**Working with Semi-structured parquet data**

## **Introduction**

This tutorial describes how you can upload Parquet data by transforming elements of a staged Parquet file directly into table columns using the [COPY INTO <table>](https://docs.snowflake.com/en/sql-reference/sql/copy-into-table) command. The tutorial also describes how you can use the [COPY INTO <location>](https://docs.snowflake.com/en/sql-reference/sql/copy-into-location) command to unload table data into a Parquet file.

### **Downloading the sample data file**

To download the sample Parquet data file, click [cities.parquet](https://docs.snowflake.com/en/_downloads/0c1e6c4f4140561029eeb20afdd02664/cities.parquet). Alternatively, right-click the link and save the link/file to your local file system.

The tutorial assumes you unpacked files in to the following directories:

* Linux/macOS: /tmp/load
* Windows: C:\temp\load

The Parquet data file includes sample continent data. The following is a representative example:

{

"continent": "Europe",

"country": {

"city": [

"Paris",

"Nice",

"Marseilles",

"Cannes"

],

"name": "France"

}

}

### 

### 

### 

### 

### **Creating the database, table, and virtual warehouse**

The following commands create objects specifically for use with this tutorial. When you have completed the tutorial, you can drop these objects.

create or replace database mydatabase;

use schema mydatabase.public;

create or replace temporary table cities (

continent varchar default null,

country varchar default null,

city variant default null

);

create or replace warehouse mywarehouse with

warehouse\_size='X-SMALL'

auto\_suspend = 120

auto\_resume = true

initially\_suspended=true;

use warehouse mywarehouse;

Note these commands create a temporary table. Temporary tables persist only for the duration of the user session and are not visible to other users.

## **Create file format object**

Execute the [CREATE FILE FORMAT](https://docs.snowflake.com/en/sql-reference/sql/create-file-format) command to create the sf\_tut\_parquet\_format file format.

CREATE OR REPLACE FILE FORMAT sf\_tut\_parquet\_format

TYPE = parquet;

TYPE = 'parquet' indicates the source file format type. CSV is the default file format type.

## **Create stage object**

Execute the [CREATE STAGE](https://docs.snowflake.com/en/sql-reference/sql/create-stage) command to create the internal sf\_tut\_stage stage.

**CREATE** **OR** **REPLACE** **TEMPORARY** **STAGE** sf\_tut\_stage

**FILE\_FORMAT** **=** sf\_tut\_parquet\_format**;**

Similar to temporary tables, temporary stages are automatically dropped at the end of the session.

## **Stage the data file**

Execute the [PUT](https://docs.snowflake.com/en/sql-reference/sql/put) command to upload the parquet file from your local file system to the named stage.

Linux or macOS  
**PUT** **file:///**tmp**/load/**cities**.parquet** **@**sf\_tut\_stage**;**

Windows  
**PUT** **file://**C**:\temp\load\**cities**.parquet** **@**sf\_tut\_stage**;**

## **Copy data into the target table**

Copy the cities.parquet staged data file into the CITIES table.

**copy** **into** cities

**from** **(select** **$**1:continent**::varchar,**

**$**1:country:name**::varchar,**

**$**1:country:city**::variant**

**from** **@**sf\_tut\_stage**/**cities**.parquet);**

Note the following:

* $1 in the SELECT query refers to the single column where the Parquet data is stored.
* The query casts each of the Parquet element values it retrieves to specific column types.

Execute the following query to verify data is copied.

**SELECT** **\*** **from** cities**;**

The query returns the following result:

**+**---------------+---------+-----------------+

| CONTINENT | COUNTRY | CITY |

|---------------+---------+-----------------|

| Europe | France | [ |

| | | "Paris", |

| | | "Nice", |

| | | "Marseilles", |

| | | "Cannes" |

| | | ] |

|---------------+---------+-----------------|

| Europe | Greece | [ |

| | | "Athens", |

| | | "Piraeus", |

| | | "Hania", |

| | | "Heraklion", |

| | | "Rethymnon", |

| | | "Fira" |

| | | ] |

|---------------+---------+-----------------|

| North America | Canada | [ |

| | | "Toronto", |

| | | "Vancouver", |

| | | "St. John's", |

| | | "Saint John", |

| | | "Montreal", |

| | | "Halifax", |

| | | "Winnipeg", |

| | | "Calgary", |

| | | "Saskatoon", |

| | | "Ottawa", |

| | | "Yellowknife" |

| | | ] |

**+**---------------+---------+-----------------+

## 

## **Unload the table**

Unload the CITIES table into another Parquet file.

**Note**

By default, Snowflake optimizes table columns in unloaded Parquet data files by setting the smallest precision that accepts all of the values. If you prefer consistent output file schema determined by the “logical” column data types (i.e. the types in the unload SQL query or source table), set the [ENABLE\_UNLOAD\_PHYSICAL\_TYPE\_OPTIMIZATION](https://docs.snowflake.com/en/sql-reference/parameters.html#label-enable-unload-physical-type-optimization) session parameter to FALSE.

**copy** **into** **@**sf\_tut\_stage**/out/**parquet\_

**from** **(select** continent**,**

country**,**

c**.value::string** **as** city

**from** cities**,**

**lateral** flatten**(input** **=>** city**)** c**)**

**file\_format** **=** **(type** **=** 'parquet'**)**

**header** **=** **true;**

**Unloading** in Snowflake is the process of exporting data from a Snowflake table into external file formats, like **Parquet**, **CSV**, or **JSON**, which can then be stored in an external stage (like S3, Azure, etc.) or downloaded for further use.

### **Reasons for Unloading Data:**

1. **Data Export for External Use:**
   * Unloading allows you to export data to external systems or environments for analysis, integration with other tools, or archiving.
   * For example, if you need to move data from Snowflake to a data lake or an on-premise system, exporting it in formats like Parquet is common.
2. **Backup and Archiving:**
   * Unloading data can serve as a way to back up the data or archive it for compliance, audit, or historical purposes.
   * Parquet is an efficient format for storage due to its compression and columnar storage capabilities.
3. **Data Sharing:**
   * Exporting the data allows you to share it with partners, customers, or other departments that may not have direct access to your Snowflake environment.
   * Parquet is a highly portable format used across many big data platforms like Hadoop, Spark, and more.
4. **Performance and Efficiency:**
   * **Parquet** is a columnar file format, which means it’s optimized for performance in analytic workloads. Exporting Snowflake data into **Parquet** enables efficient queries on the data when used in analytics platforms.
   * Parquet is particularly useful when dealing with large datasets, as it supports both compression and faster queries due to its structure.
5. **Interoperability:**
   * If you are working with tools or systems that require specific file formats (e.g., big data platforms like Spark, Hadoop), unloading into a Parquet file provides compatibility and ease of integration.

### **How the Query Works:**

* **LATERAL FLATTEN**: This part of the query un-nests the city field from the CITIES table, which likely contains arrays or nested structures.
* **COPY INTO**: The COPY INTO command is used to export data from Snowflake into an external file or stage.
* **Parquet Format**: The file\_format = (type = 'parquet') specifies that the output will be in **Parquet** format, a popular columnar storage format that is highly optimized for big data.

Note the following:

* The file\_format = (type = 'parquet') specifies parquet as the format of the data file on the stage. When the Parquet file type is specified, the COPY INTO <location> command unloads data to a single column by default.
* The header=true option directs the command to retain the column names in the output file.
* In the nested SELECT query:
  + The [FLATTEN](https://docs.snowflake.com/en/sql-reference/functions/flatten) function first flattens the city column array elements into separate columns.
  + The LATERAL modifier joins the output of the FLATTEN function with information outside of the object - in this example, the continent and country.

Execute the following query to verify data is copied into a staged Parquet file.

**select** **t.$**1 **from** **@**sf\_tut\_stage**/out/** **t;**

The query returns the following results (only partial result is shown):

**+**---------------------------------+

| $1 |

|---------------------------------|

| { |

| "CITY": "Paris", |

| "CONTINENT": "Europe", |

| "COUNTRY": "France" |

| } |

|---------------------------------|

| { |

| "CITY": "Nice", |

| "CONTINENT": "Europe", |

| "COUNTRY": "France" |

| } |

|---------------------------------|

| { |

| "CITY": "Marseilles", |

| "CONTINENT": "Europe", |

| "COUNTRY": "France" |

| } |

**+**---------------------------------+

## **Remove the successfully copied data files**

After you verify that you successfully copied data from your stage into the tables, you can remove data files from the internal stage using the [REMOVE](https://docs.snowflake.com/en/sql-reference/sql/remove) command to save on [data storage](https://docs.snowflake.com/en/user-guide/cost-understanding-compute).

**REMOVE** **@**sf\_tut\_stage**/**cities**.parquet;**

## **Clean up**

Execute the following [DROP <object>](https://docs.snowflake.com/en/sql-reference/sql/drop) commands to return your system to its state before you began the tutorial:

**DROP** **DATABASE** **IF** **EXISTS** mydatabase**;**

**DROP** **WAREHOUSE** **IF** **EXISTS** mywarehouse**;**

Dropping the database automatically removes all child database objects such as tables.

Credit: Snowflake official documentation

Happy Cloud computing

Regards

Saransh Jain

If the above link for downloading the data file doesn't work, then only use this below link.

For the parquet data file:

<https://drive.google.com/file/d/1oIOOto-Q38bw1FCvS94w4vNFojacz9Vn/view?usp=drive_link>