Experiment 4: Loading Data

Overview

For this experiment, we will use a data warehouse and the COPY command to initiate bulk loading of the structured data into the Snowflake table we just created.

4.1 Resize and Use a Warehouse for Data Loading

For loading data, compute power is needed for processing. Snowflake's compute nodes are called Warehouses and they can be dynamically sized up or out according to workload, whether the workload be loading data, running a query, or performing a DML operation. And each workload can have its own data warehouse so there is no resource contention.

4.1.1 Navigate to the Warehouses tab. Note the "Create…" option at the top is where you can quickly create a new warehouse. However, we want to use the existing warehouse "COMPUTE_WH" which comes with the 30-day trial environment.

Click on the row of this "COMPUTE_WH" warehouse (not the blue hyperlink that says "COMPUTE_WH") so the entire row is highlighted. Then click on the "Configure..." text above it to see the configuration detail of the "COMPUTE_WH". We will use this warehouse to load in the data from AWS S3.

4.1.2 Let's walk through the settings of this warehouse as there is a lot of functionality here, much of which is unique to Snowflake versus other data warehouses.

NOTE - If you do not have a Snowflake Edition of Enterprise or greater, you will *NOT* see the "Maximum Clusters" or "Scaling Policy" configurations from the screenshot below. Multi-clustering is not utilized in this experiment, but we will still discuss it as it is a key capability of Snowflake.

- The "Size" drop-down is where the size of the warehouse is selected. For larger data loading operations or more compute-intensive queries, a larger warehouse will be needed. The t-shirt sizes translate to underlying compute nodes, either AWS EC2 or Azure Virtual Machines. The larger the t-shirt size, the more compute resources from the cloud provider are allocated to that warehouse. As an example, the 4-XL option allocates 128 nodes. Also, this sizing can be changed up or down on the fly with a simple click.
- If you have Snowflake Edition or greater you will see the Maximum Clusters section. This is where you can set up a single warehouse to be multi-cluster up to 10 clusters. As an example, if the 4-XL warehouse we just mentioned was assigned a maximum cluster size of 10, it could scale up to be 1280 (128 * 10) AWS EC2 or Azure VM machines modes powering that warehouse...and it can do this in seconds! Multi-cluster is ideal for concurrency scenarios, such as many business analysts simultaneously running different queries using the same warehouse. In this scenario, the various queries can be allocated across the multiple clusters to ensure they run fast.

- The final sections allow youf to automatically suspend the warehouse so it suspends (stops) itself when not in use and no credits are consumed. There is also an option to automatically resume (start) a suspended warehouse so when a new workload is assigned to it, it will automatically start back up. This functionality enables Snowflake's fair "pay as you use" compute pricing model which enables customers to minimize their data warehouse costs.

Snowflake Compute vs Other Warehouses

Many of the warehouse/compute capabilities we just covered, like being able to create, scale up and out, and auto-suspend/resume warehouses are things that are simple in Snowflake and can be done in seconds. Yet for on-premise data warehouses these capabilities are very difficult (or impossible) to do as they require significant physical hardware, over-provisioning of hardware for workload spikes, significant configuration work, and more challenges. Even other cloud data warehouses cannot scale up and out like Snowflake without significantly more configuration work and time.

Warning - Watch Your Spend!

If you do any of the following without good reason, you may burn through your $400 of free credits more quickly than desired:

- Disable auto-suspend. If auto-suspend is disabled, your warehouses will continue to run and consume credits even when not being utilized.
- Use a warehouse size that is excessive given the workload. The larger the warehouse, the more credits are consumed.e

4.1.3 We are going to use this data warehouse to load the structured data into Snowflake. However, we are first going to decrease the size of the warehouse to reduce the compute power it contains. Then in later steps we will note the time this load takes, then re-do the same load operation with a larger warehouse, and observe how much faster the load is with a larger warehouse.

Change the Size of this data warehouse from X-Large to Small. Then click the "Finish" button.

4.2 Load the Data

Now we can run a COPY command to load the data into the TRIPS table we created earlier.

4.2.1 Via the top of the UI, navigate back to the Worksheets tab and click on it. Make sure the context is correct by setting these in the top right of the worksheet:

Role: ACCOUNTADMIN
Warehouse: COMPUTE_WH (S)

Database: CITIBIKE
Schema = PUBLIC

4.2.2 Execute the following statements in the worksheet to load the staged data into the table. This may take up to 30 seconds.

```
copy into trips from @citibike_trips
file_format=CSV;
```

In the above, we will see that we ran into an issue because there are JSON and Parquet files in that folder. The following error would occur if we didn't add a pattern, which we'll do next.

Found character '\u0006' instead of field delimiter ',' File 'citibike-tripsparquet/2013/06/05/data_01a19496-0601-8b21-003d-9b03003c624a_206_2_0.snappy.parquet', line 323, character 593 Row 323, column "TRIPS"["START_STATION_ID":4] If you would like to continue loading when an error is encountered, use other values such as 'SKIP_FILE' or 'CONTINUE' for the ON_ERROR option. For more information on loading options, please run 'info loading_data' in a SQL client.

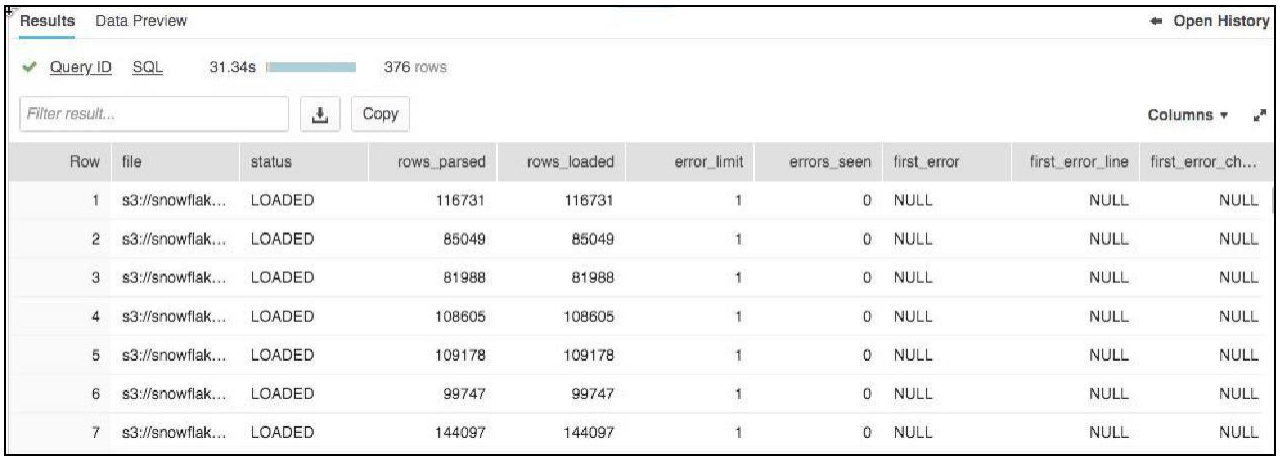
We could experiment with another option if we go back and recreate our citibike_trips without the trailing slash in the stage URL. Without the trailing slash the above command would see the Parquet files that are in another subfolder and cause an issue in the data loading.

```
copy into trips from @citibike_trips
file_format=CSV PATTERN='.*[.]csv.gz';
```

Or for a more refined set that specified the year pattern we could use the following

```
COPY INTO trips FROM
@citibike_trips
file_format = CSV
pattern = '.*trips_2018.*[.]csv[.]gz'
ON_ERROR = CONTINUE;
```

In the Results window, you should see the status of the load:



Results | Data Preview | Open History

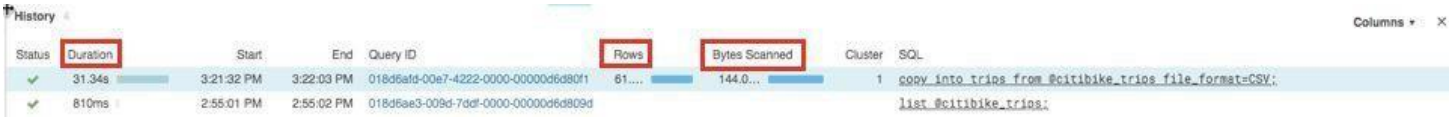
Query ID | SQL | 31.34s | 376 rows

Filter result... | Copy | Columns

Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_ch...
1	s3://snowflak...	LOADED	116731	116731	1	0	NULL	NULL	NULL
2	s3://snowflak...	LOADED	85049	85049	1	0	NULL	NULL	NULL
3	s3://snowflak...	LOADED	81988	81988	1	0	NULL	NULL	NULL
4	s3://snowflak...	LOADED	108605	108605	1	0	NULL	NULL	NULL
5	s3://snowflak...	LOADED	109178	109178	1	0	NULL	NULL	NULL
6	s3://snowflak...	LOADED	99747	99747	1	0	NULL	NULL	NULL
7	s3://snowflak...	LOADED	144097	144097	1	0	NULL	NULL	NULL

4.2.3 Once the load is done, at the bottom right of the worksheet click on the small arrow next to the “Open History” text to show the history of Snowflake operations performed in that specific worksheet.

4.2.4 In the History window see the “copy into trips from @citibike_trips file_format=CSV;” SQL query you just ran and note the duration, bytes scanned and rows. Use the slider on the left side of the pane to expand it if needed to see all the detail in it.



4.2.5 Go back to the worksheet to clear the table of all data and metadata, by using the TRUNCATE TABLE command. truncate

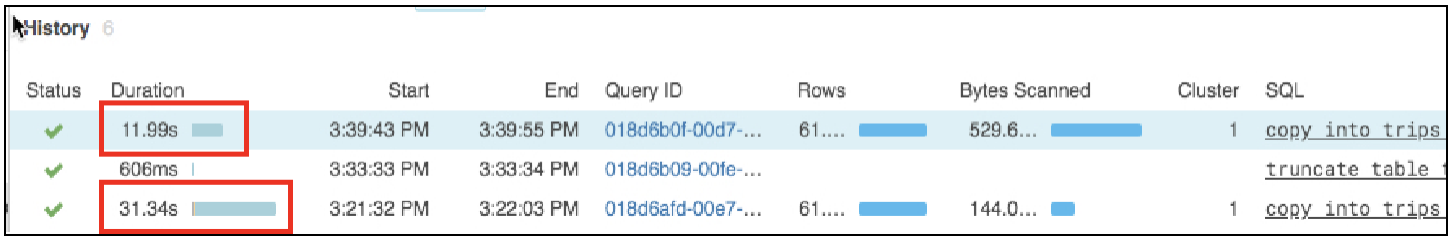
table trips;

4.2.6 At the top of the UI go to the Warehouses tab, then click on “Configure..” Resize the warehouse to size Large and click “Finish”. This warehouse is four times larger than the Small size.

4.2.7 Go back to the Worksheets tab at the top of the UI and click on it. Execute the following statements in the worksheet to load the same data again.

```
copy into trips from @citibike_trips
file_format=CSV PATTERN='.*[.]csv.gz';
```

4.2.8 Once the load is done, at the bottom of the worksheet in the History window compare the times between the two loads. The one with the Large warehouse was significantly faster.

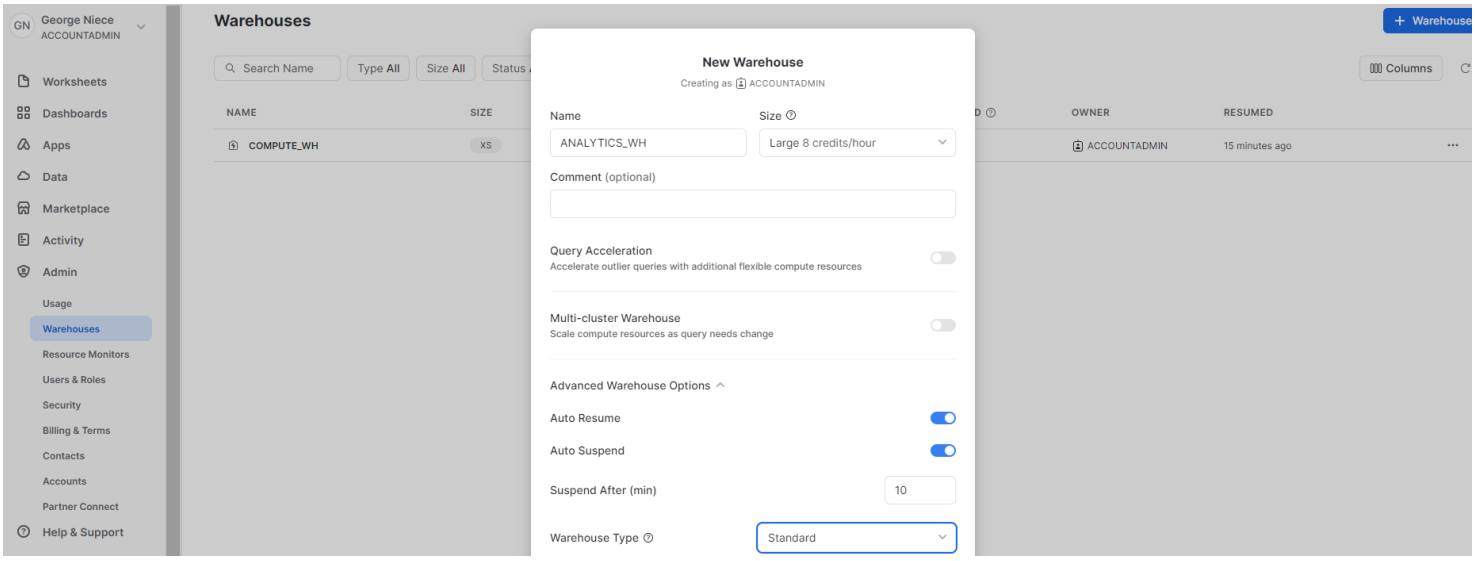


4.3 Create a New Warehouse for Data Analytics

Going back to the experiment story, let's assume the Citi Bike team wants to ensure no resource contention between their data loading/ETL workloads and the analytical end users using BI tools to query Snowflake. As mentioned earlier, Snowflake can easily do this by assigning different, appropriately-sized warehouses to different workloads. Since Citi Bike already has a warehouse for data loading, let's create a new warehouse for the end users running analytics. We will then use this warehouse to perform analytics in the next experiment.

4.3.1 Navigate to Admin -> Warehouses tab then "+Warehouse". Name it "ANALYTICS_WH" with size "Large". If you have Snowflake edition Enterprise or greater, you will see a setting for "Maximum Clusters". Set this to "1".

Leave the other settings in their default settings. It should look like:



Then click on the "Create Warehouse" button to create the warehouse.