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 realworld, 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:

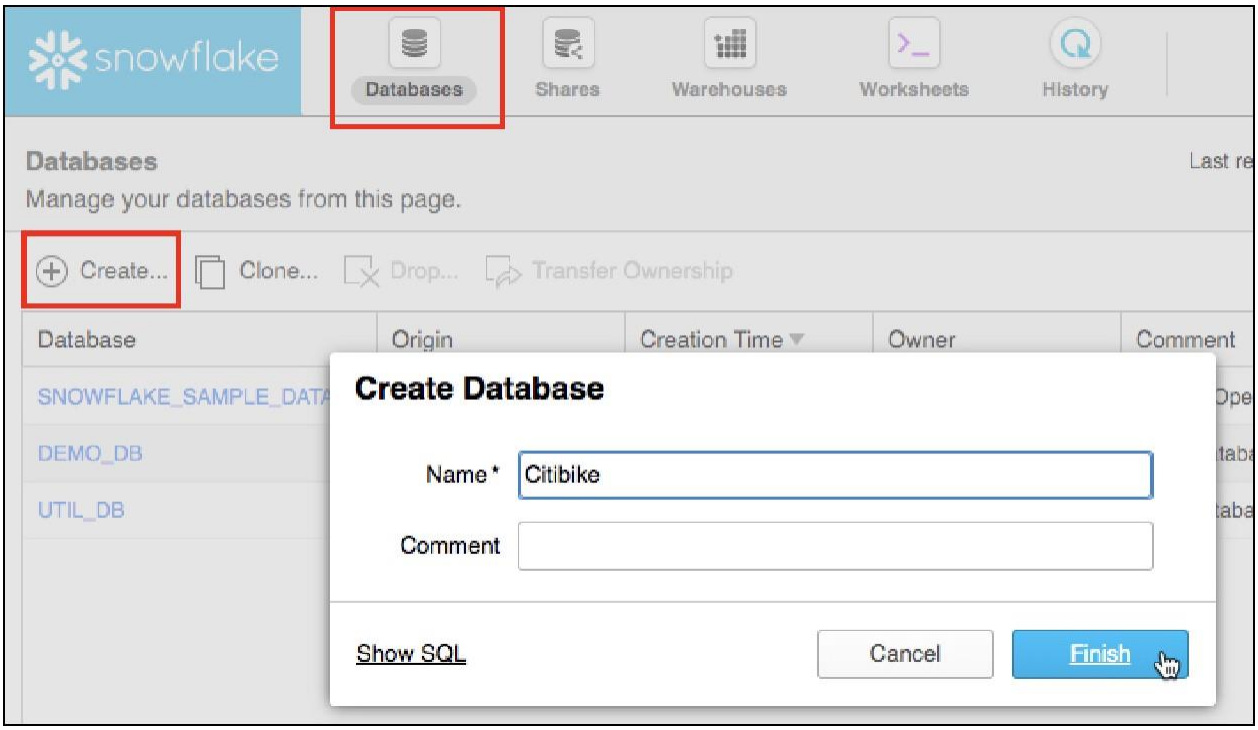
```
"tripduration","starttime","stoptime","start station id","start station name","start
station latitude","start station longitude","end station id","end station name","end
station latitude","end station
longitude","bikeid","name_localizedValue0","usertype","birth year","gender"
196,"2018-01-01 00:01:51","2018-01-01 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.

You'll find the Create Database command in our Foundations SQL that we downloaded earlier and will use for our experiments. In the Classic Console there was a dialog for creating objects, but that's been removed along with other creation dialogs in the new UI that is the only view for Trial Accounts.



3.1.2 At the top of the Snowflake Worksheets UI, Check your Context and ensure it is set correctly. If it shows the Select options, click on them and set as noted below to ACCOUNT_ADMIN, COMPUTE_WH, CITIBIKE as noted. You can click on any of the Select areas and enter the details for the Role/Warehouse/Database/Schema.

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

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.

```
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"

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.

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

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 In the old Classic UI there was dialogs for creating stages, but that’s not available in Trial Accounts, so we’ll have to use SQL.

3.2.2 You’ll navigate to the Worksheet that we’ve loaded our Foundation SQL and find the create stage statement

3.2.3 Alternatively we could just enter that as

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

NOTE - The S3 bucket for this experiment is public, so we don’t need credential information. In the “real world” this bucket would likely require key information.

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

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:

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 Notice that in the Classic Console there were dialogs for creating File Formats, but we'll use the SQL in the Foundations SQL.

```
CREATE OR REPLACE FILE FORMAT csv
  TYPE = CSV
  FIELD_DELIMITER = ','
  FIELD_OPTIONALLY_ENCLOSED_BY = ""
  ERROR_ON_COLUMN_COUNT_MISMATCH = FALSE
  EMPTY_FIELD_AS_NULL = TRUE
  SKIP_HEADER = 1;
```