Snowflake Export Integration

Learn more about Snowflake Import Integration.

You can write job results directly to Snowflake. For example, you might integrate data from multiple sources into Treasure Data, run queries to organize the data, and then use a job that applies the connector to export the data to Snowflake.

Prerequisites

- Basic knowledge of Treasure Data, including the TD Toolbelt.
- A Snowflake account.

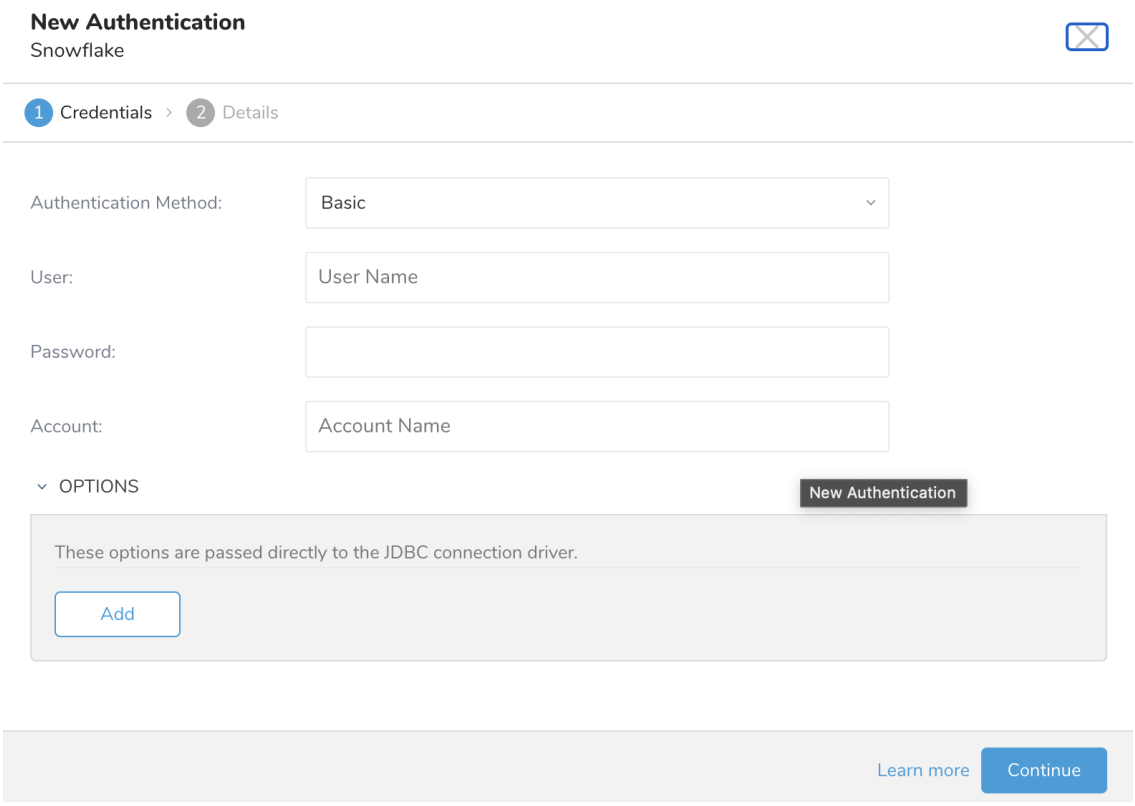
Use the TD Console to create your connection

You can use the Treasure Data Console to configure your connection.

Create a new connection

Go to Treasure Data Connections and search and select Snowflake.

The following dialog opens.

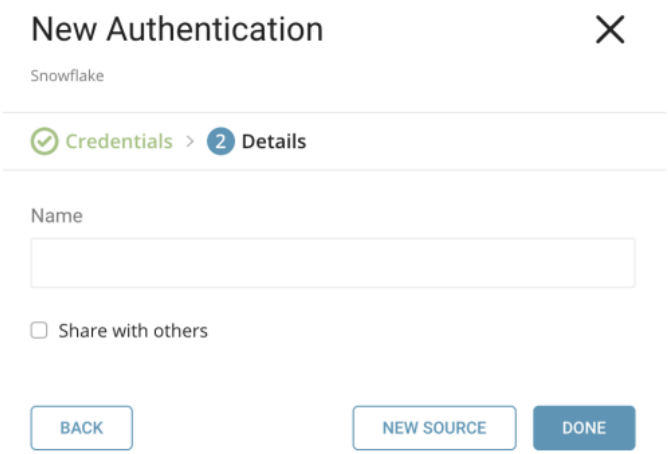


Select Authentication Method:

- **Basic**: Provide the required credentials: **User**, **Password**, and **Account** to authenticate Treasure Data to Snowflake.
    - **User**: Snowflake login username.
    - **Password**: Snowflake login password.
- **Key Pair**: Provide the **Private Key** and its **Passphrase** if it is the encrypted private key
    - **Private Key**: Your generated private key. See configuring-key-pair-authentication
    - **Passphrase**: The Private Key passphrase, or leave it empty if the Private Key is unencrypted.
    - **User**: Snowflake login username.
- **Account**: Snowflake provided the account name. See how to find your account name in Snowflake.
- **OPTIONS**: This option is not supported for this connector.

Provide the required credentials: **User**, **Password**, and Snowflake **Account** to authenticate Treasure Data to Snowflake.

Then select `Continue` and give your connection a name:



Select **Done**.

Specify Output in Your Snowflake Connector

Create or reuse a query that configures the data connection.

Sometimes you need to define the column mapping in the query.

Configure the Connection by Specifying the Parameters

Go to the Treasure Data console. Go to the Query Editor. Access the query that you plan to use to export data.

Sometimes you need to define the column mapping in the query.

For Example:

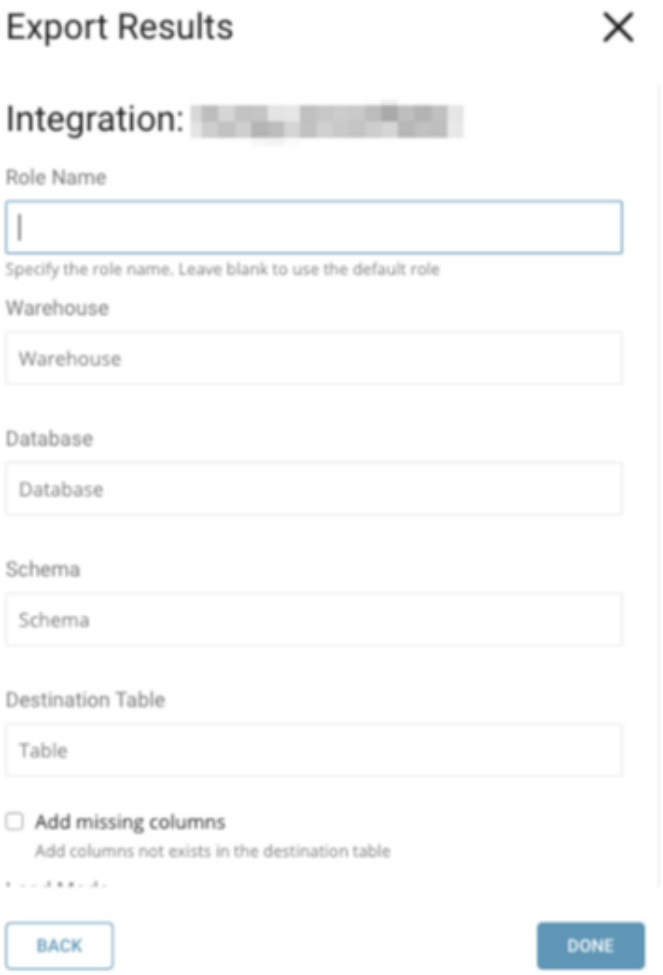
```
SELECT c_id, c_double, c_long, c_string, c_boolean, c_timestamp, c_json FROM (
 VALUES (1, 100, 10, 'T4', true, '2018-01-01','{ "name":"John"}'),
 (2, 100, 99, 'P@#4', false, '2018-01-01', '{ "name":"John"}'),
 (3, 100.1234, 22, 'C!%^&*', false, '2018-01-01','{ "name":"John"}')
) tbl1 (c_id, c_double, c_long, c_string, c_boolean, c_timestamp, c_json)
```

Select **Output Results** located at top of your query editor. The Choose Saved Connection dialog opens.

Configure the connection by specifying the parameters

Type the connection name in the search box to filter and select your connection.

After you select your *Snowflake* connection, the *Configuration* or Export Results dialog pane appears:



Specify parameters. The parameters are as follows:

- **Role Name (optional):** The default access control role to use for exporting, it should be an existing role that has already been assigned to the user.
- **Warehouse (required):** The virtual warehouse to use, it should be an existing warehouse for which the specified role has privileges.
- **Database (required):** The database to use, it should be an existing database for which the specified role has privileges.
- **Schema (required):** The schema to use for the specified database, it should be an existing schema for which the specified role has privileges.
- **Destination table (required):** The table which results will be exported to. If it does not exist, the new one will be created.
- **Add missing columns:** If columns from the target table are less than columns in TD results, an exception will be thrown if the option is unselected. If it is selected, new columns which are missing will be added to the target table.
- **Load mode (required):** There are 3 mode to fulfill results to Snowflake table, **Append**, **Overwrite**, **Merge**. If **Merge** is selected, Merge Field(s) will be shown.
    - **Append:** TD results will be appended to the target table
    - **Overwrite:** The target table will be erased, then TD results will be appended from the beginning of target table
    - **Merge:** The target table will be merged with the TD results based on conditions from **Merge Fields**.
- **Merge Fields (required if in merge mode):** The fields will be used to compare between target table and TD results, if those fields values are equal, a record in target table will be overridden otherwise new record will be inserted into the target table. The format for merge fields is a comma-separated text field.
- **Data type mapping:** It will be explained in the following section.

Data type mapping

Here is the table for 1-1 mapping from the type in TD results to the target table in case target table does not exist or new columns are added:

TD results	Snowflake

string	STRING
double	FLOAT
long	BIGINT
timestamp	TIMESTAMP
boolean	BOOLEAN

You might want to set the different type as the default. Data Type Mapping is used to explicitly set a specific type (for example, VARCHAR) to a specific column. Data Type Mapping applies a type only to columns in your target table.

If you use Data Mapping, the following is true:

- If the target table does not exist, then the export job will create a new target table.
- If the target table does not have enough columns as compared to TD results, you can specify to add more columns.

The syntax for data type mapping parameter is: **col_name_1: VARCHAR; col_name2: BIGINT**, You must provide: column name and Snowflake data type.

JSON is not fully supported when you export data using a Treasure Data Presto or Hive query. Therefore when a target table does not exist and is created, a JSON data type value is saved as VARCHAR by default in the newly created target table. If you want to save a JSON value as a semi-structured type instead, then you must update the type mapping. Use the Data Mapping syntax to specify a semi-structured data type.

For example, in the query:

```
SELECT c_id, c_double, c_long, c_string, c_boolean, c_timestamp, c_json FROM (
 VALUES (1, 100, 10, 'T4', true, '2018-01-01','{ "name":"John"}'),
 (2, 100, 99, 'P@#4', false, '2018-01-01', '{ "name":"John"}'),
 (3, 100.1234, 22, 'C!%^&*', false, '2018-01-01','{ "name":"John"}')
) tbl1 (c_id, c_double, c_long, c_string, c_boolean, c_timestamp, c_json)
```

The **c_json** column has JSON content, but the type of that **c_json** column in Snowflake would be VARCHAR by default. If you want a VARIANT type in Snowflake, update the **Data type mapping** field to **c_json: VARIANT** to explicitly set **c_json** to **VARIANT** type.

Optional: Use of Scheduled Jobs for Output

You can use Scheduled Jobs with Result Output, to periodically write the output result to a target destination that you specify.

Optional: Configure Export Results in Workflow

Within Treasure Workflow, you can specify the use of this data connector to output data.

```
timezone: UTC

_export:
  td:
    database: sample_datasets

+td-result-into-snowflake:
  td>: queries/sample.sql
  result_connection: your_connection_name
  result_settings:
    roleName: role
    warehouse: wh
    schema: public
    database: OUTPUT
    table: TARGET_TABLE
    mode: insert
    is_add_missing_columns: false
```

Learn about using data connectors in the workflow to export data.

Use the CLI to create your connection

Install ‘td’ command

Install the TD Toolbelt.

For On-demand Jobs

Add the Snowflake result output destination by using the `-r` / `--result` option for the `td query` command:

```
$ td query -d mydb -w 'SELECT id, name FROM source_table' --type presto -r '{"type":"snowflake", "warehouse":"
wh", "user":"owner", "account_name":"owner", "password":"********", "roleName":"role", "schema":"public",
"database":"OUTPUT", "table":"TARGET_TABLE", "mode":"insert", "is_add_missing_columns":"false"}'
```

Some parameters, such as **warehouse**, **database,** are self-explanatory and are the same parameters that are used in the in TD Console (described in the "Configure the connection by specifying the parameters" section). However, some parameters are either different in the key or value:

- **mode (required):** The value for mode is a raw value of **Load Mode** parameter:
    - **insert:** TD results will be appended to target table
    - **truncate_insert:** The target table will be erased, then TD results will be appended from the beginning of target table.
    - **merge:** The target table will be merged with the TD results based on conditions from.
- **merge_keys:** Your fields to compare in merge mode, same as **Merge Field(s)**.
- **column_options:** The data type mapping.

For Scheduled Jobs

Add the Snowflake result output destination by using the `-r` / `--result` option for the `td sched:create` command:

```
$ td sched:create every_6_mins "*/6 * * * *" -d mydb -w 'SELECT id, name FROM source_table' --type presto -r
'{"type":"snowflake", "warehouse":"wh", "user":"owner", "account_name":"owner", "password":"********",
"roleName":"role", "schema":"public", "database":"OUTPUT", "table":"TARGET_TABLE", "mode":"insert",
"is_add_missing_columns":"false"}'
```

Data type mapping

Here is the table for 1-1 mapping from the type in TD results to the target table in case target table does not exist or new columns are added:

TD results	Snowflake
string	STRING
double	FLOAT
long	BIGINT
timestamp	TIMESTAMP
boolean	BOOLEAN

You might want to set the different type as the default. Data Type Mapping is used to explicitly set a specific type (for example, VARCHAR) to a specific column. Data Type Mapping applies a type only to columns in your target table.

If you use Data Mapping, the following is true:

- If the target table does not exist, then the export job will create a new target table.
- If the target table does not have enough columns as compared to TD results, you can specify to add more columns.

The syntax for data type mapping parameter is: **col_name_1: VARCHAR; col_name2: BIGINT**, You must provide: column name and Snowflake data type.

> JSON is not fully supported when you export data using a Treasure Data Presto or Hive query. Therefore when a target table does not exist and is created, a JSON data type value is saved as VARCHAR by default in the newly created target table. If you want to save a JSON value as a semi-structured type instead, then you must update the type mapping. Use the Data Mapping syntax to specify a semi-structured data type.

For example, in the query:

```
SELECT c_id, c_double, c_long, c_string, c_boolean, c_timestamp, c_json FROM (
 VALUES (1, 100, 10, 'T4', true, '2018-01-01','{ "name":"John"}'),
 (2, 100, 99, 'P@#4', false, '2018-01-01', '{ "name":"John"}'),
 (3, 100.1234, 22, 'C!%^&*', false, '2018-01-01','{ "name":"John"}')
) tbl1 (c_id, c_double, c_long, c_string, c_boolean, c_timestamp, c_json)
```

The **c_json** column has JSON content, but the type of that **c_json** column in Snowflake would be VARCHAR by default. If you want a VARIANT type in Snowflake, update the **Data type mapping** field to **c_json: VARIANT** to explicitly set **c_json** to **VARIANT** type.

Optional: Use of Scheduled Jobs for Output

You can use Scheduled Jobs with Result Output, to periodically write the output result to a target destination that you specify.

Optional: Configure Export Results in Workflow

Within Treasure Workflow, you can specify the use of this data connector to output data.

```
timezone: UTC

_export:
  td:
    database: sample_datasets

+td-result-into-snowflake:
  td>: queries/sample.sql
  result_connection: your_connection_name
  result_settings:
    roleName: role
    warehouse: wh
    schema: public
    database: OUTPUT
    table: TARGET_TABLE
    mode: insert
    is_add_missing_columns: false
```

Learn about using data connectors in the workflow to export data.

Use the CLI to create your connection

Install ‘td’ command

Install the TD Toolbelt.

For On-demand Jobs

Add the Snowflake result output destination by using the `-r` / `--result` option for the `td query` command:

```
$ td query -d mydb -w 'SELECT id, name FROM source_table' --type presto -r '{"type":"snowflake", "warehouse":"
wh", "user":"owner", "account_name":"owner", "password":"********", "roleName":"role", "schema":"public",
"database":"OUTPUT", "table":"TARGET_TABLE", "mode":"insert", "is_add_missing_columns":"false"}'
```

Some parameters, such as **warehouse**, **database,** are self-explanatory and are the same parameters that are used in the in TD Console (described in the "Configure the connection by specifying the parameters" section). However, some parameters are either different in the key or value:

- **mode (required):** The value for mode is a raw value of **Load Mode** parameter:
    - **insert:** TD results will be appended to target table
    - **truncate_insert:** The target table will be erased, then TD results will be appended from the beginning of target table.
    - **merge:** The target table will be merged with the TD results based on conditions from.
- **merge_keys:** Your fields to compare in merge mode, same as **Merge Field(s)**.
- **column_options:** The data type mapping.

For Scheduled Jobs

Add the Snowflake result output destination by using the `-r` / `--result` option for the `td sched:create` command:

```
$ td sched:create every_6_mins "*/6 * * * *" -d mydb -w 'SELECT id, name FROM source_table' --type presto -r
'{"type":"snowflake", "warehouse":"wh", "user":"owner", "account_name":"owner", "password":"********",
"roleName":"role", "schema":"public", "database":"OUTPUT", "table":"TARGET_TABLE", "mode":"insert",
"is_add_missing_columns":"false"}'
```

Appendix

Support for SSL

Connection to Snowflake server is made via their official JDBC driver. The JDBC driver forces the usage of SSL as default and mandatory (i.e. connection with SSL = false will be rejected).