

Lab 2: Getting Started with SnowSQL

Overview

SnowSQL is the software CLI tool used to interact with Snowflake. Using SnowSQL, you can control all aspects of your Snowflake Data Cloud, including uploading data, querying data, changing data, and deleting data. This guide will review SnowSQL and use it to create a database, load data, and learn helpful commands to manage your tables and data directly from your CLI.

What You'll Learn

- how to create a Snowflake account
- how to install SnowSQL locally
- how to set up a cloud database and table
- how to create a virtual warehouse
- how to migrate sample data to the cloud
- how to query cloud data
- how to insert additional data
- how to drop database objects and close SnowSQL connection

Be sure to check the needed computing requirements before beginning. Also, download the sample files to complete this tutorial and note the folder location for later use.

What You'll Need

- Local [Browser and OS Requirements]
- Download the [Sample Data Files]

What You'll Build

- A connection to cloud host and manage data with SnowSQL.

Set up SnowSQL

First, you'll get a Snowflake account and get comfortable navigating in the web console. After downloading the SnowSQL installer, you'll install and confirm your success.

Create a Snowflake Account

Snowflake lets you try out their services for free with a [trial account](#).


Access Snowflake's Web Console

```
https://<account-name>.snowflakecomputing.com/console/login
```

Log in to the web interface on your browser. The URL contains your [account name] and potentially the region.

Increase Your Account Permission

The Snowflake web interface has a lot to offer, but for now, we'll just switch the account role from the default `sysadmin` to `ACCOUNTADMIN`. This increase in permissions will allow the user account to create objects.

 account-role-change-image

Download the SnowSQL Installer

SnowSQL can be downloaded and installed on Linux, Windows, or Mac. Download the SnowSQL Installer for Windows from here:

```
https://sfc-repo.snowflakecomputing.com/snowsql/bootstrap/1.2/windows_x86_64/snowsql-1.2.21-windows_x86_64.msi
```

Install SnowSQL Locally

- Double-click the installer file and walk through the wizard prompts.
- Confirm the install was successful by checking the version `$ snowsql -v`.
- Reboot your machine and check again if necessary.

After completing these steps, you'll be ready to use SnowSQL to make a database in the next section.

Create a Database

With your account active and SnowSQL installed, you'll use the terminal to sign in and create the needed objects for cloud storage.

Sign in From the Terminal

```
snowsql -a <account-name> -u <username>
```

The `-a` flag represents the Snowflake account, and the `-u` represents the username.

Create a Database and Schema

```
create or replace database sf_tuts;
```

The command [create or replace database] makes a new database and auto-creates the schema 'public.' It'll also make the new database active for your current session.

To check which database is in use for your current session, execute:

```
select current_database(),  
current_schema();
```

Generate a Table

```
create or replace table emp_basic (  
  first_name string ,  
  last_name string ,  
  email string ,  
  streetaddress string ,  
  city string ,  
  start_date date  
);
```

Running [create or replace table] will build a new table based on the parameters specified. This example reflects the same columns in the sample CSV employee data files.

```
snowsql • snowsql -a -u - 94x29
#COMPUTE_WH@SF_TUTS.PUBLIC>create or replace table emp_basic (
    first_name string ,
    last_name string ,
    email string ,
    streetaddress string ,
    city string ,
    start_date date
);
+-----+
| status |
+-----+
| Table EMP_BASIC successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.331s
#COMPUTE_WH@SF_TUTS.PUBLIC>
```

Make a Virtual Warehouse

```
create or replace warehouse sf_tuts_wh with
warehouse_size='X-SMALL'
auto_suspend = 180
auto_resume = true
initially_suspended=true;
```

After creation, this virtual warehouse will be active for your current session and begin running once the computing resources are needed.

```
snowsql • snowsql -a -u - 94x29
#COMPUTE_WH@SF_TUTS.PUBLIC>create or replace warehouse sf_tuts_wh with
    warehouse_size='X-SMALL'
    auto_suspend = 180
    auto_resume = true
    initially_suspended=true;
+-----+
| status |
+-----+
| Warehouse SF_TUTS_WH successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 3.662s
#SF_TUTS_WH@SF_TUTS.PUBLIC>
```

With the database objects ready, you'll employ SnowSQL to move the sample data onto the `emp_basic` table.

Upload Data

In this section, you'll stage your sample CVS employee files and execute a SQL command to copy the data onto your table.

If you have not already done so, you can download the sample files here:

[Download Sample Data](#)

Stage Files With PUT

Linux

```
put file:///tmp/employees0*.csv @<database-name>.<schema-name>.%<table-name>;
```

Windows

```
put file://c:\temp\employees0*.csv @sf_tuts.public.%emp_basic;
```

- `file` specifies the *local* file path of the files to be staged. File paths are OS-specific.
- `@<database-name>.<schema-name>.%<table-name>` is the specific database, schema, and table the staged files are headed.
- The `@` sign before the database and schema name `@sf_tuts.public` indicates that the files are being uploaded to an internal stage, rather than an external stage. The `%` sign before the table name `%emp_basic` indicates that the internal stage being used is the stage for the table. For more details about stages, see Staging Data Files from a Local File System.

```
put file:///tmp/employees0*.csv @sf_tuts.public.%emp_basic;
```

Here is a PUT call to stage the sample employee CSV files from a macOS `file:///tmp/` folder onto the `emp_basic` table within the `sf_tuts` database.

```
snowsql -a -u - 80x37
#SF_TUTS_WH@SF_TUTS.PUBLIC>put file:///tmp/employees0*.csv @sf_tuts.publ
ic.%emp_basic;
employees03.csv_c.gz(0.00MB): [#####] 100.00% Done (1.724s, 0.00MB/s).
employees01.csv_c.gz(0.00MB): [#####] 100.00% Done (1.203s, 0.00MB/s).
employees02.csv_c.gz(0.00MB): [#####] 100.00% Done (1.224s, 0.00MB/s).
employees04.csv_c.gz(0.00MB): [#####] 100.00% Done (0.790s, 0.00MB/s).
employees05.csv_c.gz(0.00MB): [#####] 100.00% Done (0.566s, 0.00MB/s).
+-----+
| source | target | source_size | target_size | source_comp |
| session | target_compression | status | message |
+-----+
| employees01.csv | employees01.csv.gz | 370 | 288 | NONE |
| | GZIP | UPLOADED | |
| employees02.csv | employees02.csv.gz | 364 | 276 | NONE |
| | GZIP | UPLOADED | |
| employees03.csv | employees03.csv.gz | 407 | 298 | NONE |
| | GZIP | UPLOADED | |
| employees04.csv | employees04.csv.gz | 375 | 290 | NONE |
| | GZIP | UPLOADED | |
| employees05.csv | employees05.csv.gz | 404 | 303 | NONE |
| | GZIP | UPLOADED | |
+-----+
5 Row(s) produced. Time Elapsed: 5.661s
#SF_TUTS_WH@SF_TUTS.PUBLIC>
```

LIST Staged Files

```
list @<database-name>.<schema-name>.<table-name>;
```

To check your staged files, run the `list` command.

```
list @sf_tuts.public.%emp_basic;
```

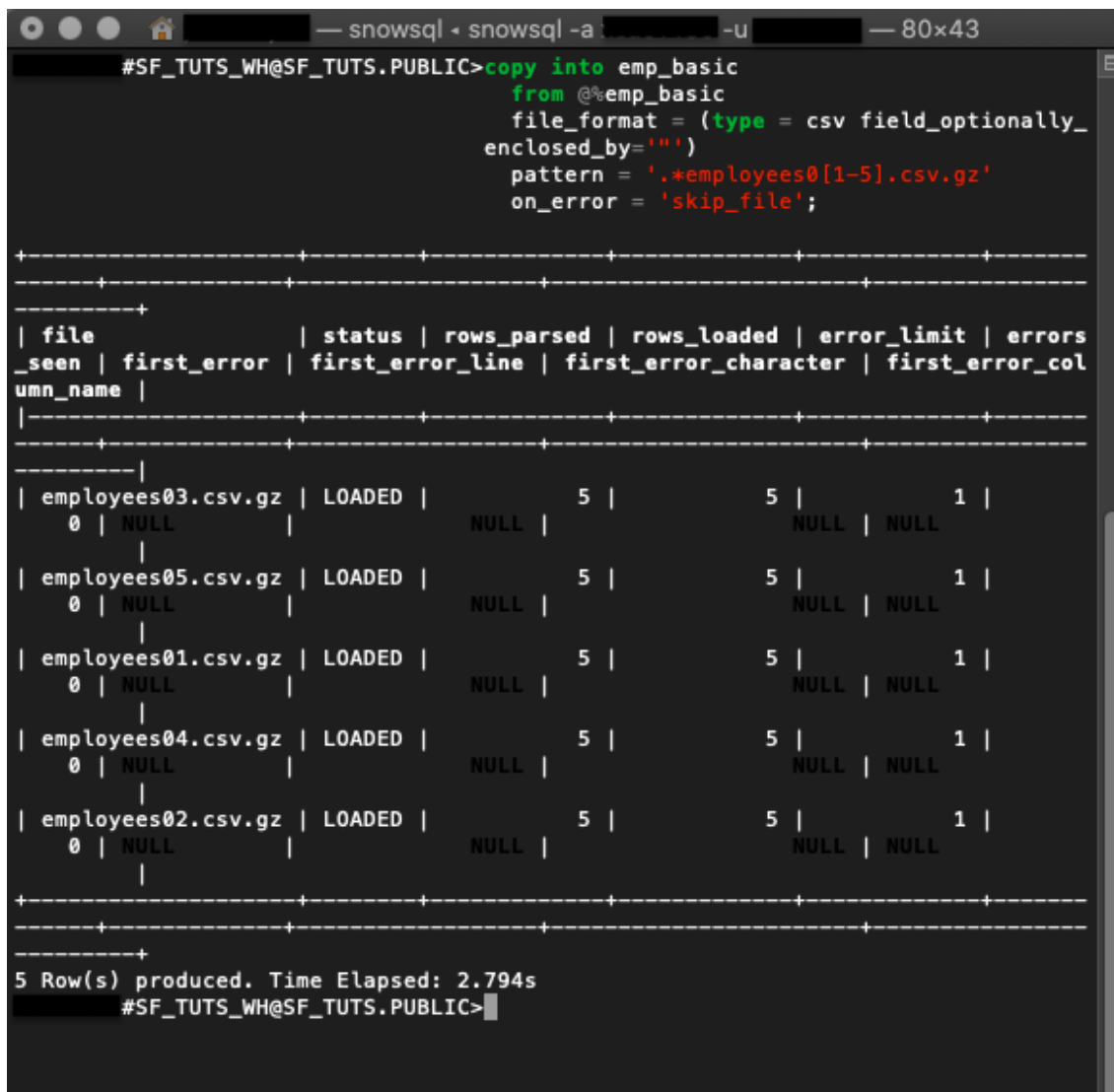
The example command above is to output the staged files for the `emp_basic` table.

```
snowsql -a -u - 100x50
#SF_TUTS_WH@SF_TUTS.PUBLIC>list @sf_tuts.public.%emp_basic;
+-----+
| name | size | md5 | last_modified |
+-----+
| employees01.csv.gz | 304 | 63451cc8f4db701b3918d2475aa360a2 | GMT |
| employees02.csv.gz | 288 | fc1e7fced992fd4f2d3da3464c7be0e4 | GMT |
| employees03.csv.gz | 304 | 0f687fe91584c057a461163999a3baf3 | GMT |
| employees04.csv.gz | 304 | adc308d8224f2d432cb57c91f6b36e5e | GMT |
| employees05.csv.gz | 304 | d4c1882c6eafcdf3d70b9f6df18ac4b0 | GMT |
+-----+
5 Row(s) produced. Time Elapsed: 0.797s
#SF_TUTS_WH@SF_TUTS.PUBLIC>
```

[COPY INTO] Your Table

```
copy into emp_basic
  from @%emp_basic
  file_format = (type = csv field_optionally_enclosed_by='')
  pattern = '*.employees0[1-5].csv.gz'
  on_error = 'skip_file';
```

After getting the files staged, the data is copied into the `emp_basic` table. This DML command also auto-resumes the virtual warehouse made earlier.



```
#SF_TUTS_WH@SF_TUTS.PUBLIC>copy into emp_basic
                             from @%emp_basic
                             file_format = (type = csv field_optionally_
                             enclosed_by='')
                             pattern = '*.employees0[1-5].csv.gz'
                             on_error = 'skip_file';
```

file	status	rows_parsed	rows_loaded	error_limit	errors
_seen	first_error	first_error_line	first_error_character	first_error_col	umn_name
employees03.csv.gz	LOADED	5	5	1	
0	NULL	NULL	NULL	NULL	NULL
employees05.csv.gz	LOADED	5	5	1	
0	NULL	NULL	NULL	NULL	NULL
employees01.csv.gz	LOADED	5	5	1	
0	NULL	NULL	NULL	NULL	NULL
employees04.csv.gz	LOADED	5	5	1	
0	NULL	NULL	NULL	NULL	NULL
employees02.csv.gz	LOADED	5	5	1	
0	NULL	NULL	NULL	NULL	NULL

```
5 Row(s) produced. Time Elapsed: 2.794s
#SF_TUTS_WH@SF_TUTS.PUBLIC>
```

The output indicates if the data was successfully copied and records any errors.

Query Data

With your data in the cloud, you need to know how to query it. We'll go over a few calls that will put your data on speed-dial.

- The `select` command followed by the wildcard `*` returns all rows and columns in `<table-name>`.

```
select * from emp_basic;
```

Here is an example command to `select` everything on the `emp_basic` table.

FIRST_NAME	LAST_NAME	EMAIL	STREETADDRESS	CITY	START_DATE
Althea	Featherstone	afeatherstona@sf_tuts.com	8172 Browning Street, Apt B	Calatrava	2017-07-12
Hollis	Anneslie	hanneslieb@sf_tuts.com	3248 Roth Park	Aleysk	2017-11-16
Betti	Cicco	bciccoc@sf_tuts.com	121 Victoria Junction	Sinegor'ye	2017-06-22
Brendon	Durnall	bdurnalld@sf_tuts.com	26814 Weeping Birch Place	Sabadell	2017-11-14
Kylila	MacConnal	kmacconnale@sf_tuts.com	04 Valley Edge Court	Qingshu	2017-06-22
Arlene	Davidovits	adavidovitsk@sf_tuts.com	7571 New Castle Circle	Meniko	2017-05-03
Violette	Shermore	vshermorel@sf_tuts.com	899 Merchant Center	Troitsk	2017-01-19
Ron	Mattys	rmattysm@sf_tuts.com	423 Lien Pass	Bayaguana	2017-11-15
Shurlocke	Oluwatoyin	soluwatoyinn@sf_tuts.com	40637 Portage Avenue	Semënovskoye	2017-09-12
Granger	Bassford	gbassfordo@sf_tuts.co.uk	6 American Ash Circle	Karditsa	2016-12-30
Lem	Boissier	lboissier@sf_tuts.com	3002 Ruskin Trail	Shikārpur	2017-08-25
Iain	Hanks	ihanks1@sf_tuts.com	2 Pankratz Hill	Monte-Carlo	2017-12-10
Avo	Laudham	alaudham2@sf_tuts.com	6948 Debs Park	Pražmów	2017-10-18
Emili	Cornner	ecornner3@sf_tuts.com	177 Magdeline Avenue	Norrrköping	2017-08-13
Harrietta	Goolding	hgoolding4@sf_tuts.com	450 Heath Trail	Osielsko	2017-11-27
Wallis	Sizey	wsizeyf@sf_tuts.com	36761 American Lane	Taibao	2016-12-30
Di	McGowran	dmcgowrang@sf_tuts.com	1856 Maple Lane	Banjar Bengkelgede	2017-04-22
Carson	Bedder	cbedderh@sf_tuts.co.au	71 Clyde Gallagher Place	Leninskoye	2017-03-29
Dana	Avory	davoryi@sf_tuts.com	2 Holy Cross Pass	Wenlin	2017-05-11
Ronny	Talmdage	rtalmdagej@sf_tuts.co.uk	588 Chinook Street	Yawata	2017-06-02
Nyssa	Dorgan	ndorgan5@sf_tuts.com	7 Tomscot Way	Pampas Chico	2017-04-13
Catherin	Devereu	cdevereu6@sf_tuts.co.au	535 Basil Terrace	Magapit	2016-12-17
Grazia	Glaserman	gglaserman7@sf_tuts.com	162 Debra Lane	Shiquanhe	2017-06-06
Ivett	Casemore	icasemore8@sf_tuts.com	84 Holmberg Pass	Campina Grande	2017-03-29
Cesar	Hovie	chovie9@sf_tuts.com	5 7th Pass	Miami	2016-12-21

Sifting through everything on your table may not be the best use of your time. Getting specific results are simple, with a few functions and some query syntax.

- [WHERE] is an additional clause you can add to your select query.

```
select * from emp_basic where first_name = 'Ron';
```

This query returns a list of employees by the `first_name` of 'Ron' from the `emp_basic` table.

FIRST_NAME	LAST_NAME	EMAIL	STREETADDRESS	CITY	START_DATE
Ron	Mattys	rmattysm@sf_tuts.com	423 Lien Pass	Bayaguana	2017-11-15

- [LIKE] function supports wildcard `%` and `_`.

```
select email from emp_basic where email like '%.au';
```

The like function checks all emails in the `emp_basic` table for `au` and returns a record.



```
#SF_TUTS_WH@SF_TUTS.PUBLIC>select email from emp_basic where email like '%.au';
```

EMAIL
cbedderh@sf_tuts.co.au
cdevereu6@sf_tuts.co.au

```
2 Row(s) produced. Time Elapsed: 1.275s
#SF_TUTS_WH@SF_TUTS.PUBLIC>
```

Snowflake supports many [functions], [operators], and [commands]. However, if you need more specific tasks performed, consider setting up an [external function].

Manage and Delete Data

Often data isn't static. We'll review a few common ways to maintain your cloud database.

If HR updated the CSV file after hiring another employee, downloading, staging, and copying the whole CSV would be tedious. Instead, simply insert the new employee information into the targeted table.

Insert Data

[INSERT] will update a table with additional values.

```
insert into emp_basic values
  ('Clementine','Adamou','cadamou@sf_tuts.com','10510 Sachs Road','Klenak','2017-9-22') ,
  ('Marlowe','De Anesy','madamouc@sf_tuts.co.uk','36768 Northfield Plaza','Fangshan','2017-1-26');
```

Drop Objects

In the command displayed, `insert` is used to add two new employees to the `emp_basic` table.


```
snowsql -a -u 90x25
#SF_TUTS_WH@SF_TUTS.PUBLIC>insert into emp_basic values
('Clementine','Adamou','cadamou@sf_tuts.com','10510 S
achs Road','Klenak','2017-9-22'),
('Marlowe','De Anesy','madamouc@sf_tuts.co.uk','36768
Northfield Plaza','Fangshan','2017-1-26');

+-----+
| number of rows inserted |
+-----+
| 2 |
+-----+

2 Row(s) produced. Time Elapsed: 2.011s
#SF_TUTS_WH@SF_TUTS.PUBLIC>
```

- [DROP] objects no longer in use.

```
drop database if exists sf_tuts;

drop warehouse if exists sf_tuts_wh;
```

After practicing the basics covered in this tutorial, you'll no longer need the `sf-tuts` database and warehouse. To remove them altogether, use the `drop` command.

- Close Your Connection with `!exit` or `!disconnect`

For security reasons, it's best not to leave your terminal connection open unnecessarily. Once you're ready to close your SnowSQL connection, simply enter `!exit`.

Conclusion

Use SnowSQL for Your Application

You've created a Snowflake account, set up a cloud database with compute resources, and migrated data to the cloud with SnowSQL. Nice work! There are many advantages to using the cloud. Now that you know how easy getting started with Snowflake is, it's time to consider your next steps.

With your firm grasp of loading data with SnowSQL, start using it to run your application. Continue by [developing an application] with SnowSQL to learn how to connect your data to a Python application. If you already have application data, consider migrating it to the cloud with the same steps we used to complete the `emp_basic` table. Snowflake's tools and documentation are extensive and give you the power of cloud computing.

What we've covered

- SnowSQL setup
- Uploading data using SnowSQL
- Querying data using SnowSQL
- Managing and deleting data using SnowSQL