Snowflake Standards

Kirk Wilson

This is a cheat sheet simplifying the different data types in Snowflake, and what I generally share with clients who are new to Snowflake. I've found that because Snowflake has so many synonyms for data types (NUMBER, DECIMAL, NUMERIC, INT, INTEGER, BIGINT, <and more> are all effectively the same thing), this helps to cut through the noise.

- Data Types - Quick Reference
  - Numbers
    - Whole numbers: NUMBER
    - Decimal numbers: NUMBER(38, <scale>), where <scale> is the required accuracy to the right of the decimal point
  - Text
    - VARCHAR
  - Dates
    - DATETIME
  - Semi-structured Data
    - VARIANT
- Data Types - Details
  - Numbers - https://docs.snowflake.net/manuals/sql-reference/data-types-numeric.html
    - Snowflake supports both fixed point (NUMBER) and floating point (FLOAT) numbers. Unless there is a very clear and specific requirement for floating point numbers (which is generally rare), the NUMBER data type will be used for all numeric data in Snowflake.
    - If there is no need for decimal/fractional values, a column should be defined with the type NUMBER.
    - If there is a need for decimal/fractional values, a column should be defined with the type NUMBER(38, X), where X is the level of required accuracy to the right of the decimal point.
      - Any currency columns for example would be defined as NUMBER(38, 2).
    - Precision and Scale
      - Precision represents the total number of digits that can be stored in the column. This includes numbers to the left and right of the decimal point.
      - Scale represents the number of digits to the right of the decimal point.
      - If you define a column just as NUMBER, the default precision and scale will be (38, 0).
      - Per the Snowflake documentation: Precision (total number of digits) does not impact storage. However, scale (number of digits following the decimal point) does have an impact on storage. For example, the same value stored in a column of type NUMBER(10,5) consumes more space than NUMBER(5,0). Also, processing values with a larger scale could be slightly slower and consume more memory.
      - Examples:
        - CREATE TABLE NUMBER_EXAMPLE (NUM_COLUMN NUMBER);
          - You can insert any whole number (with no decimal/fractional values) up to 38 digits long.

            - Trying to insert a number with 39 or more digits would result in the Snowflake error "Number out of representable range"
            - Snowflake will allow you to insert a fractional value, such as 123.456, into the column without any errors. However, it will automatically truncate and round the number to 123.
        - CREATE TABLE NUMBER_EXAMPLE (NUM_COLUMN NUMBER(4, 0);
            - You can insert any number up to the value of 9999 into the column. Trying to insert the value 10000 (or anything higher) will result in an error.
        - CREATE TABLE NUMBER_EXAMPLE (NUM_COLUMN NUMBER(4, 4);
            - In this example, you can only insert fractional values. The value 0.1234 would be stored exactly. Snowflake will allow you to insert the value of 0.12345 without errors, but it will automatically round the number to 0.1235.
            - Since the precision and scale are the same, Snowflake only allows values to the right of the decimal point. Trying to insert the value 1 would result in a "Number of representable range" error. Trying to insert 1.1, or anything with a non-zero value to the left of the decimal point would result in the same error.
        - CREATE TABLE NUMBER_EXAMPLE (NUM_COLUMN NUMBER(7, 4);
            - This column allows 3 digits to the left of the decimal point (7 - 4 = 3) and 4 to the right. The value 123.4567 for example would be stored exactly.
            - Inserting a value of 1234.5678 would throw an error.
            - Inserting a value of 123.45678 will not throw an error, but Snowflake will automatically truncate and round the number to 123.4568
    - Characters - https://docs.snowflake.net/manuals/sql-reference/data-types-text.html
        - The VARCHAR type will be used for all text data in Snowflake.
            - Do not specify a length. Per the Snowflake documentation: A column only consumes storage for the amount of actual data stored. For example, a 1-character string in a VARCHAR(16777216) column only consumes a single character. There is no performance difference between using the full-length VARCHAR declaration VARCHAR(16777216) or a smaller size.
    - Dates - https://docs.snowflake.net/manuals/sql-reference/data-types-datetime.html
        - The DATETIME type will be used for all date/time data in Snowflake.
        - Note about time zones
            - Snowflake has support for three different time zone representations, described in the above documentation link. This can cause some minor frustration, but it's easy to work around any issues that arise.
            - TIMESTAMP_LTZ and TIMESTAMP_NTZ are the two primary date/time types with varying time zone information. The third type, TIMESTAMP_TZ will be ignored for our use cases.
                - TIMESTAMP_LTZ represents data in UTC time, also factoring in the current session's time zone.
                - TIMESTAMP_NTZ represents data in a manner that is agnostic of the session's time zone, and as such is a more generic and safer option. DATETIME is synonymous with TIMESTAMP_NTZ.
            - Examples:

- SELECT CURRENT_TIMESTAMP;
  - When run in the Pacific time zone, this returns a value of 2018-08-29 10:12:13.144 -0700 (for example). Note that it includes the -0700 UTC offset.
- CREATE TABLE DATE_EXAMPLE (DATE_COLUMN DATETIME);
  - Both of the following statements will insert data as expected:
    INSERT INTO DATE_EXAMPLE VALUES ('2018-08-29 10:04:16.34');
    INSERT INTO DATE_EXAMPLE VALUES ('2018-08-29');
  - The following statement will fail, since CURRENT_TIMESTAMP is in TIMESTAMP_LTZ format while the DATETIME type is equivalent to TIMESTAMP_NTZ.
    INSERT INTO DATE_EXAMPLE VALUES (CURRENT_TIMESTAMP);
    Error: Expression type does not match column data type, expecting TIMESTAMP_NTZ(9) but got TIMESTAMP_LTZ(9)
  - Converting CURRENT_TIMESTAMP to TIMESTAMP_NTZ format for storage in a DATETIME column can be successfully done by:
    INSERT INTO DATE_EXAMPLE VALUES (CURRENT_TIMESTAMP::TIMESTAMP_NTZ);
    - Note that the "::" operator is shorthand in Snowflake for casting a data type.
    - SELECT CURRENT_TIMESTAMP::VARCHAR; would convert the time to a string, for example.
- See more examples at https://docs.snowflake.net/manuals/user-guide/date-time-examples.html

- Semi-Structured Data (JSON, XML, and others) - https://docs.snowflake.net/manuals/sql-reference/data-types-semistructured.html
  - The VARIANT type will be used for all semi-structured data in Snowflake.
  - There are also OBJECT and ARRAY types which can be used for specific situations, but they're not as common in my experience.
  - If you need to load this type of data, just refer to the documentation since there are way too many different use cases for me to cover here.