Experiment 6: Data Wrangling

Overview

In this experiment we'll be working with semi-structured data, views, Joins, and JSON

NOTE - The first steps here are similar to prior Experiments 3 (Preparing to Load Data) and 4 (LoadingData) but we will do most of it via SQL in the worksheet, as opposed to via the UI, to save time.

Going back to the experiment "story", the Citi Bike analytics team wants to see how weather impacts ridecounts. To do this, in this experiment we will:

- Load weather data in JSON format held in a public S3 bucket
- Create a View and query the semi-structured data using SQL dot notation
- Run a query that joins the JSON data to the TRIPS data from a prior experiment of this guide
- See how weather impacts trip counts

The JSON data consists of weather information provided by OpenWeatherMap detailing the historical conditions of New York City from 2016-07-05 to 2019-06-25. It is also staged on AWS S3 where the data represents 57.9k rows, 61 objects, and 2.5MB total size compressed.

The raw JSON in GZ files and in a text editor looks like:


```
{"city":{"coord":{"lat":43.000351,"lon":-75.499901},"country":"US","findname":"NEW YORK","id":5128638,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":93,"pressure":1008,"temp":293.47,"temp_max":295.37,"temp_min":292.04},"time":
1561467737,"weather":[{"description":"moderate rain","icon":"10d","id":501,"main":"Rain"}],"wind":{"deg":170,"speed":4.1}}
{"city":{"coord":{"lat":40.714272,"lon":-74.005966},"country":"US","findname":"NEW YORK","id":5128581,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":94,"pressure":1010,"temp":295.16,"temp_max":296.15,"temp_min":294.15},"time":
1561467737,"weather":[{"description":"light rain","icon":"10d","id":500,"main":"Rain"},{"description":"mist","icon":"50d","id":
701,"main":"Mist"}],"wind":{"deg":0,"speed":2.1}}
{"city":{"coord":{"lat":43.000351,"lon":-75.499901},"country":"US","findname":"NEW YORK","id":5128638,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":94,"pressure":1008,"temp":294.58,"temp_max":297.04,"temp_min":292.04},"time":
1561471336,"weather":[{"description":"overcast clouds","icon":"04d","id":804,"main":"Clouds"}],"wind":{"deg":270,"speed":3.1}}
{"city":{"coord":{"lat":40.714272,"lon":-74.005966},"country":"US","findname":"NEW YORK","id":5128581,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":100,"pressure":1010,"temp":295.37,"temp_max":296.48,"temp_min":294.26},"time":
1561471336,"weather":[{"description":"mist","icon":"50d","id":701,"main":"Mist"}],"wind":{"deg":170.797,"speed":0.4}}
```


SEMI-STRUCTURED DATA
Snowflake can easily load and query semi-structured data, such as JSON, Parquet, or Avro, without transformation. This is important because an increasing amount of business-relevant data being generated today is semi-structured, and many traditional data warehouses cannot easily load and query this sort of data. With Snowflake it is easy!

6.1 Create a Database and Table

6.1.1 First, via the Worksheet, let's create a database called WEATHER that will be used for storing the unstructured data.

```
create database weather;
```

6.1.2 Set the context appropriately within the Worksheet.

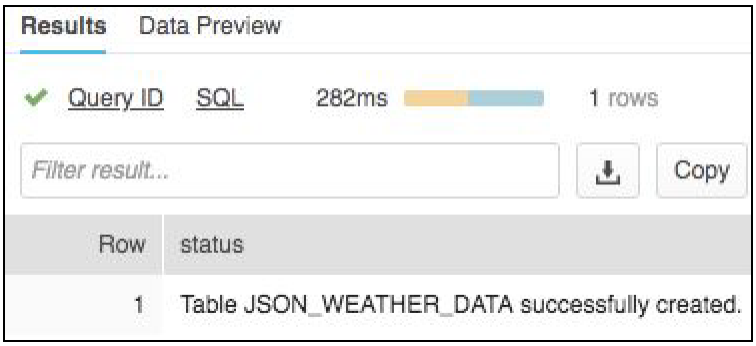
```
use warehouse compute_wh;
use database weather;
use schema public;
```

6.1.3 Via the worksheet, let's now create a table called JSON_WEATHER_DATA that will be used for loading the JSON data. In the worksheet, run the SQL text below. Snowflake has a special column type called VARIANT which will allow us to store the entire JSON object and eventually query it directly.

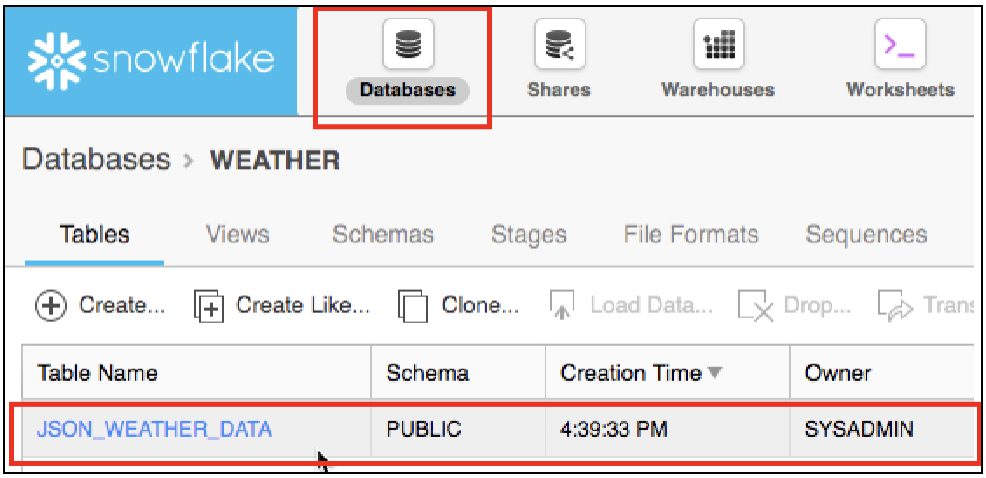
```
create table json_weather_data (v variant);
```

Semi-Structured Data Magic
Snowflake's VARIANT data type allows Snowflake to ingest semi-structured data without having to pre-define the schema.
https://docs.snowflake.com/en/sql-reference/data-types-semistructured.html#variant

6.1.4 Verify that your table JSON_WEATHER_DATA has been created. At the bottom of the worksheet you should see a "Results" section which says "Table JSON_WEATHER_DATA successfully created"



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



6.2 Create an External Stage

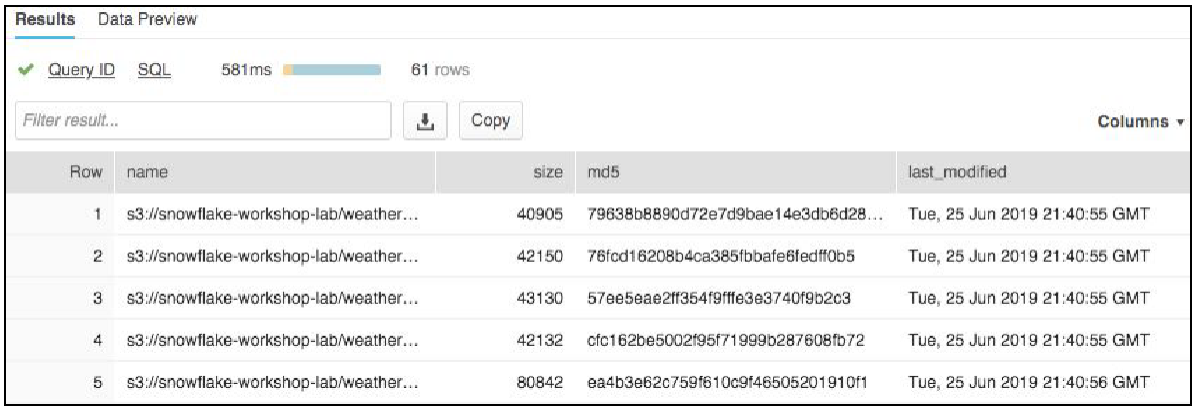
6.2.1 Via the Worksheet create a stage from where the unstructured data is stored on AWS S3.

```
create stage nyc_weather  url = 's3://snowflake-workshop-lab/weather-nyc';
```

6.2.2 Now let’s take a look at the contents of the nyc_weather stage. At the top of the page, click on the Worksheets tab. In the worksheet, then execute the following statement with a LIST command to see the list of files:

```
list @nyc_weather;
```

You should see the output in the Results window in the bottom pane showing many gz files from S3:



Row	name	size	md5	last_modified
1	s3://snowflake-workshop-lab/weather...	40905	79638b8890d72e7d9bae14e3db6d28...	Tue, 25 Jun 2019 21:40:55 GMT
2	s3://snowflake-workshop-lab/weather...	42150	76fcd16208b4ca385fbbafe6fedff0b5	Tue, 25 Jun 2019 21:40:55 GMT
3	s3://snowflake-workshop-lab/weather...	43130	57ee5eae2ff354f9fffe3e3740f9b2c3	Tue, 25 Jun 2019 21:40:55 GMT
4	s3://snowflake-workshop-lab/weather...	42132	cfc162be5002f95f71999b287608fb72	Tue, 25 Jun 2019 21:40:55 GMT
5	s3://snowflake-workshop-lab/weather...	80842	ea4b3e62c759f610c9f46505201910f1	Tue, 25 Jun 2019 21:40:56 GMT

6.3 Loading and Verifying the Unstructured Data

For this section, we will use a warehouse to load the data from the S3 bucket into the Snowflake table we just created.

6.3.1 Via the worksheet, run a COPY command to load the data into the JSON_WEATHER_DATA table we created earlier.

Note how in the SQL here we can specify a FILE FORMAT object inline. In the prior experiment where we loaded structured data, we had to define a file format in detail. Butbecause the JSON data here is well-formatted, we use default settings and simply specify the JSON type.

```
copy into json_weather_data
from @nyc_weather
file_format = (type=json);
```

6.3.2 Take a look at the data that has been loaded.

```
select * from json_weather_data limit 10;
```

6.3.3 Click on one of the values. Notice how the data is stored in raw JSON format. Click “Done” when finished.

6.4 Create a View and Query Semi-Structured Data

Let's look at how Snowflake allows us to create a view and also query the JSON data directly using SQL.

Views & Materialized Views

A View allows the result of a query to be accessed as if it were a table. Views can help you: present data to end users in a cleaner manner (like in this experiment wewill present "ugly" JSON in a columnar format), limit what end users can view ina source table for privacy/security reasons, or write more modular SQL.

There are also Materialized Views in which SQL results are stored, almost as though the results were a table. This allows faster access, but requires storagespace. Materialized Views require Snowflake Enterprise Edition or higher.

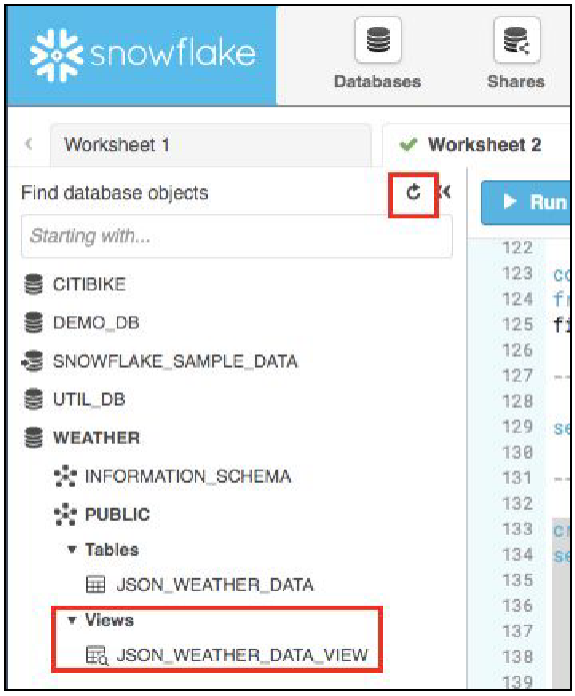
6.4.1 From the Worksheet tab, go into the worksheet and run the following command. It will create a view of the unstructured JSON weather data in a columnar view so it is easier for analysts to understand and query. FYI - the city ID for New York City is 5128638.

```
create view json_weather_data_view as
select
  v:time::timestamp as observation_time,
  v:city.id::int as city_id,
  v:city.name::string as city_name,
  v:city.country::string as country,
  v:city.coord.lat::float as city_lat,
  v:city.coord.lon::float as city_lon,
  v:clouds.all::int as clouds,
  (v:main.temp::float)-273.15 as temp_avg,
  (v:main.temp_min::float)-273.15 as temp_min,
```

```
  (v:main.temp_max::float)-273.15 as temp_max,
  v:weather[0].main::string as weather,
  v:weather[0].description::string as weather_desc,
  v:weather[0].icon::string as weather_icon,
  v:wind.deg::float as wind_dir,
  v:wind.speed::float as wind_speed
from json_weather_data
where city_id = 5128638;
```

6.4.2 The prior step showed how you can use SQL dot notation (**v.city.coord.lat**) to pull out values at lower levels in the JSON hierarchy. This allows us to treat each field as if it were a column in a relational table.

6.4.3 Verify the view at the top left of the UI where the new view should appear just under the table json_weather_data. You may need to expand and and/or refresh the database objects browser in order to see it.



6.4.4 Via the worksheet, verify the view with the following query. Notice the results look just like a regular structured data source (NOTE - your result set may have different observation_time values)

```
select * from json_weather_data_view
where date_trunc('month',observation_time) = '2018-01-01'
limit 20;
```

Results Data Preview

Query ID SQL 352ms 20 rows

Filter result... Copy

Row	OBSERVATION_TIME	CITY_ID	CITY_NAME	COUNTRY	CITY_LAT	CITY_LON	CLOUDS
1	2018-01-03 01:02:38.000	5128638	New York	US	43.000351	-75.499901	90
2	2018-01-03 02:05:20.000	5128638	New York	US	43.000351	-75.499901	90
3	2018-01-15 13:04:58.000	5128638	New York	US	43.000351	-75.499901	1
4	2018-01-15 14:03:24.000	5128638	New York	US	43.000351	-75.499901	1
5	2018-01-14 04:02:46.000	5128638	New York	US	43.000351	-75.499901	1
6	2018-01-14 05:04:41.000	5128638	New York	US	43.000351	-75.499901	1

6.5 Use a Join Operation to Correlate Against Data Sets

We will now join the JSON weather data to our CITIBIKE.PUBLIC.TRIPS data to determine the answer to our original question of how weather impacts the number of rides.

6.5.1 Run the command below to join WEATHER to TRIPS and count the number of trips associated with certain weather conditions .

Note - Since we are still in a worksheet use the WEATHER database as default, we will fully qualify our reference to the TRIPS table by providing its database and schema name.

```
select weather as conditions
    ,count(*) as num_trips
 from citibike.public.trips
 left outer join json_weather_data_view
    on date_trunc('hour', observation_time) = date_trunc('hour', starttime)
 where conditions is not null
 group by 1 order by 2 desc;
```

Results | Data Preview

Query ID | SQL | 435ms | 10 rows

Filter result... | Copy | Columns

Row	CONDITIONS	NUM_TRIPS
1	Clear	9249531
2	Clouds	8934964
3	Rain	3206720
4	Snow	1380418
5	Mist	798487
6	Fog	362496
7	Thunderstorm	230539
8	Drizzle	98779
9	Haze	40724
10	Smoke	2871

6.5.2 The Citi Bike initial goal was to see if there was any correlation between the number of bike rides and weather by analyzing both ridership and weather data. Per the table above we have a clear answer. As you would expect, the number of bike rides people take are significantlyhigher when the weather is good!