Vector data types

This topic describes the vector data types.

Data types

Snowflake supports a single vector data type, VECTOR.

> **Note**
> The VECTOR data type is only supported in SQL, the Python connector, and the Snowpark Python library. No other languages are supported.

VECTOR

With the VECTOR data type, Snowflake encodes and processes vectors efficiently. This data type supports semantic vector search and retrieval applications, such as RAG-based applications, and common operations on vectors in vector-processing applications.

To specify a VECTOR type, use the following syntax:

```
VECTOR( <type>, <dimension> )
```

Where:

- `<type>` is the Snowflake data type of the elements, which can be 32-bit integers or 32-bit floating-point numbers.

  You can specify one of the following types:

  - INT
  - FLOAT
- `<dimension>` is the dimension (length) of the vector. This must be a positive integer value with a maximum value of 4096.

Note

Direct vector comparisons (for example, v1 < v2) are byte-wise lexicographic and, although deterministic, won't produce the results that you might expect from number comparisons. So although you can use VECTOR columns in ORDER BY clauses, for vector comparisons, use the vector similarity functions provided.

The following definitions are examples of valid vector definitions:

- Define a vector of 256 32-bit floating-point values:

```
VECTOR(FLOAT, 256)
```

```
VECTOR(INT, 16)
```

The following definitions are examples of invalid vector definitions:

- A vector definition using an invalid value type:

```
VECTOR(STRING, 256)
```

- A vector definition using an invalid vector size:

```
VECTOR(INT, -1)
```

Vector conversion

This section describes how to convert to and from a VECTOR value. For details about casting, see Data type conversion.

Converting a value to a VECTOR value

VECTOR values can be explicitly cast from the following types:

- Variant containing an ARRAY

Converting a value from a VECTOR value

VECTOR values can be explicitly cast to the following types:

- ARRAY
- Structured ARRAY

Loading and unloading vector data

Directly loading and unloading a VECTOR column isn't supported. For VECTOR columns, you must load and unload data as an ARRAY and then cast it to a VECTOR when you use it. To learn how to load and unload ARRAY data types, see Introduction to Loading Semi-structured Data. A common use case for vectors is to generate a vector embedding.

The following example shows how to unload a table with a VECTOR column to an internal stage named `mystage`:

```
CREATE OR REPLACE TABLE myvectortable (a VECTOR(float, 3), b VECTOR(float, 3));
INSERT INTO myvectortable SELECT [1.1,2.2,3]::VECTOR(FLOAT,3), [1,1,1]::VECTOR(FLOAT,3);
INSERT INTO myvectortable SELECT [1,2.2,3]::VECTOR(FLOAT,3), [4,6,8]::VECTOR(FLOAT,3);

COPY INTO @mystage/unload/
  FROM (SELECT TO_ARRAY(a), TO_ARRAY(b) FROM myvectortable);
```

```
CREATE OR REPLACE TABLE arraytable (a ARRAY, b ARRAY);

COPY INTO arraytable
  FROM @mystage/unload/mydata.csv.gz;

SELECT a::VECTOR(FLOAT, 3), b::VECTOR(FLOAT, 3)
  FROM arraytable;
```

Examples

Construct a VECTOR by casting a constant ARRAY:

```
SELECT [1, 2, 3]::VECTOR(FLOAT, 3) AS vec;
```

Add a column with the VECTOR data type:

```
ALTER TABLE myissues ADD COLUMN issue_vec VECTOR(FLOAT, 768);

UPDATE TABLE myissues
  SET issue_vec = SNOWFLAKE.CORTEX.EMBED_TEXT_768('e5-base-v2', issue_text);
```

The following limitations apply to VECTOR data:

- The VECTOR data type has limited language support. Languages not represented in this table aren't supported.

Snowflake feature	Python	SQL
UDFs	✓	✓
UDTFs	✓	✓
Drivers/Connectors	✓	✓
Snowpark API	✓	

- Vectors aren't supported in VARIANT columns.
- Vectors aren't supported as clustering keys.
- Server-side binding isn't supported. This means that when writing to a VECTOR column through a Snowflake driver, you must cast the VECTOR values in the query before running the query.
- Vectors are allowed in hybrid tables, but not as primary keys or secondary index keys.
- The VECTOR data type isn't supported for use with the following Snowflake features:
  - Snowflake Scripting
  - Apache Iceberg™ tables
  - Search Optimization Service
  - Snowpipe
  - Bind variables