;
```

```
+-------------------------+
| TO_TIMESTAMP_DIFFERENCE |
|-------------------------|
|                     229 |
+-------------------------+
```

```
SELECT DATEDIFF('day',
                TO_DATE ('12-jan-2024 00:00:00','dd-mon-yyyy hh:mi:ss'),
                CURRENT_DATE())
  AS to_date_difference;
```

```
+--------------------+
| TO_DATE_DIFFERENCE |
|--------------------|
|                229 |
+--------------------+
```

Add one day to a specified date:

```
SELECT TO_DATE('2024-01-15') + 1 AS date_plus_one;
```

```
+---------------+
| DATE_PLUS_ONE |
|---------------|
| 2024-01-16    |
+---------------+
```

Subtract nine days from the current date (for example, Aug 28, 2024):

```
+------------------+
| DATE_MINUS_NINE  |
|------------------|
| 2024-08-19       |
+------------------+
```

Calculating differences between dates or times

Calculate the difference between the current date and the date in three years:

```
SELECT DATEDIFF(year, CURRENT_DATE(),
       DATEADD(year, 3, CURRENT_DATE()));
```

Calculate the difference between the current date and the date in three months:

```
SELECT DATEDIFF(month, CURRENT_DATE(),
       DATEADD(month, 3, CURRENT_DATE()));
```

Calculate the difference between the current date and the date in three days:

```
SELECT DATEDIFF(day, CURRENT_DATE(),
```

Calculate the difference between the current time and the time in three hours:

```
SELECT DATEDIFF(hour, CURRENT_TIMESTAMP(),
       DATEADD(hour, 3, CURRENT_TIMESTAMP()));
```

Calculate the difference between the current time and the time in three minutes:

```
SELECT DATEDIFF(minute, CURRENT_TIMESTAMP(),
       DATEADD(minute, 3, CURRENT_TIMESTAMP()));
```

Calculate the difference between the current time and the time in three seconds:

```
SELECT DATEDIFF(second, CURRENT_TIMESTAMP(),
       DATEADD(second, 3, CURRENT_TIMESTAMP()));
```

Creating yearly calendar views

```
CREATE OR REPLACE VIEW calendar_2016 AS
  SELECT n,
         theDate,
         DECODE (EXTRACT('dayofweek',theDate),
```

```
            5 , 'Friday',
            6 , 'Saturday',
            0 , 'Sunday') theDayOfTheWeek,
          DECODE (EXTRACT(month FROM theDate),
            1 , 'January',
            2 , 'February',
            3 , 'March',
            4 , 'April',
            5 , 'May',
            6 , 'June',
            7 , 'July',
            8 , 'August',
            9 , 'september',
            10, 'October',
            11, 'November',
            12, 'December') theMonth,
          EXTRACT(year FROM theDate) theYear
  FROM
    (SELECT ROW_NUMBER() OVER (ORDER BY seq4()) AS n,
            DATEADD(day, ROW_NUMBER() OVER (ORDER BY seq4())-1, TO_DATE('2016-01-01')) AS theDate
      FROM table(generator(rowCount => 365)))
  ORDER BY n ASC;

SELECT * from CALENDAR_2016;
```

```
+-----+------------+-----------------+-----------+---------+
|   N | THEDATE    | THEDAYOFTHEWEEK | THEMONTH  | THEYEAR |
|-----+------------+-----------------+-----------+---------|
|   1 | 2016-01-01 | Friday          | January   |    2016 |
|   2 | 2016-01-02 | Saturday        | January   |    2016 |
  ...
| 364 | 2016-12-29 | Thursday        | December  |    2016 |
```