Unstructured data types

Snowflake supports three different kinds of data:

- *Structured data* (such as a CSV file) follows a strict tabular schema. Structured data can be easily loaded into SQL tables.
- *Semi-structured data* (such as a JSON or XML file) has a flexible schema. Snowflake can access fields in semi-structured data using special functions, but the data is not as easily queried as structured data. Semi-structured data can be loaded into SQL tables using VARIANT columns.
- *Unstructured data* (such as a document, image, or audio file) has no inherent schema. Unstructured data might still have an internal structure (for example, PNG image files must follow a documented format) but such technical details do not generally relate to the information in the file.

Snowflake provides ways of working with data in unstructured files, such as the AI COMPLETE function and Document AI.

To use unstructured data in Snowflake, it must first be stored on an internal or external stage. The Snowflake function that processes the unstructured data reads it from there. Depending on the function, you specify the file in one or more of the following ways:

- By passing a stage name and a relative path to the file as two separate arguments to the function that will use it.
- By passing a staged or scoped URL as a string.
- By passing a FILE object created using the TO_FILE or TRY_TO_FILE function.

FILE data type

PREVIEW FEATURE — OPEN

Available to all accounts.

Snowflake provides the FILE data type for unstructured data. A FILE value represents a file stored in an internal or external stage, but does not store the file's data, only a reference to it. It includes the following metadata:

- STAGE: The name of the stage on which the file resides.
- RELATIVE_PATH: The relative path of the file in its stage.
- STAGE_FILE_URL: The stage file URL.
- SCOPED_FILE_URL: A scoped file URL.
- CONTENT_TYPE: The MIME type of the file.
- SIZE: The size, in bytes, of the file.
- ETAG: A unique hash of the file contents.
- LAST_MODIFIED: The timestamp at which the file was last modified.

Not all of these fields are required. A FILE must have CONTENT_TYPE, SIZE, ETAG, and LAST_MODIFIED fields, and also the file's location specified by STAGE plus RELATIVE_PATH, STAGE_FILE_URL, or SCOPED_FILE_URL.

You can create a file by passing a scoped file URL, a stage and path, or a metadata object to the TO_FILE or TRY_TO_FILE function.

Sub-category	Function
Constructors	TO_FILE
	TRY_TO_FILE
Accessors	FL_GET_CONTENT_TYPE
	FL_GET_ETAG
	FL_GET_FILE_TYPE
	FL_GET_LAST_MODIFIED
	FL_GET_RELATIVE_PATH
	FL_GET_SCOPED_FILE_URL
	FL_GET_SIZE
	FL_GET_STAGE
	FL_GET_STAGE_FILE_URL
Utility Functions	FL_IS_AUDIO
	FL_IS_COMPRESSED
	FL_IS_DOCUMENT

Sub-category	Function
	FL_IS_VIDEO

Usage notes

- FILE values may become inconsistent with the underlying staged files. FILE values are not updated when you modify or delete the underlying file. Conversely, if a FILE value is deleted from a table, the underlying file is not affected.
- Permissions on the underlying files are governed by the type of URL that was specified when creating the FILE. Stage file URLs and stage/path combinations give permanent permission to callers that have access to the associated stage. Scoped URLs give temporary user-based access to the underlying file for a 24-hour period.

Using unstructured data in Snowflake via SQL

Create a table with a FILE column.

```
CREATE TABLE images_table(img FILE);
```

Load data into the table from an external stage `my_images` that contains image files. `mpy_images` can be an internal or external stage.

```
ALTER STAGE my_images DIRECTORY=(ENABLE=true);
```

Load data into the Snowflake table.

```
INSERT INTO images_table
    SELECT TO_FILE(file_url) FROM DIRECTORY(@my_images);
```

Run SQL statements against `images_table`. For example, the following query returns the relative path of each file in the table that was last modified between January 1, 2021 and January 1, 2023.

```
SELECT FL_GET_RELATIVE_PATH(f)
    FROM images_table
    WHERE FL_GET_LAST_MODIFIED(f) BETWEEN '2021-01-01' and '2023-01-01';
```

Known limitations

The FILE data type currently cannot be used in:

- CLUSTER BY, GROUP BY, and ORDER BY clauses
- Hybrid tables, Iceberg tables, and external tables
- SnowScript

- Search optimization
- Clients and connectors except Snowpark Python