

## Experiment 3: Data Prep

### Overview

Let's start by preparing to load the structured data on Citi Bike rider transactions into Snowflake.

This experiment will walk you through the steps to:

- Create a database and table
- Create an external stage
- Create a file format for the data

#### Getting Data into Snowflake

There are many ways to get data into Snowflake from many locations including the COPY command, Snowpipe auto-ingestion, an external connector, or a third-party ETL/ELT product. More information on getting data into Snowflake, see <https://docs.snowflake.net/manuals/user-guide-data-load.html>



We are using the COPY command and S3 storage for this experiment in a manual process so you can see and learn from the steps involved. In the real-world, a customer would likely use an automated process or ETL product to make the data loading process fully automated and much easier.

The data we will be using is bike share data provided by [Citi Bike NYC](#). The data has been exported and pre-staged for you in an Amazon AWS S3 bucket in the US-EAST region. The data consists of information about trip times, locations, user type, gender, age of riders, etc. On AWS S3, the data represents 61.5M rows, 377 objects, and 1.9GB total size compressed.

Below is a snippet from one of the Citi Bike CSV data files:

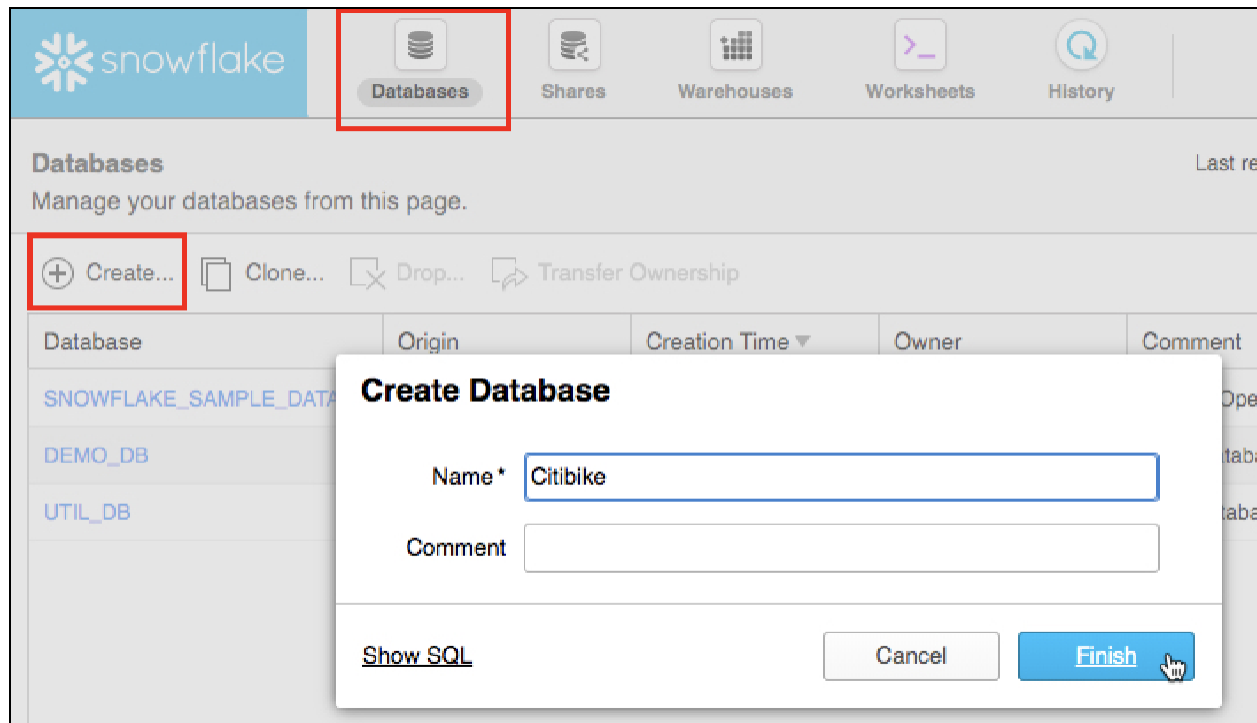
```
"tripduration","starttime","stoptime","start station id","start station name","start
station latitude","start station longitude","end station id","end station name","end
station latitude","end station
longitude","bikeid","name_localizedValue0","usertype","birth year","gender"
196,"2018-01-01 00:01:51","2018-01-01 00:05:07",315,"South St & Gouverneur Ln",
40.70355377,-74.00670227,259,"South St & Whitehall St",
40.70122128,-74.01234218,18534,"Annual Membership","Subscriber",1997,1
207,"2018-01-01 00:02:44","2018-01-01 00:06:11",3224,"W 13 St & Hudson St",
40.73997354103409,-74.00513872504234,470,"W 20 St & 8 Ave",
40.74345335,-74.00004031,19651,"Annual Membership","Subscriber",1978,1
613,"2018-01-01 00:03:15","2018-01-01 00:13:28",386,"Centre St & Worth St",
40.71494807,-74.00234482,2008,"Little West St & 1 Pl",
40.70569254,-74.01677685,21678,"Annual Membership","Subscriber",1982,1
```

It is in comma-delimited format with double quote enclosing and a single header line. This will come into play later in this experiment as we configure the Snowflake table which will store this data.

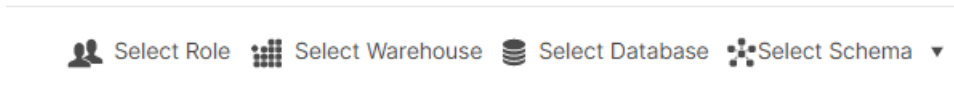
### 3.1 Create a Database and Table

3.1.1 First, let's create a database called CITIBIKE that will be used for loading the structured data.

At the top of the UI select the "Databases" tab. Then click on "Create" and name the database "Citibike" and click "Finish".



- 3.1.2 At the top of the Snowflake UI, click the Worksheets tab. You should see the worksheet with all the SQL we loaded in a prior step. Check your Context and ensure it is set correctly. If it shows the Select options, click on them and set as noted below to SYSADMIN, COMPUTE\_WH, CITIBIKE as noted. You can click on any of the Select areas and enter the details for the Role/Warehouse/Database/Schema.

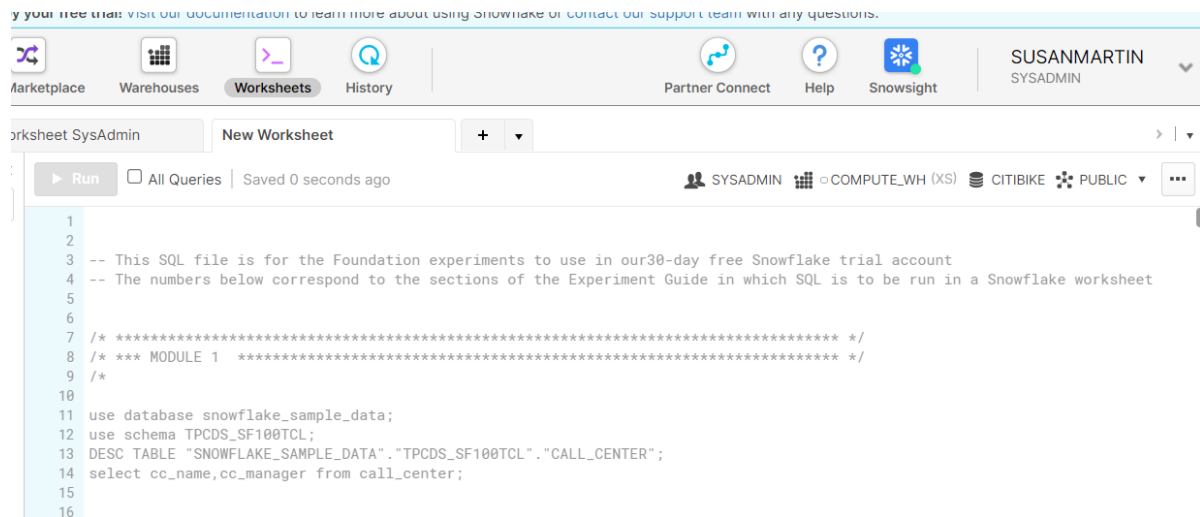
The screenshot shows the Snowflake UI. At the top, there's a navigation bar with icons for Databases, Shares, Warehouses, and Worksheets (which is highlighted with a red box). Below this, there's a 'Worksheet 1' tab. The main area shows a SQL query in a text editor. The query is a CREATE TABLE statement for a table named 'trips'. The context bar at the top right shows 'Context: SYSADMIN, COMPUTE\_WH (XL), CITIBIKE'.

```
1
2
3 -- This SQL file is for the Hands On Lab Guide for the 30-day free Snowflake trial account
4 -- The numbers below correspond to the sections of the Lab Guide in which SQL is to be run in a Snowflake worksheet
5 -- Modules 1 and 2 of the Lab Guide have no SQL to be run
6
7
8 /* ***** */
9 /* *** MODULE 3 ***** */
10 /* ***** */
11
12 -- 3.1.4
13
14 create or replace table trips
15 (tripduration integer,
16  starttime timestamp,
17  stoptime timestamp,
18  start_station_id integer,
19  start_station_name string,
20  start_station_latitude float,
21  start_station_longitude float,
22  end_station_id integer,
23  end_station_name string,
24  end_station_latitude float,
```

3.1.3 First, we need to set the context appropriately within the Worksheet. In the top right, click on the drop-down arrow next to the “Context” section to show the worksheet context menu. Here we control what elements the user can see and run from each worksheet. We are using the UI here to set the context. Later in the experiment we will set the worksheet context via SQL commands in the worksheet.

As needed use the downward arrows to select and show the showing:

Role: SYSADMIN  
Warehouse: COMPUTE\_WH (XL)  
Database: CITIBIKE  
Schema = PUBLIC



3.1.4 Let's now create a table called TRIPS that will be used for loading the comma-delimited data. We will be using the Worksheets tab in the Snowflake UI to run the DDL (data



#### DDL operations are free!

Note that all the DDL operations we have done so far do NOT require compute resources, so we can create all our objects for free.

definition language) to create the table. Based on a prior step, the SQL text below should be showing on the worksheet.

create or replace table trips  
(tripduration integer,  
starttime timestamp,  
stoptime timestamp,  
start\_station\_id integer,  
start\_station\_name string,  
start\_station\_latitude float,

```
start_station_longitude float,  
end_station_id integer,  
end_station_name string,  
end_station_latitude float,  
end_station_longitude float,  
bikeid integer,  
membership_type string,  
usertype string,  
birth_year integer,  
gender integer);
```



### Many Options to Run Commands.

SQL commands can be executed through the UI (limited), via the Worksheets tab, using our SnowSQL command line tool, a SQL editor of your choice via ODBC/JDBC, or through our Python or Spark connectors.

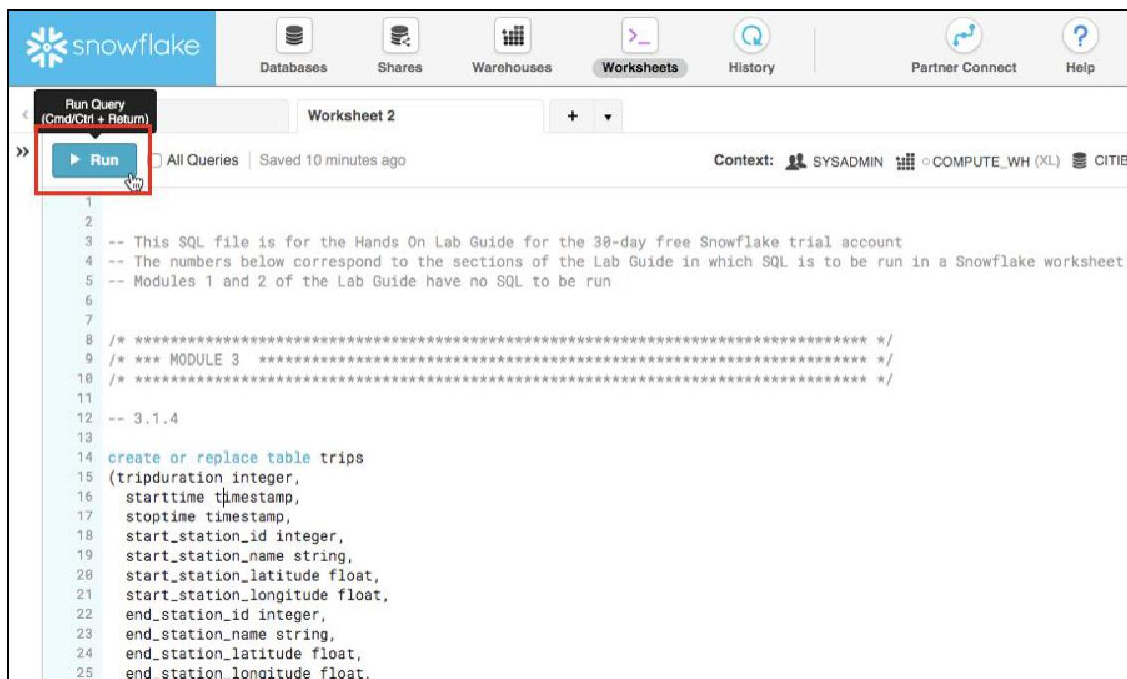
As mentioned earlier, in this experiment we will run some operations via pre-written SQL in the worksheet (as opposed to using the UI) to save time.

- 3.1.5 Run the query by placing your cursor anywhere in the command and clicking the blue “Run” button at the top of the page or by hitting Ctrl/Cmd+Enter on your keyboard.

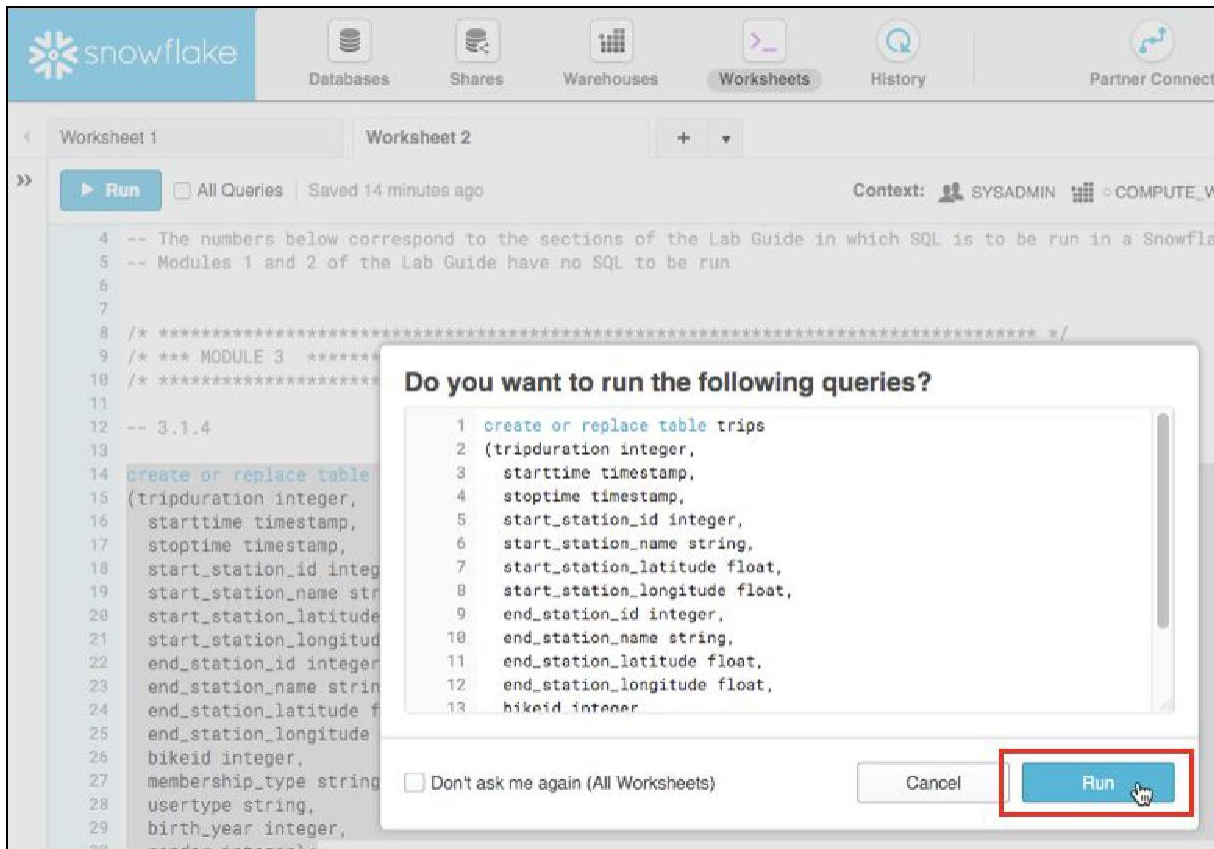


### Warning

In this experiment, never check the “All Queries” box. We run SQL queries one at a time in a specific order; not all at once.



- 3.1.6 \*If\* you highlighted the entire SQL text of the command (did not just place your cursor in the command) and ran it, a confirmation box should appear asking “Do you want to run the following queries?”. Click the blue “Run” button in the box. In the future you can keep clicking this “Run” button on this confirmation box or check the “Don’t ask me again (All Worksheets)” option in this box. As noted previously we don’t recommend that.



3.1.7 Verify that your table TRIPS has been created. At the bottom of the worksheet you should see a “Results” section which says “Table TRIPS successfully created”

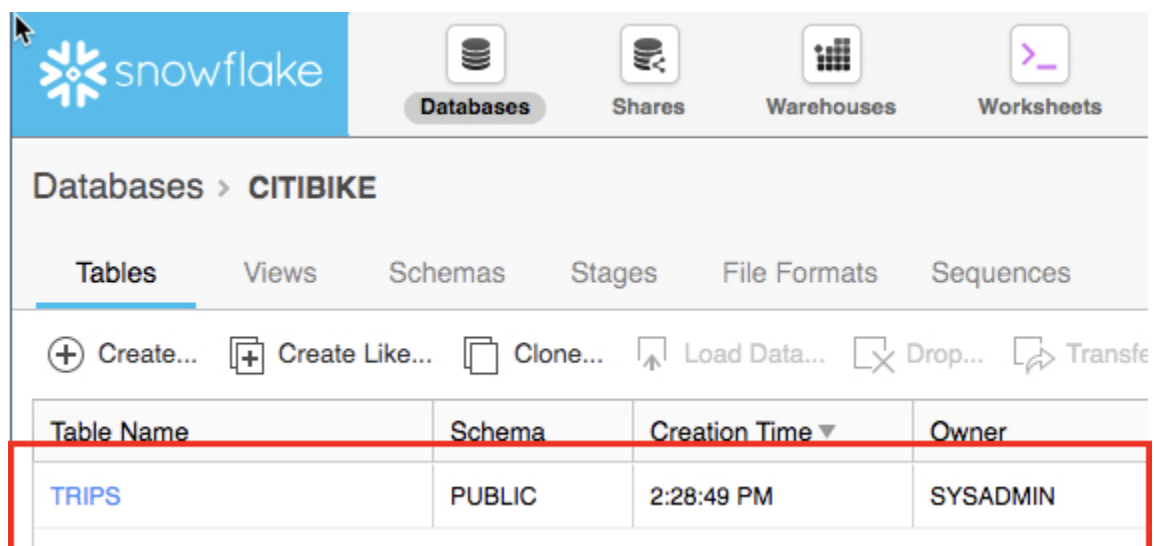


The screenshot shows the 'Results' tab in Snowflake. At the top, it indicates 'Query ID', 'SQL', '292ms' execution time, and '1 rows' returned. Below this, there are buttons for 'Filter result...', 'Download', and 'Copy'. A 'Columns' dropdown menu is also visible. The results table has two columns: 'Row' and 'status'. The first row shows '1' in the 'Row' column and 'Table TRIPS successfully created.' in the 'status' column.

Row	status
1	Table TRIPS successfully created.

3.1.8 At the top of the page, go to the Databases tab and then click on the “CITIBIKE” database link. You should see your newly created TRIPS table.

IMPORTANT: If you do not see the databases, expand your browser as they may be hidden.

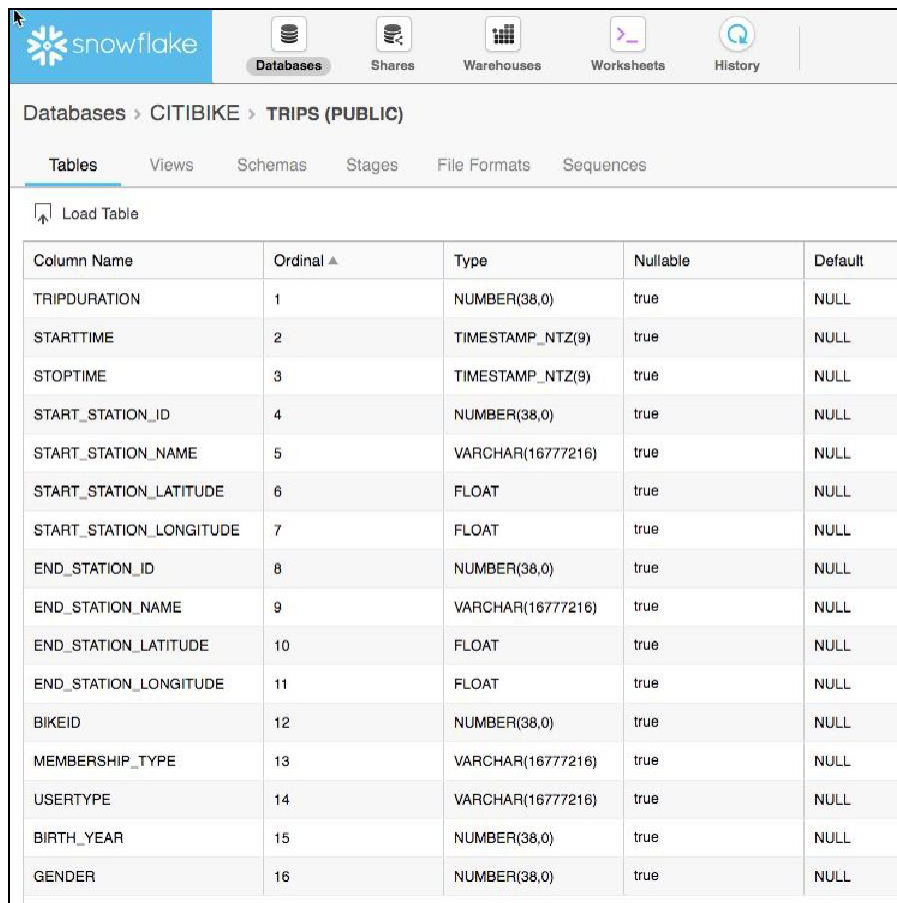


The screenshot shows the Snowflake interface. At the top, there are tabs for 'Databases', 'Shares', 'Warehouses', and 'Worksheets'. The 'Databases' tab is selected, and the 'CITIBIKE' database is chosen. Below this, there are tabs for 'Tables', 'Views', 'Schemas', 'Stages', 'File Formats', and 'Sequences'. The 'Tables' tab is selected, and a table named 'TRIPS' is listed. The table has columns for 'Table Name', 'Schema', 'Creation Time', and 'Owner'. The 'TRIPS' table is highlighted with a red box.

Table Name	Schema	Creation Time	Owner
TRIPS	PUBLIC	2:28:49 PM	SYSADMIN

3.1.9 Click on the “TRIPS” hyperlink to see the table structure you just configured for it.





The screenshot shows the Snowflake web interface. At the top, there's a navigation bar with icons for Databases, Shares, Warehouses, Worksheets, and History. Below this, the breadcrumb path is 'Databases > CITIBIKE > TRIPS (PUBLIC)'. Underneath, there are tabs for Tables, Views, Schemas, Stages, File Formats, and Sequences. The 'Tables' tab is selected, and a 'Load Table' button is visible. The main content area displays a table with 16 columns, each with a name, ordinal, type, nullable status, and default value.

Column Name	Ordinal ▲	Type	Nullable	Default
TRIPDURATION	1	NUMBER(38,0)	true	NULL
STARTTIME	2	TIMESTAMP_NTZ(9)	true	NULL
STOPTIME	3	TIMESTAMP_NTZ(9)	true	NULL
START_STATION_ID	4	NUMBER(38,0)	true	NULL
START_STATION_NAME	5	VARCHAR(16777216)	true	NULL
START_STATION_LATITUDE	6	FLOAT	true	NULL
START_STATION_LONGITUDE	7	FLOAT	true	NULL
END_STATION_ID	8	NUMBER(38,0)	true	NULL
END_STATION_NAME	9	VARCHAR(16777216)	true	NULL
END_STATION_LATITUDE	10	FLOAT	true	NULL
END_STATION_LONGITUDE	11	FLOAT	true	NULL
BIKEID	12	NUMBER(38,0)	true	NULL
MEMBERSHIP_TYPE	13	VARCHAR(16777216)	true	NULL
USERTYPE	14	VARCHAR(16777216)	true	NULL
BIRTH_YEAR	15	NUMBER(38,0)	true	NULL
GENDER	16	NUMBER(38,0)	true	NULL

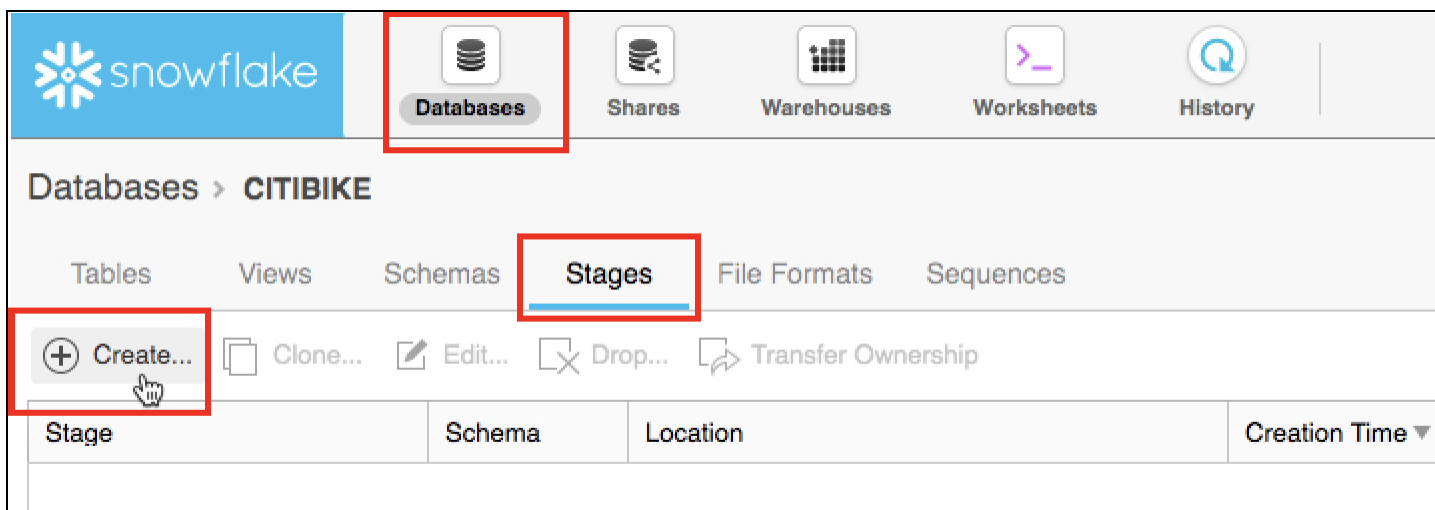
## 3.2 Create an External Stage

We are working with structured, comma-delimited data that has already been staged in a public, external S3 bucket. Before we can use this data, we first need to create a Stage that specifies the location of our external bucket.

NOTE - For this experiment we are using an AWS-East bucket. In the real-world, to prevent data egress/transfer costs, you would want to select a staging location from the same cloud provider and region that your Snowflake environment is in.

- 3.2.1 From the Databases tab, click on the “CITIBIKE” database, then click on “Stages” and click “Create...”

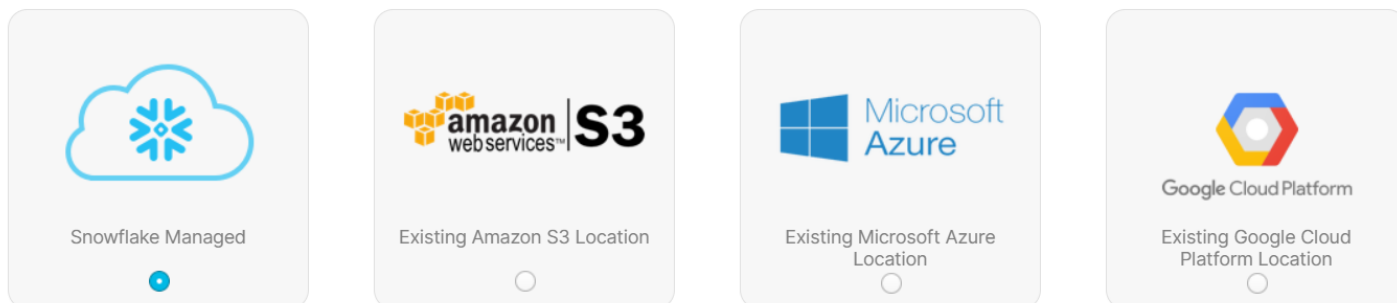




3.2.2 Select the option for “Existing Amazon S3 Location” and click “Next”:

#### Create Stage

Choose a location for files to be staged



3.2.3 On the “Create Stage” box that appears, enter/select the following settings, then click “Finish”.

Name	citibike_trips
Schema Name	PUBLIC
URL	s3://snowflake-workshop-lab/citibike-trips/

**NOTE** - The S3 bucket for this experiment is public so you can leave the key fields empty. In the “real world” this bucket would likely require key information.

Like most objects we create through the UI, you can use SQL to create the stage:

```
CREATE STAGE "CITIBIKE"."PUBLIC".citibike_trips URL = 's3://snowflake-workshop-lab/citibike-trips/';
```



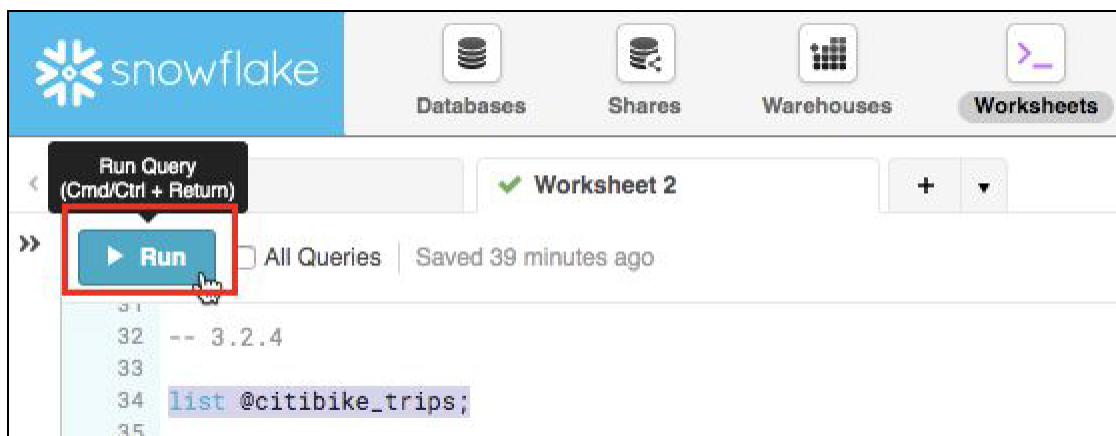
## Stages

Recall from our session discussion that stages may be internal or external. Internal are Snowflake managed, External are Azure/AWS/GCP.

The 'Create Stage' dialog box in Snowflake. It contains the following fields: Name (citibike\_trips), Schema Name (PUBLIC), URL (s3://snowflake-workshop-lab/citibike-trips), AWS Key ID, AWS Secret Key, Encryption Master Key, and Comment. At the bottom, there are buttons for 'Show SQL', 'Cancel', 'Back', and 'Finish' (which is highlighted with a red box).

3.2.4 Now let's take a look at the contents of the citibike\_trips stage. At the top of the page, click on the Worksheet tab. Then execute the following statement:

```
list @citibike_trips;
```



You should see the output in the Results window in the bottom pane:

Results

Data Preview

Open History

Query ID

SQL

810ms

376 rows

Filter result...

Download

Copy

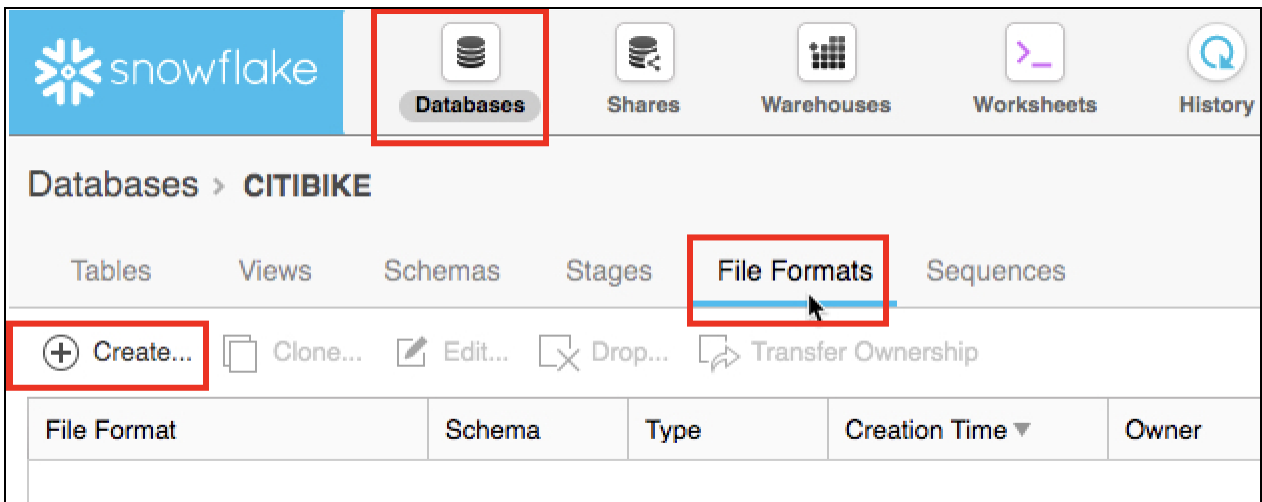
Columns

Row	name	size	md5	last_modified
1	s3://snowflake-workshop-lab/citibike-tri...	3072073	cfc69e04228a94d1337ab383a3af7472	Wed, 12 Jun 2019 16:30:58 GMT
2	s3://snowflake-workshop-lab/citibike-tri...	2877852	92a1c064a3c632f338b57d5c6531e97d	Wed, 12 Jun 2019 16:30:58 GMT
3	s3://snowflake-workshop-lab/citibike-tri...	3174598	39faac098802c1b29f2d4d99f31378be	Wed, 12 Jun 2019 16:30:58 GMT
4	s3://snowflake-workshop-lab/citibike-tri...	3031012	cd0dca1dcfa309c0bb4bd40d1265c3a9	Wed, 12 Jun 2019 16:30:58 GMT
5	s3://snowflake-workshop-lab/citibike-tri...	3005838	fb24c0cc5fb6ee54d2aa4d50265792c7	Wed, 12 Jun 2019 16:30:58 GMT
6	s3://snowflake-workshop-lab/citibike-tri...	3099881	441efe806352c57a50c4f31afcccb2e3	Wed, 12 Jun 2019 16:30:58 GMT
7	s3://snowflake-workshop-lab/citibike-tri...	3060952	372765a1ca713eeb1d56f79e0956e48f	Wed, 12 Jun 2019 16:30:59 GMT

## Create a File Format

Before we can load the data into Snowflake, we have to create a File Format that matches the data structure.

- 3.2.5 From the Databases tab, click on the CITIBIKE database hyperlink. Then click on “File Formats”. Then click “Create”.



- 3.2.6 On the resulting page, we then create a file format. In the box that appears, leave all the default settings as-is but make the changes below:

Name: **CSV**

Field optionally enclosed by: **Double Quote**

Null string: <Delete the existing text in this field so it is empty>

[ ] Error on Column Count Mismatch: <uncheck this box>

**IMPORTANT:** If you do not see the “Error on Column Count Mismatch” box, scroll down in the dialogue box

When you are done, the box should look like:

**Create File Format**

Name\* CSV

Schema Name PUBLIC

Format Type CSV

Compression Method Auto

Column separator Comma

Row separator New Line

Header lines to skip 0

Field optionally enclosed by Double Quote

Null String

☐ Trim space before and after

☐ Error on Column Count Mismatch

Escape Character None

Escape Unenclosed Field Backslash

Date Format Auto

Timestamp Format Auto

Comment

[Show SQL](#) Cancel Finish

Then click on the “Finish” button to create the file format.