Snowflake Introduction

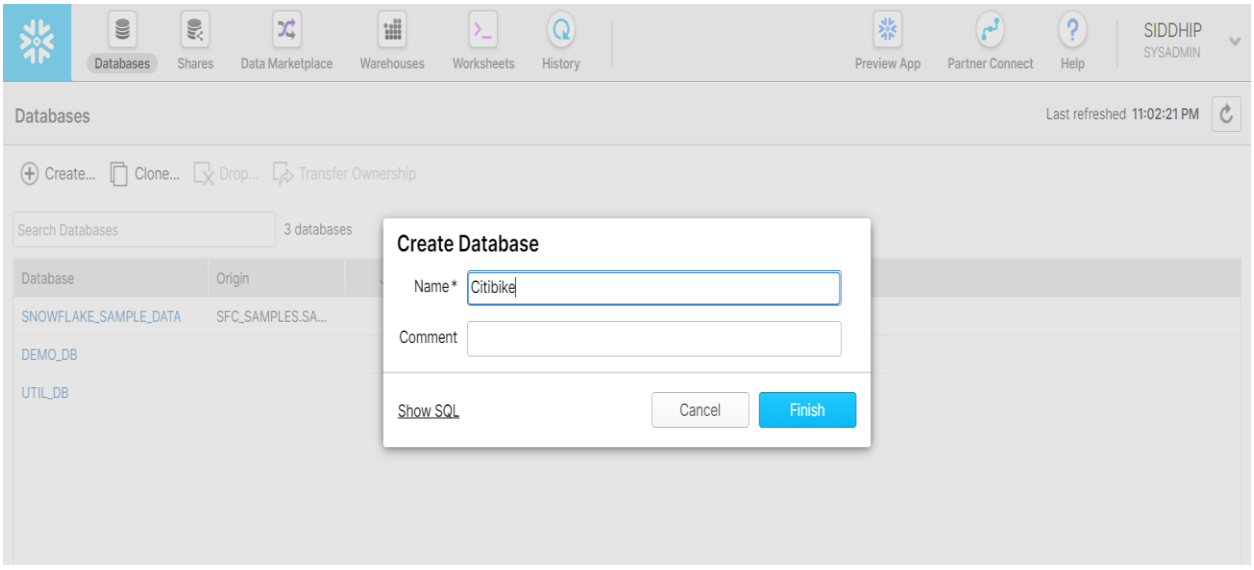
Summary of the tutorial

Load structured .csv data from rider transactions into Snowflake. Work on open-source, semi-structured JSON weather data to determine if there is any correlation between the number of bike rides and the weather.

Preparing to Load Data

Create a Database and Table

Created a database called Citibike



Context settings will help to determine what elements the user can see and run from each worksheet.

Go to Worksheet Tab -> Run Create table SQL -> Creates Trip table to store comma delimited data.

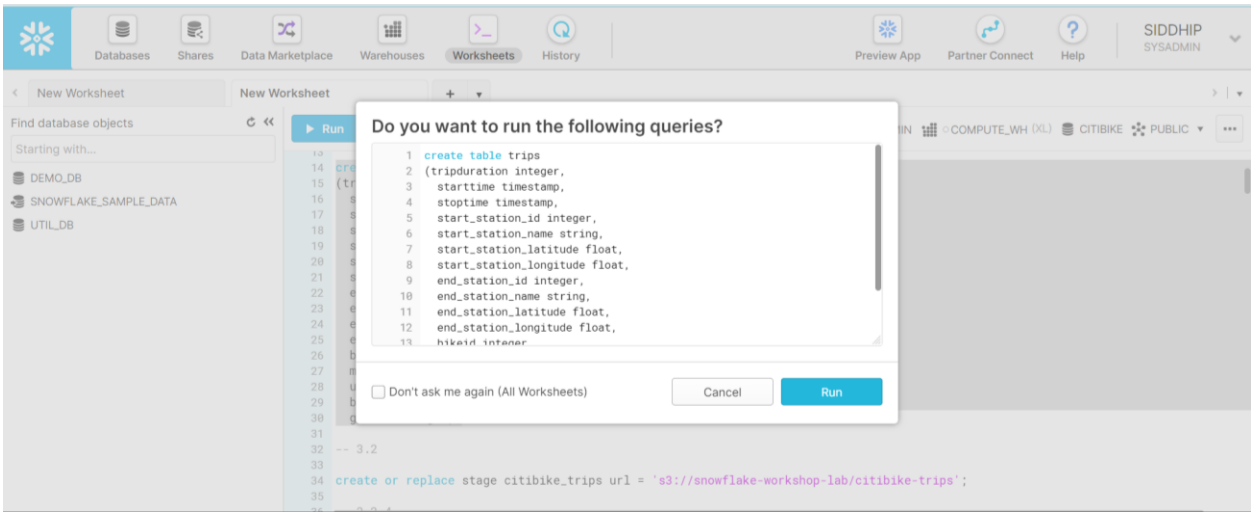
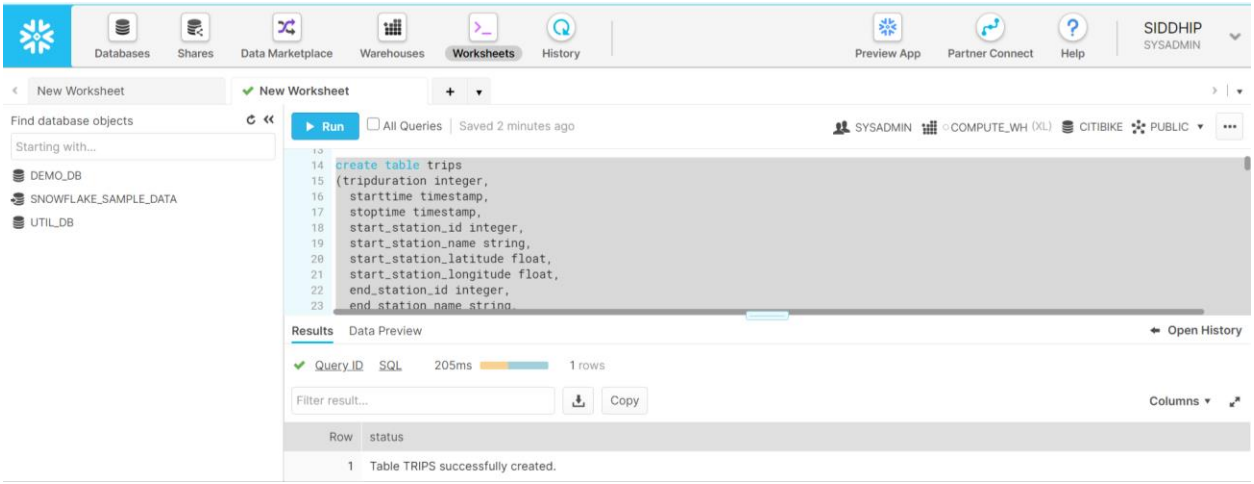
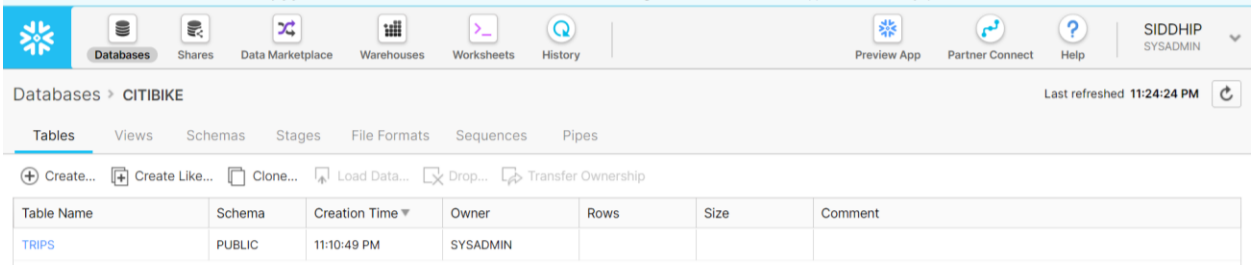
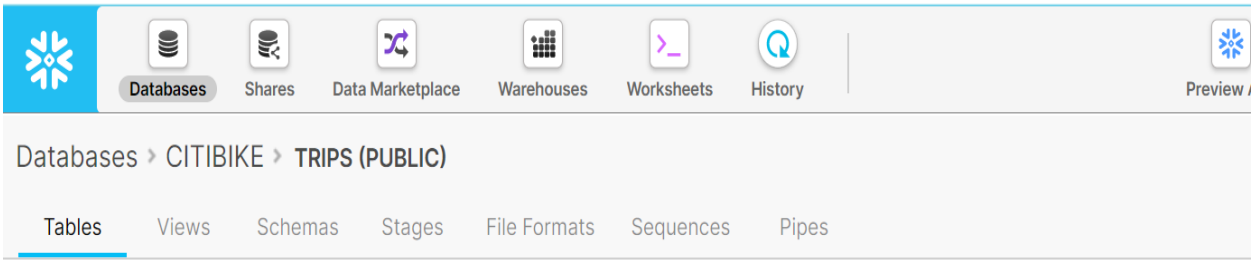


Table Trips was successfully created.



TRIPS table in CITIBIKE Database





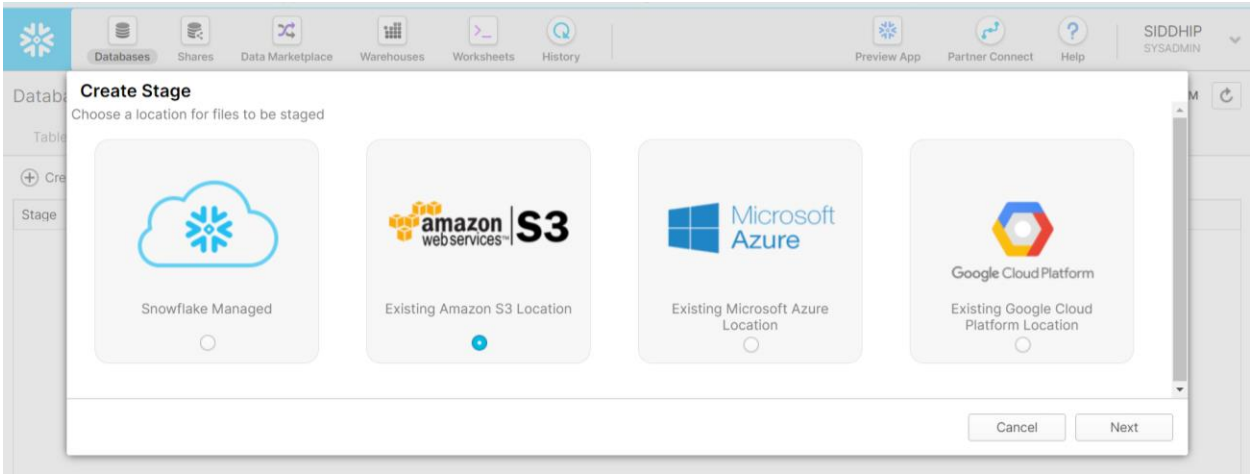
Column Name	Ordinal ▲	Type	Nullable	Default	Comment
TRIPDURATION	1	NUMBER(38,0)	true	NULL	
STARTT	2	TIMESTAMP_NTZ(9)	true	NULL	
STOPTIME	3	TIMESTAMP_NTZ(9)	true	NULL	
START_STATION_ID	4	NUMBER(38,0)	true	NULL	
START_STATION_NAME	5	VARCHAR(16777216)	true	NULL	
START_STATION_LATITUDE	6	FLOAT	true	NULL	
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	

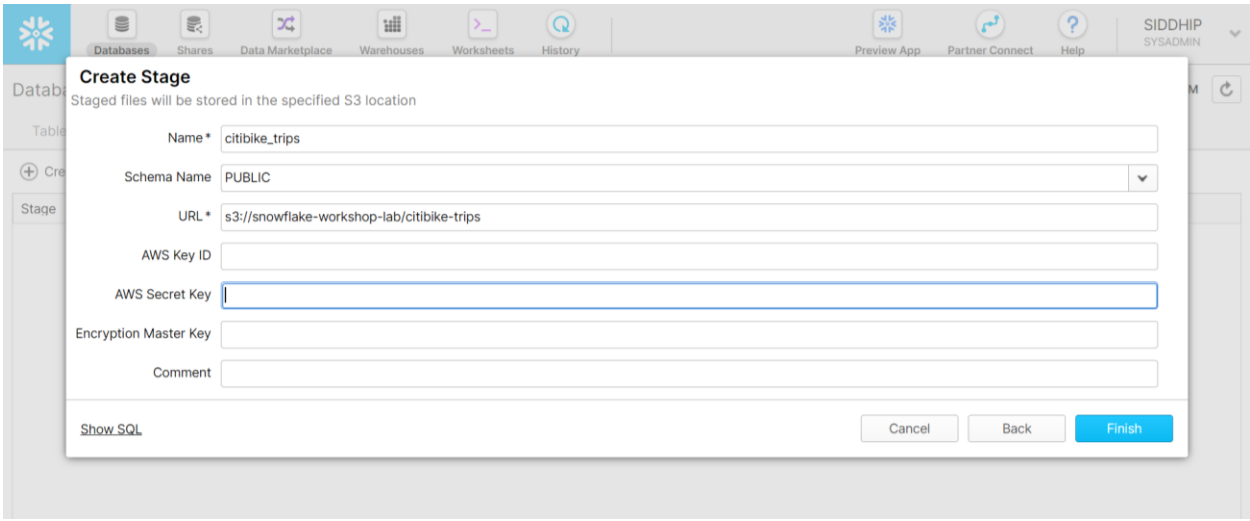
Table Name: TRIPS

Create an external stage

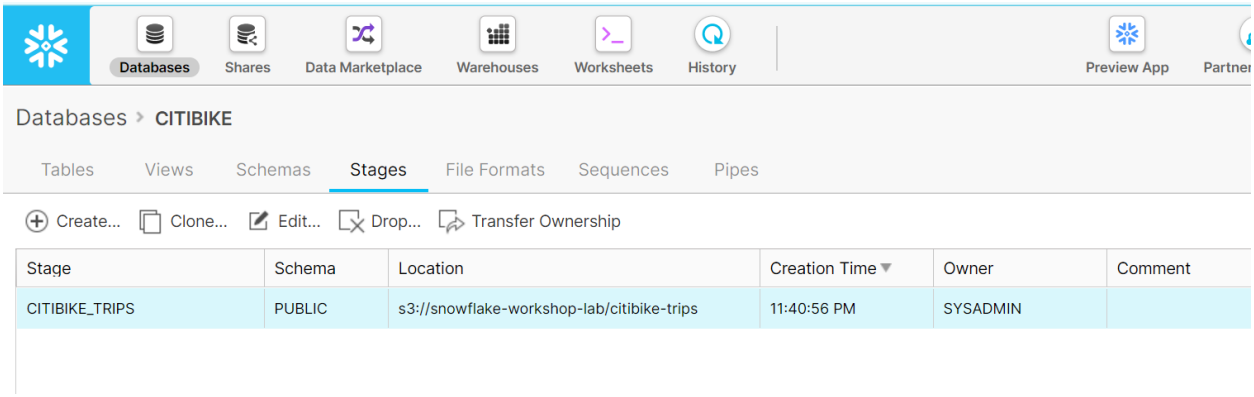
Comma delimited file already staged in a public external S3 bucket. Create a stage to specify the location of our external bucket.

Go to Database -> Stage -> Create -> Select stage location as AWS S3 ->Click Next

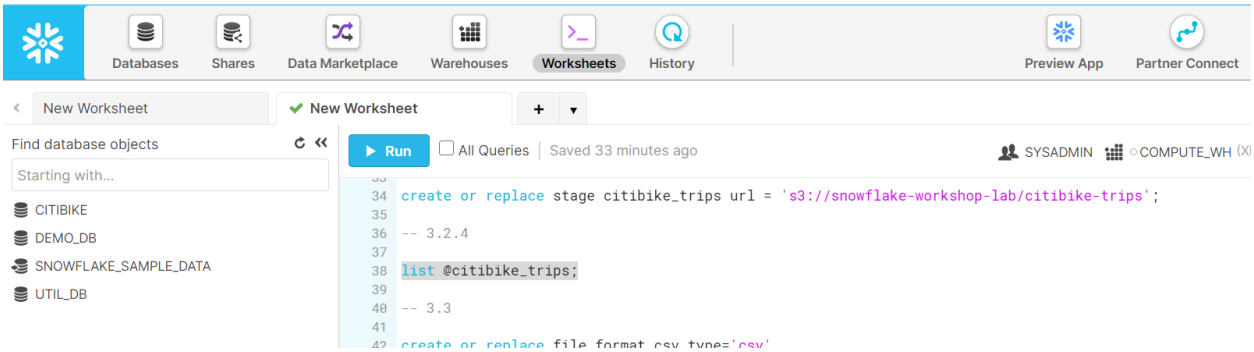


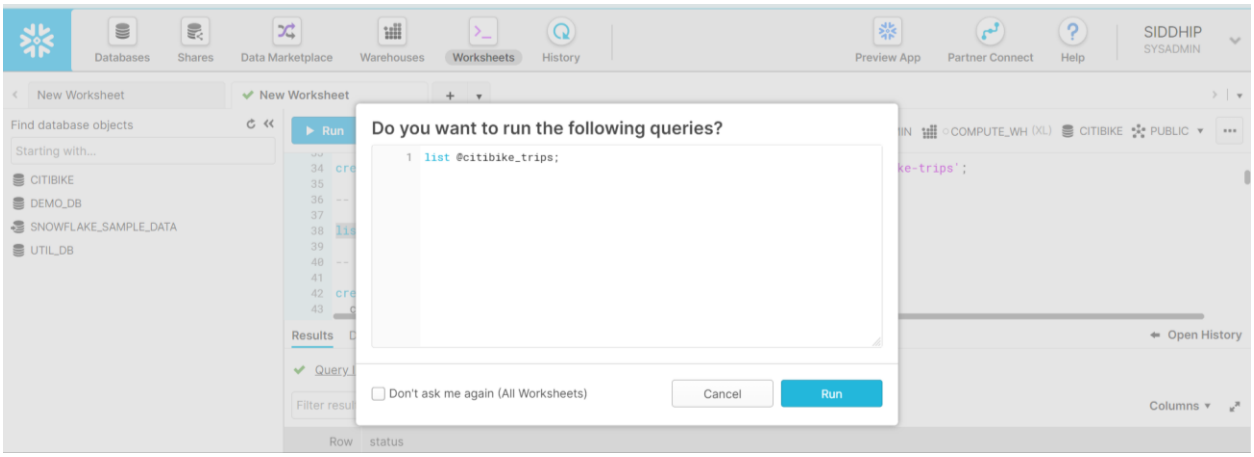


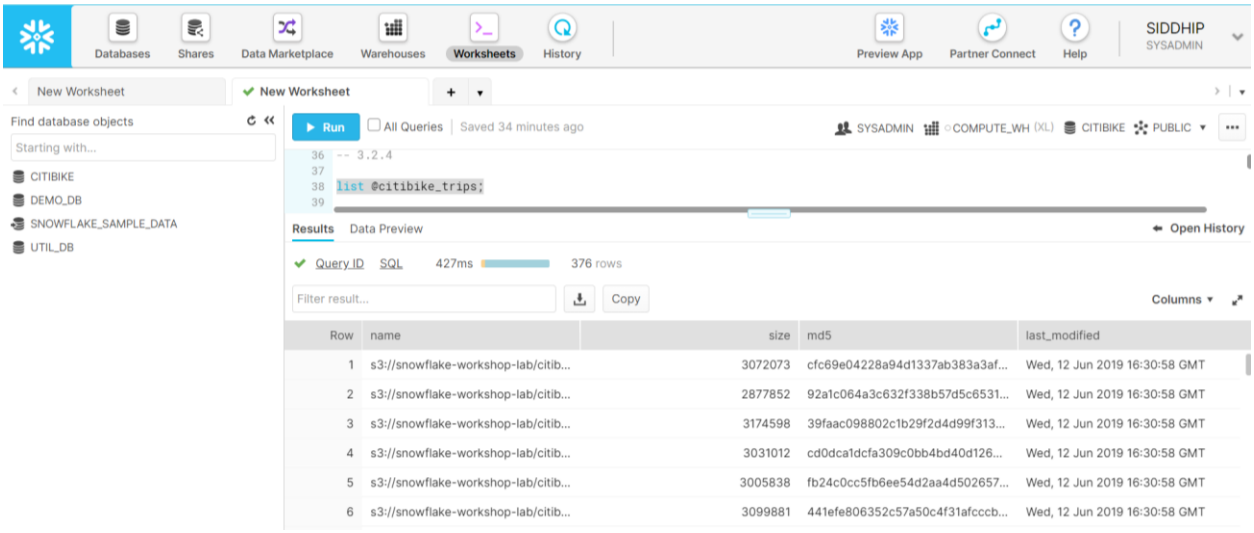
Stage created



View contents of the stage



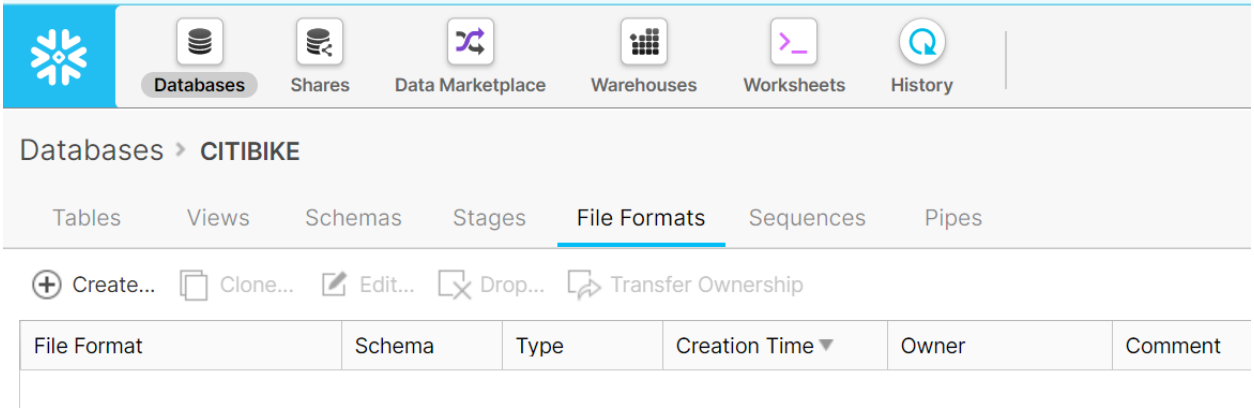


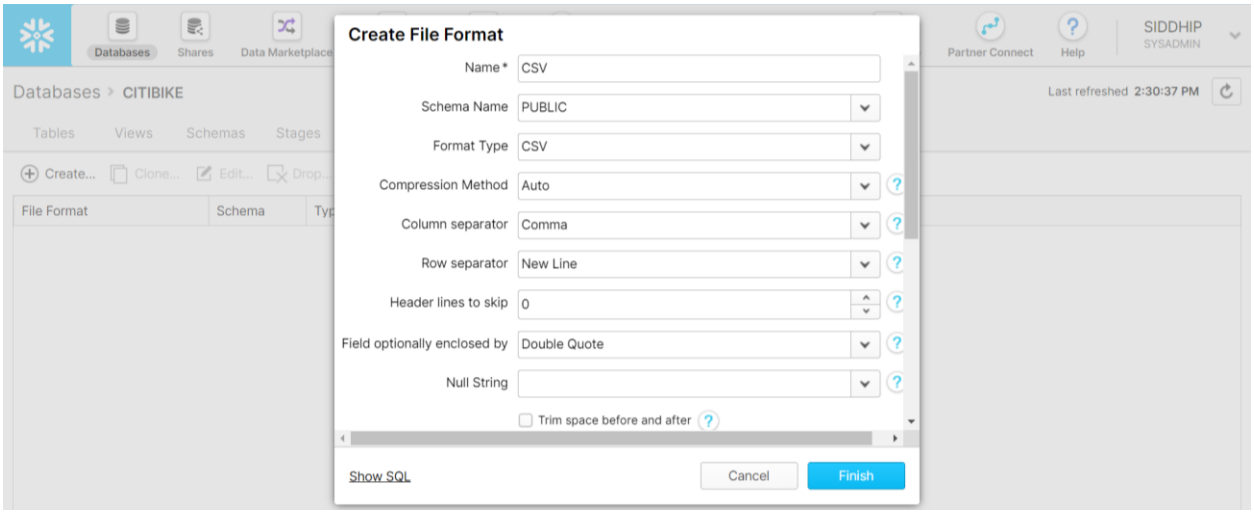

Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Preview App
Partner Connect
Help
SIDDHIP
SYSADMIN
New Worksheet
New Worksheet
Find database objects
Starting with...
CITIBIKE
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
UTIL_DB
Run
All Queries
Saved 34 minutes ago
SYSADMIN
COMPUTE_WH (XL)
CITIBIKE
PUBLIC
36 -- 3.2.4
37
38 list @citibike_trips;
39
Results
Data Preview
Open History
Query ID
SQL
427ms
376 rows
Filter result...
Copy
Columns
Row
name
size
md5
last_modified
1 s3://snowflake-workshop-lab/citib... 3072073 cfc69e04228a94d1337ab383a3af... Wed, 12 Jun 2019 16:30:58 GMT
2 s3://snowflake-workshop-lab/citib... 2877852 92a1c064a3c632f338b57d5c6531... Wed, 12 Jun 2019 16:30:58 GMT
3 s3://snowflake-workshop-lab/citib... 3174598 39faac098802c1b29f2d4d99f313... Wed, 12 Jun 2019 16:30:58 GMT
4 s3://snowflake-workshop-lab/citib... 3031012 cd0dca1dcfa309c0bb4bd40d126... Wed, 12 Jun 2019 16:30:58 GMT
5 s3://snowflake-workshop-lab/citib... 3005838 fb24c0cc5fb6ee54d2aa4d502657... Wed, 12 Jun 2019 16:30:58 GMT
6 s3://snowflake-workshop-lab/citib... 3099881 441efe806352c57a50c4f31afcccb... Wed, 12 Jun 2019 16:30:58 GMT

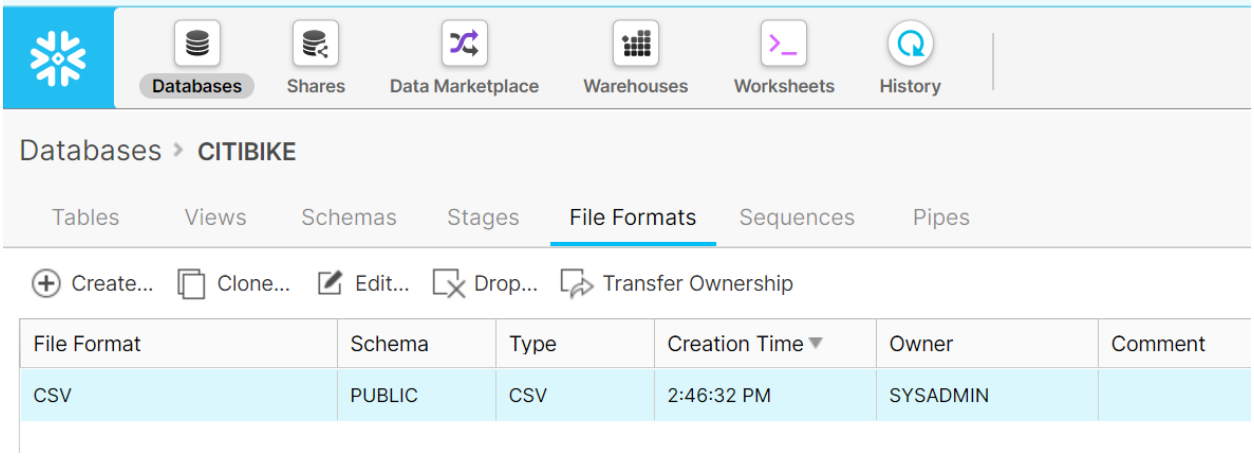
Create a File Format

In order to load data in Snowflake database a file format is created that matches the data structure.

Go to database tab -> click on ‘CITIBIKE’ database -> Select file format and Create -> Enter the required details to create the file format -> Click on Finish.







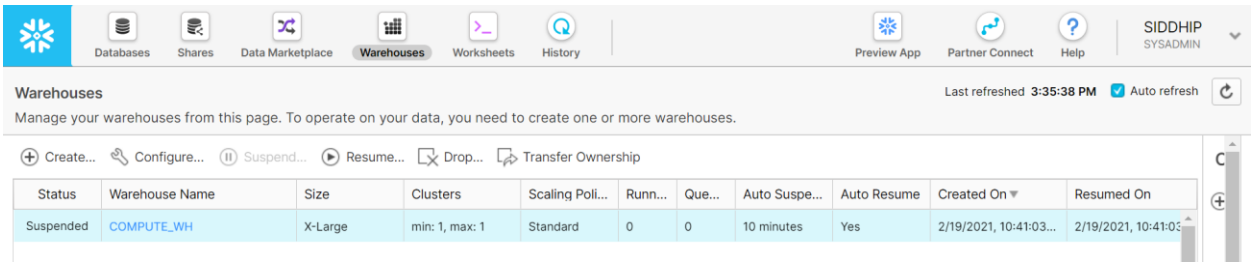
Loading Data

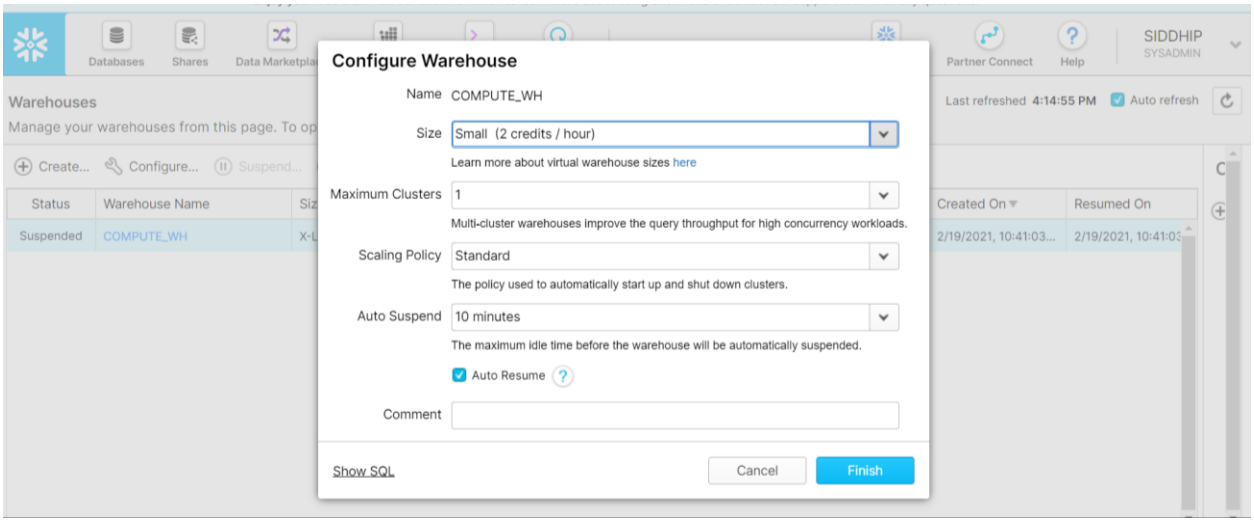
Using datawarehouse and COPY command to load structured data into Snowflake table.

Resize and Use a Warehouse for Data Loading

Go to warehouse -> Configure->COMPUTE_WH

COMPUTE_WH will be used to load data from AWS S3 into Snowflake table.



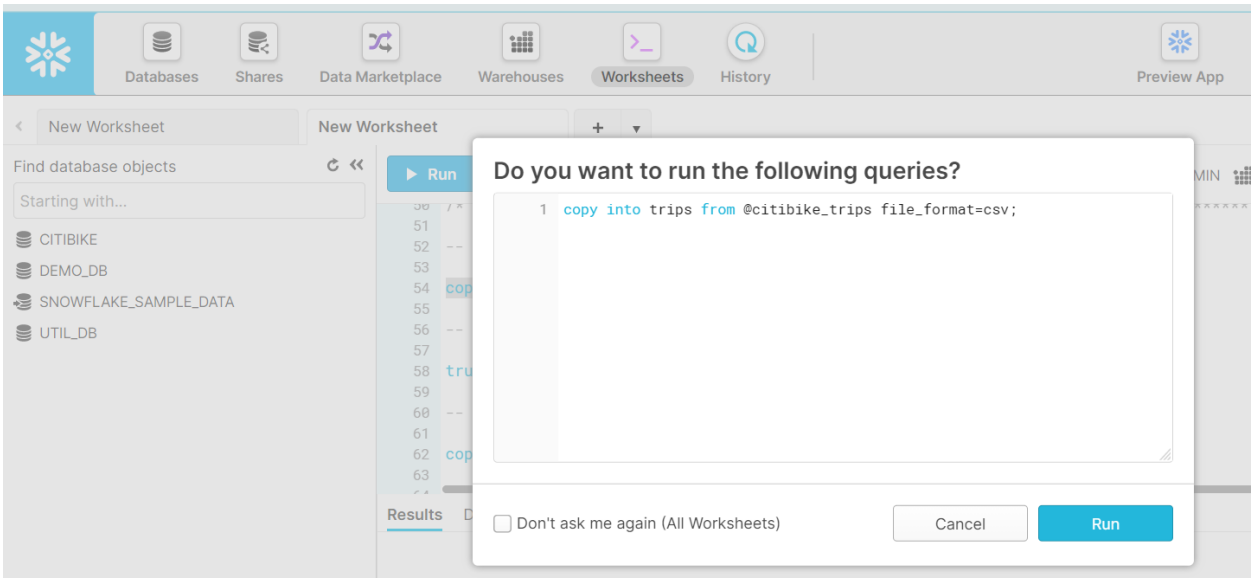


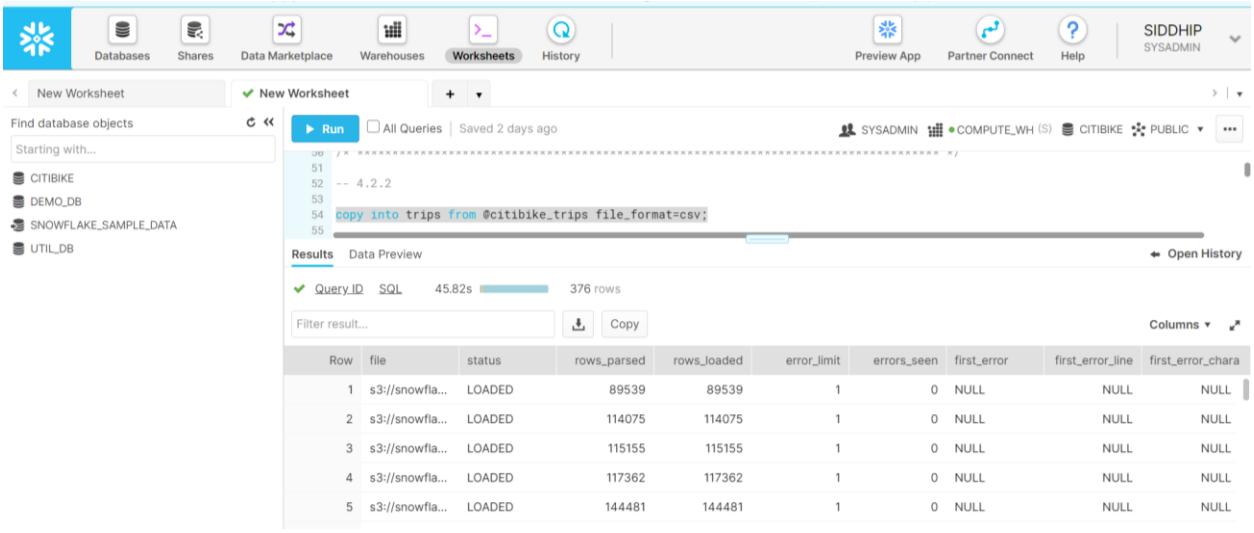
Load the Data

Copy command to load data into TRIPS table.

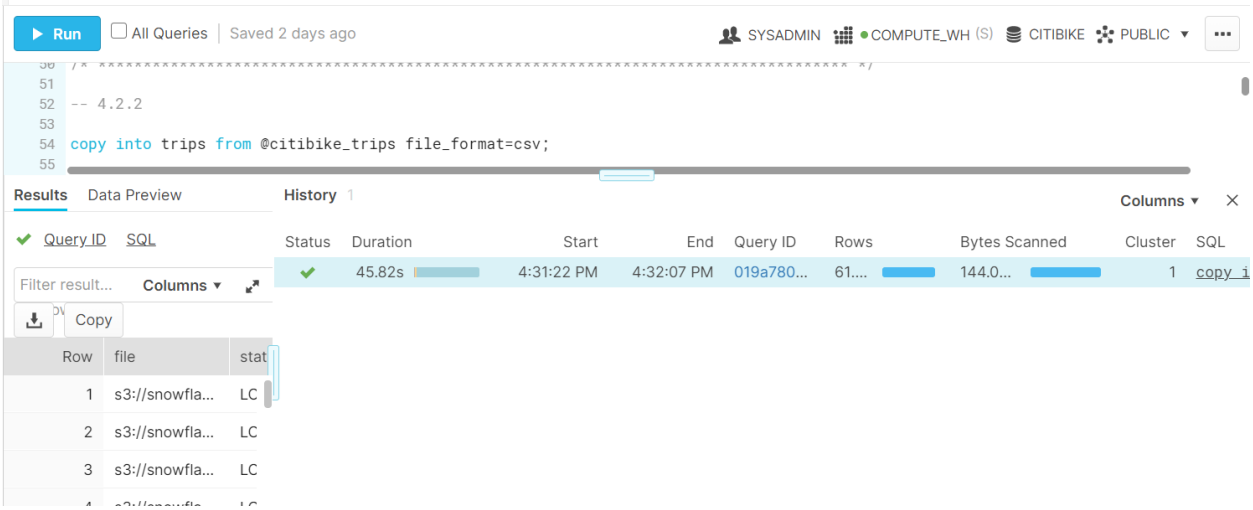
Worksheets -> check the context

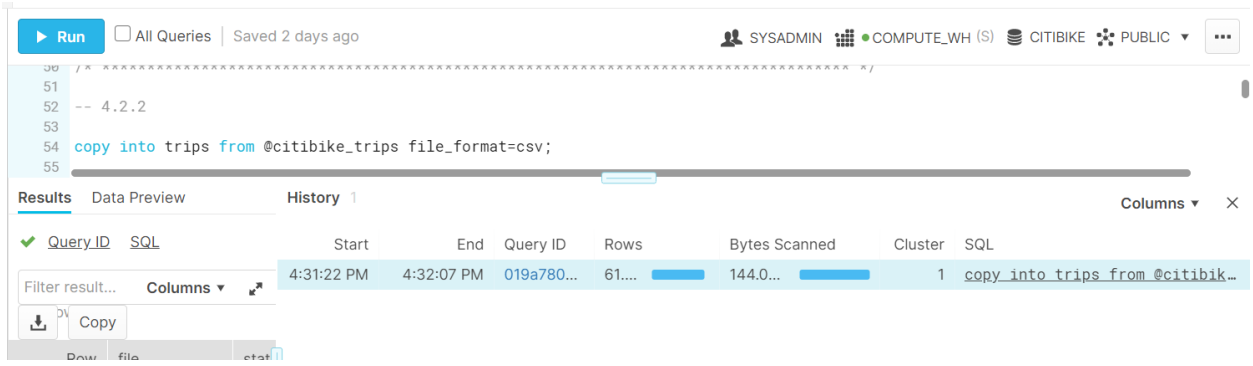
Load staged data into table in worksheet tab



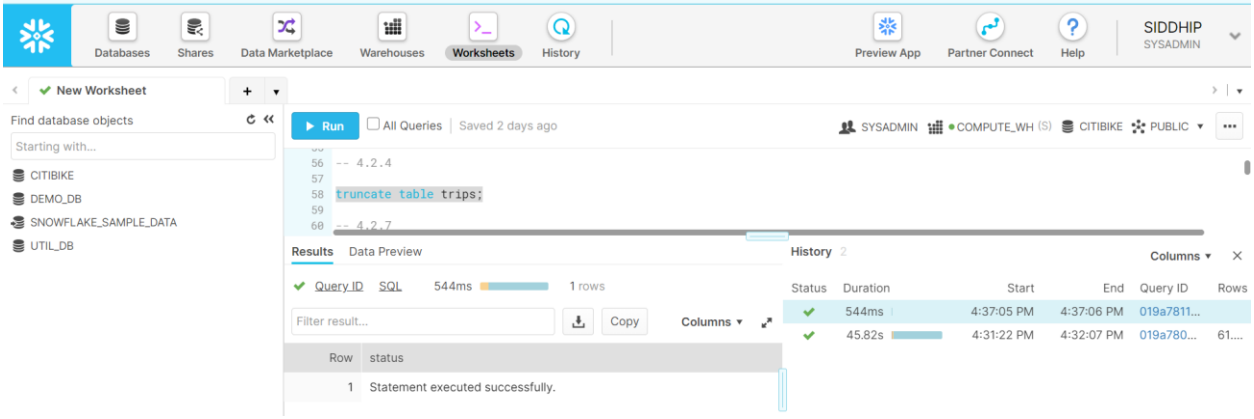


Checking history of Snowflake operations performed in that worksheet

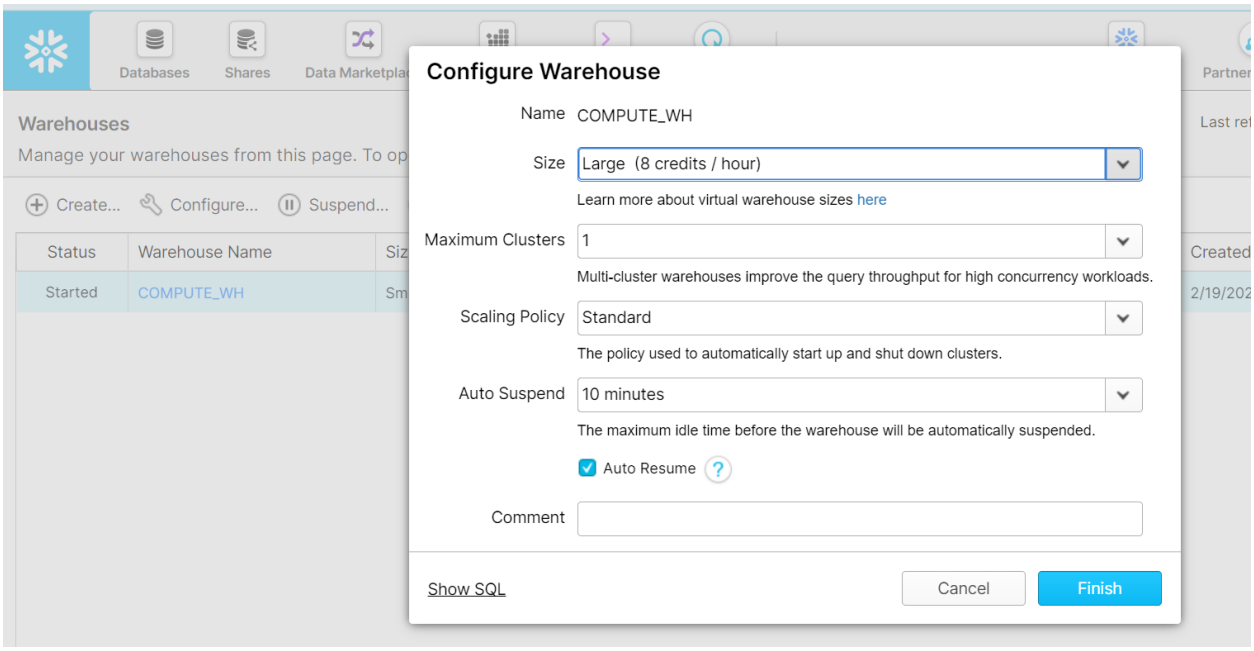




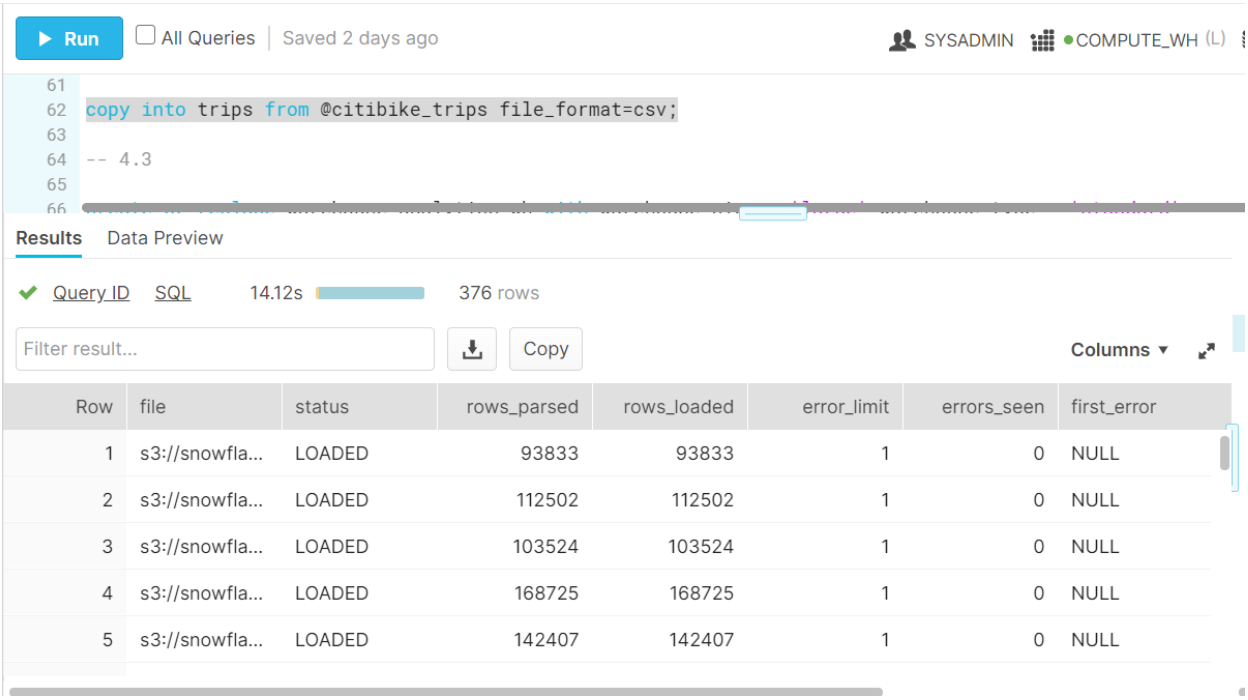
Clear the table of all data and metadata using TRUNCATE table command.



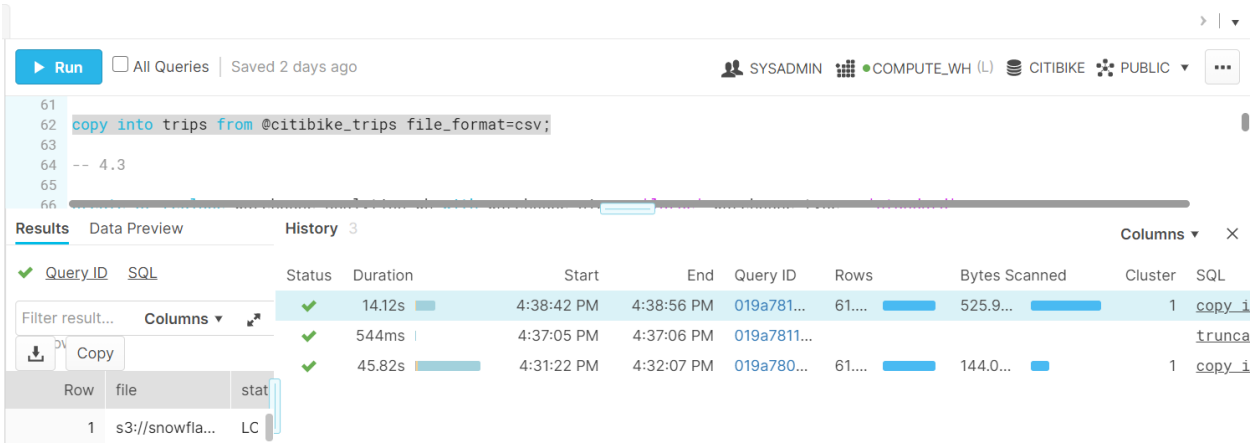
Increase the size of the warehouse to size Large.

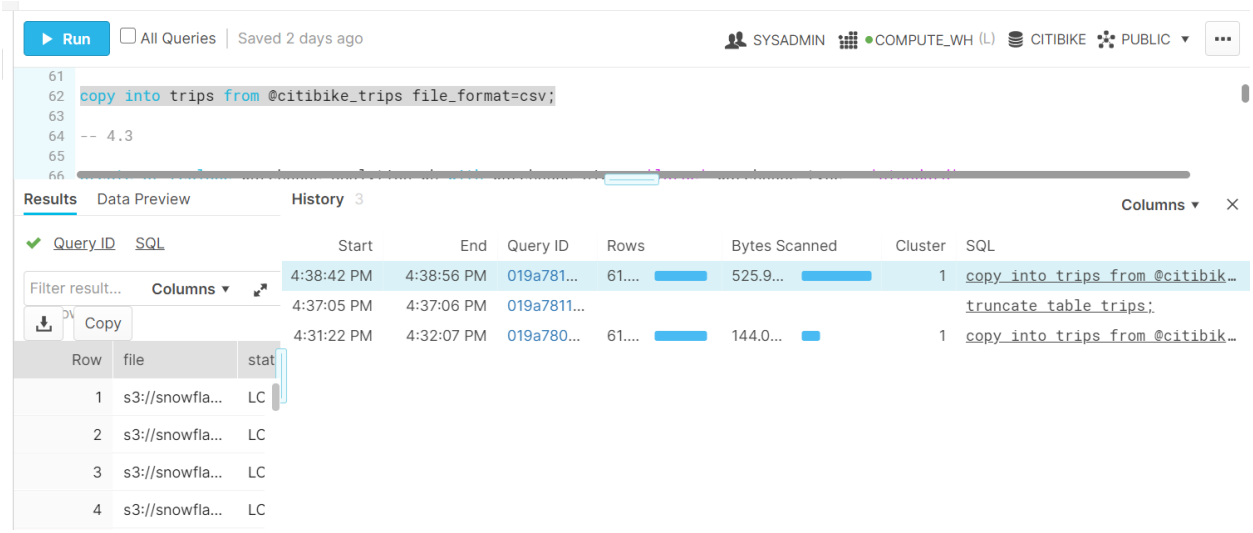


Load completed successfully in larger warehouse.



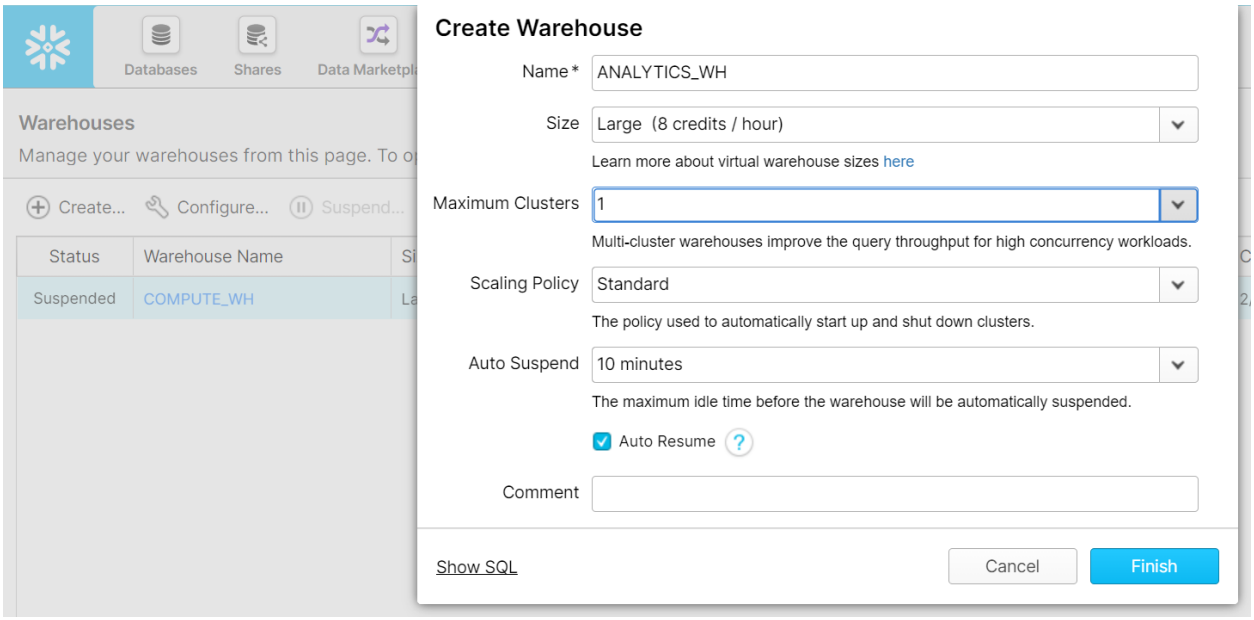
Comparing the loads using the two warehouses, load using larger warehouse was quite faster as compared to load using smaller warehouse.


Run
All Queries
Saved 2 days ago
SYSADMIN
COMPUTE_WH (L)
CITIBIKE
PUBLIC
61
62 copy into trips from @citibike_trips file_format=csv;
63
64 -- 4.3
65
66
Results
Data Preview
History 3
Columns
Query ID
SQL
Filter result...
Columns
Copy
Row
file
stat
1
s3://snowfla...
LC
Status
Duration
Start
End
Query ID
Rows
Bytes Scanned
Cluster
SQL
14.12s
4:38:42 PM
4:38:56 PM
019a781...
61....
525.9...
1
copy i
544ms
4:37:05 PM
4:37:06 PM
019a7811...
trunca
45.82s
4:31:22 PM
4:32:07 PM
019a780...
61....
144.0...
1
copy i

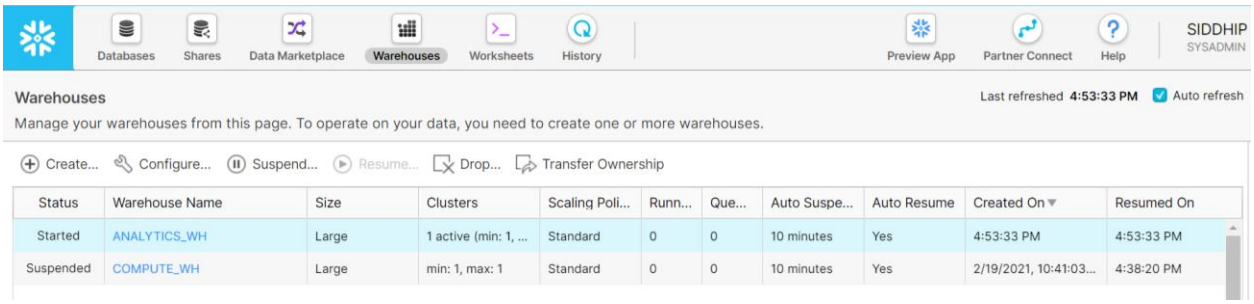

Run
All Queries
Saved 2 days ago
SYSADMIN
COMPUTE_WH (L)
CITIBIKE
PUBLIC
61
62 copy into trips from @citibike_trips file_format=csv;
63
64 -- 4.3
65
66
Results
Data Preview
History 3
Columns
Query ID
SQL
Filter result...
Columns
Copy
Row
file
stat
1
s3://snowfla...
LC
2
s3://snowfla...
LC
3
s3://snowfla...
LC
4
s3://snowfla...
LC
Start
End
Query ID
Rows
Bytes Scanned
Cluster
SQL
4:38:42 PM
4:38:56 PM
019a781...
61....
525.9...
1
copy into trips from @citibik…
4:37:05 PM
4:37:06 PM
019a7811...
truncate table trips;
4:31:22 PM
4:32:07 PM
019a780...
61....
144.0...
1
copy into trips from @citibik…

Create a New Warehouse for Data Analytics

Go to warehouse tab -> Create -> `ANALYTICS_WH` with large size -> Finish



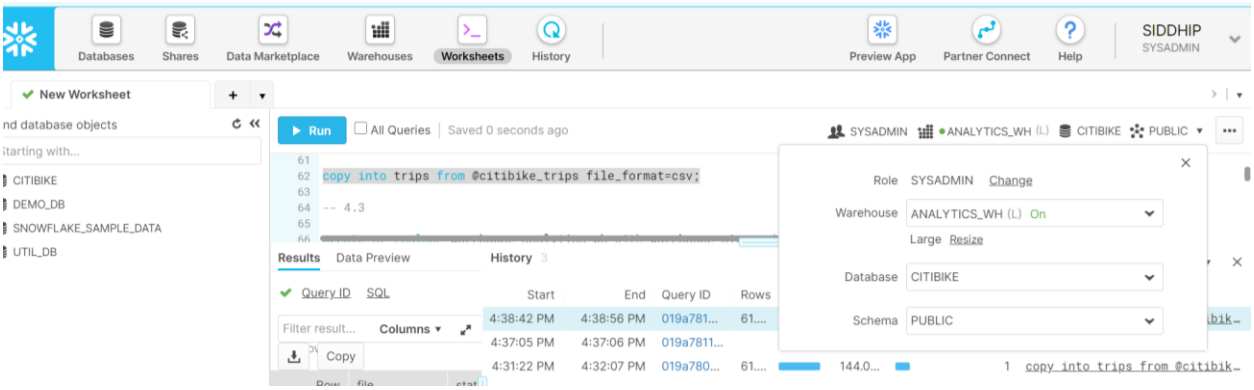
Warehouse created successfully



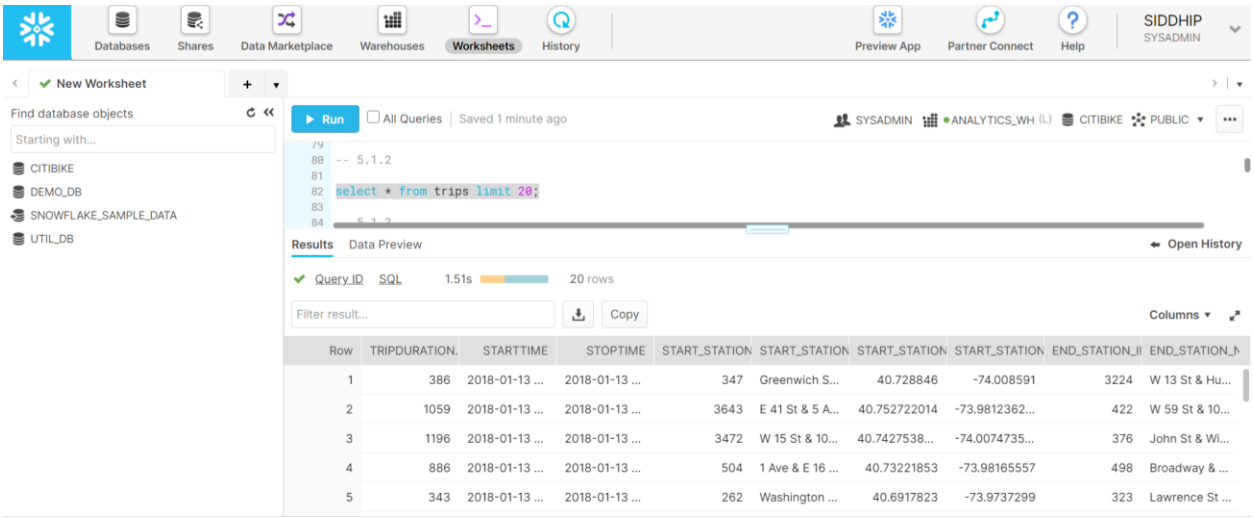
Analytical Queries, Results Cache, Cloning

Execute SELECT Statements and Result Cache

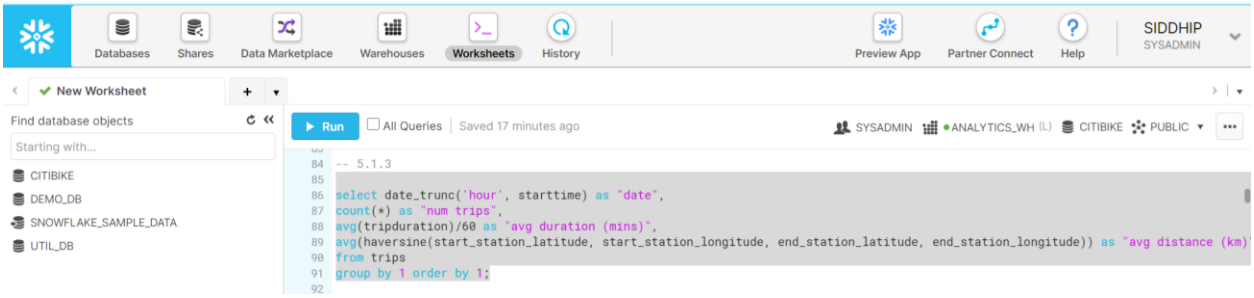
Go to Worksheet tab -> check for the context



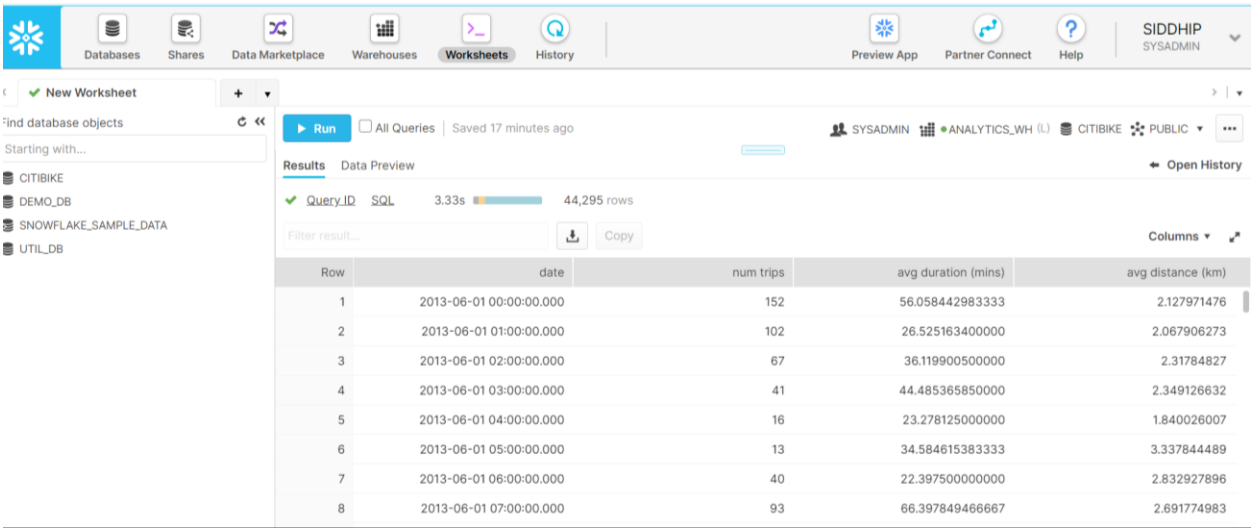
View sample of select data



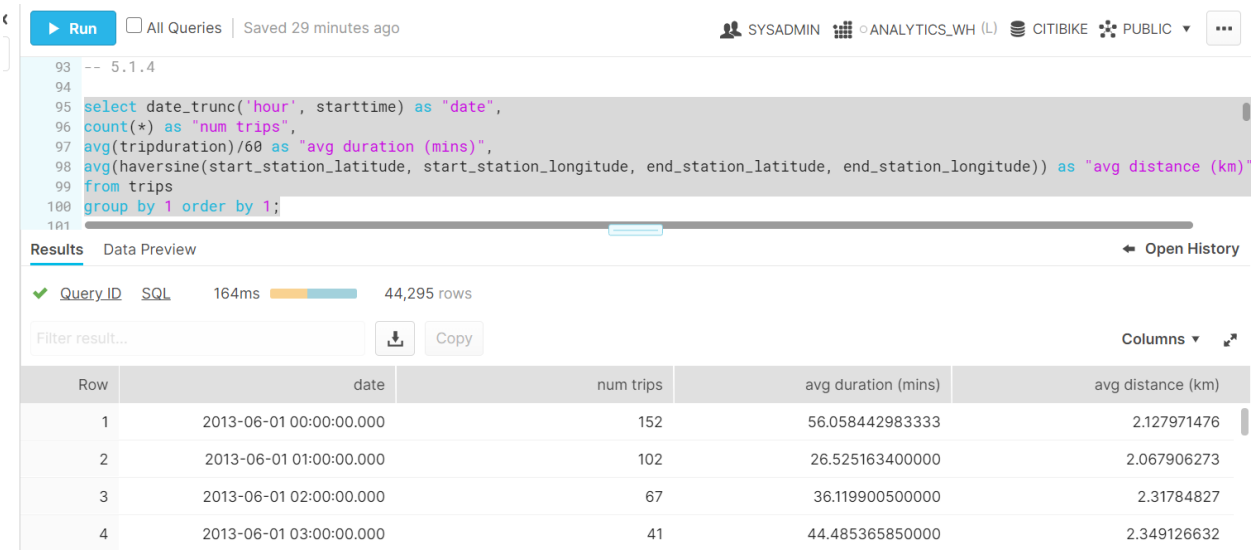
View the hourly statistics on city bike usage.



Output:-



Snowflake has a result cache that holds the results of every query executed in the past 24 hours.



```
-- 5.1.4

select date_trunc('hour', starttime) as "date",
count(*) as "num trips",
avg(tripduration)/60 as "avg duration (mins)",
avg(haversine(start_station_latitude, start_station_longitude, end_station_latitude, end_station_longitude)) as "avg distance (km)"
from trips
group by 1 order by 1;
```

Run | All Queries | Saved 29 minutes ago | SYSADMIN | ANALYTICS_WH (L) | CITIBIKE | PUBLIC

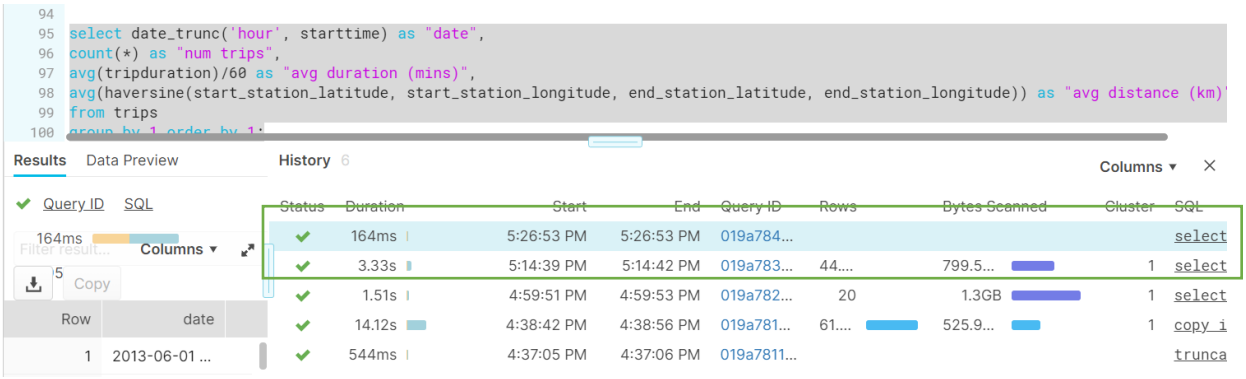
Results | Data Preview | Open History

Query ID | SQL | 164ms | 44,295 rows

Filter result... | Copy | Columns

Row	date	num trips	avg duration (mins)	avg distance (km)
1	2013-06-01 00:00:00.000	152	56.058442983333	2.127971476
2	2013-06-01 01:00:00.000	102	26.525163400000	2.067906273
3	2013-06-01 02:00:00.000	67	36.119900500000	2.31784827
4	2013-06-01 03:00:00.000	41	44.485365850000	2.349126632

In the History window note that the second query runs significantly faster because the results have been cached.



```
select date_trunc('hour', starttime) as "date",
count(*) as "num trips",
avg(tripduration)/60 as "avg duration (mins)",
avg(haversine(start_station_latitude, start_station_longitude, end_station_latitude, end_station_longitude)) as "avg distance (km)"
from trips
group by 1 order by 1;
```

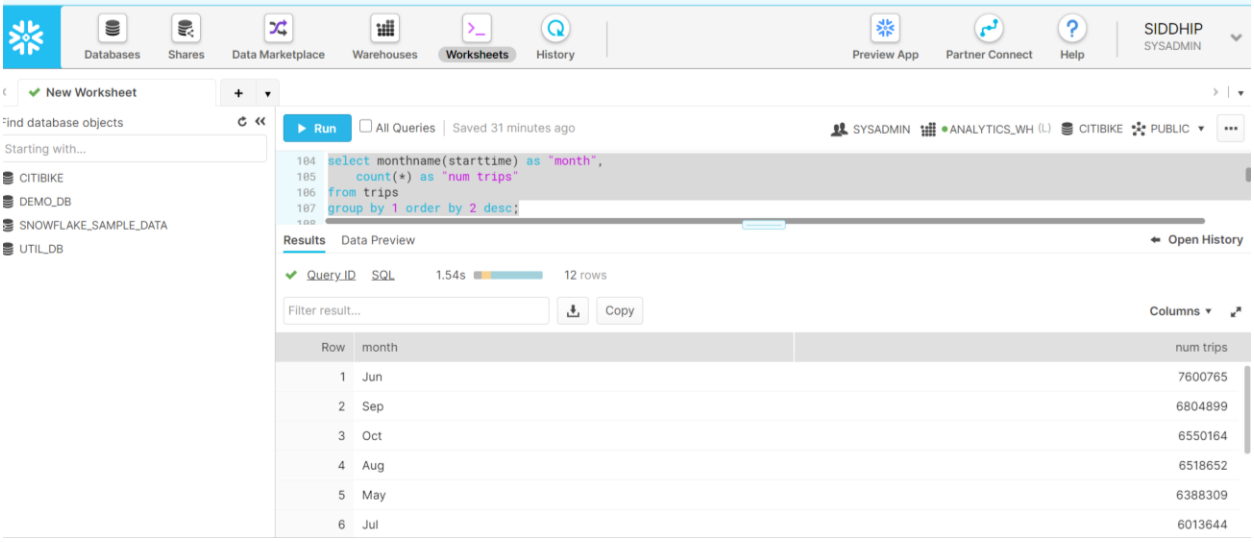
Results | Data Preview | History 6 | Columns

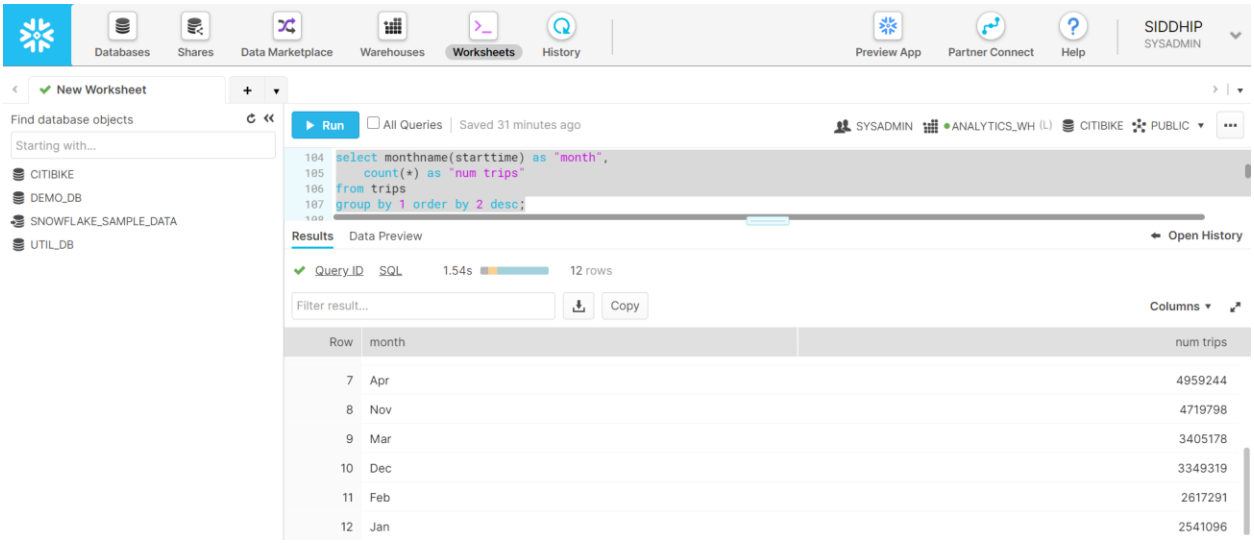
Query ID | SQL | 164ms | Columns | Copy

Row	date
1	2013-06-01 ...

Status	Duration	Start	End	Query ID	Rows	Bytes Scanned	Cluster	SQL
✓	164ms	5:26:53 PM	5:26:53 PM	019a784...				select
✓	3.33s	5:14:39 PM	5:14:42 PM	019a783...	44....	799.5...	1	select
✓	1.51s	4:59:51 PM	4:59:53 PM	019a782...	20	1.3GB	1	select
✓	14.12s	4:38:42 PM	4:38:56 PM	019a781...	61....	525.9...	1	copy i
✓	544ms	4:37:05 PM	4:37:06 PM	019a7811...				trunca

Verify which months are the busiest :-


Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Preview App
Partner Connect
Help
SIDDHIP
SYSADMIN
New Worksheet
Find database objects
Starting with...
CITIBIKE
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
UTIL_DB
Run
All Queries
Saved 31 minutes ago
SYSADMIN
ANALYTICS_WH (L)
CITIBIKE
PUBLIC
104 select monthname(starttime) as "month",
105 count(*) as "num trips"
106 from trips
107 group by 1 order by 2 desc;
Results
Data Preview
Open History
Query ID
SQL
1.54s
12 rows
Filter result...
Copy
Columns
Row
month
num trips
1 Jun 7600765
2 Sep 6804899
3 Oct 6550164
4 Aug 6518652
5 May 6388309
6 Jul 6013644

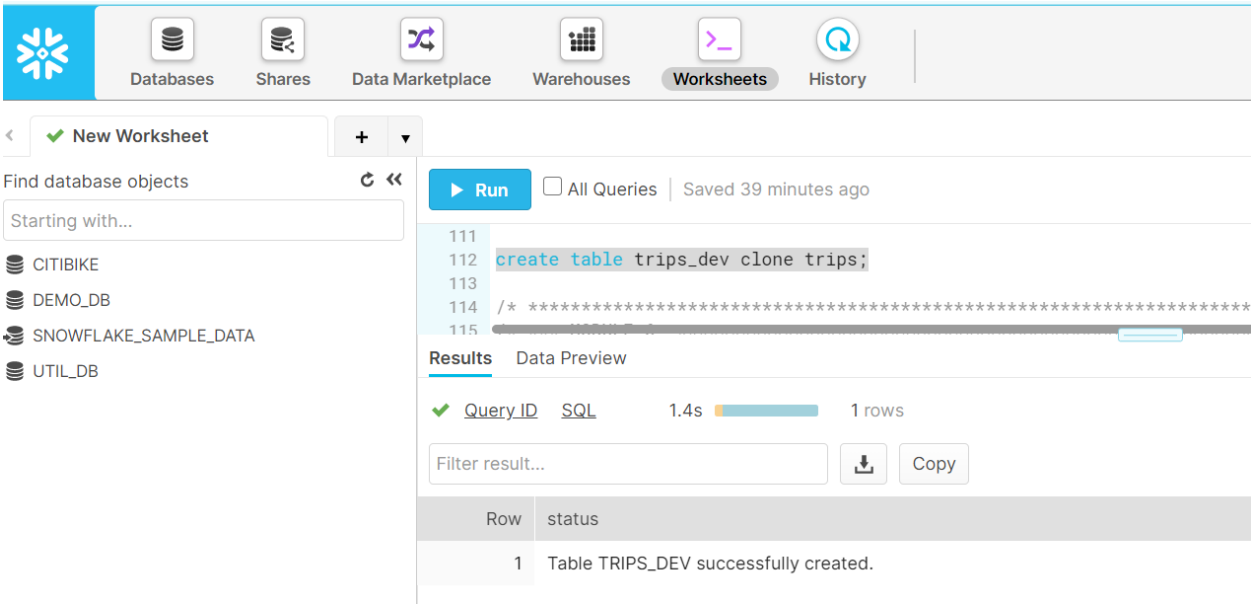

Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Preview App
Partner Connect
Help
SIDDHIP
SYSADMIN
New Worksheet
Find database objects
Starting with...
CITIBIKE
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
UTIL_DB
Run
All Queries
Saved 31 minutes ago
SYSADMIN
ANALYTICS_WH (L)
CITIBIKE
PUBLIC
104 select monthname(starttime) as "month",
105 count(*) as "num trips"
106 from trips
107 group by 1 order by 2 desc;
Results
Data Preview
Open History
Query ID
SQL
1.54s
12 rows
Filter result...
Copy
Columns
Row
month
num trips
7 Apr 4959244
8 Nov 4719798
9 Mar 3405178
10 Dec 3349319
11 Feb 2617291
12 Jan 2541096

Clone a Table

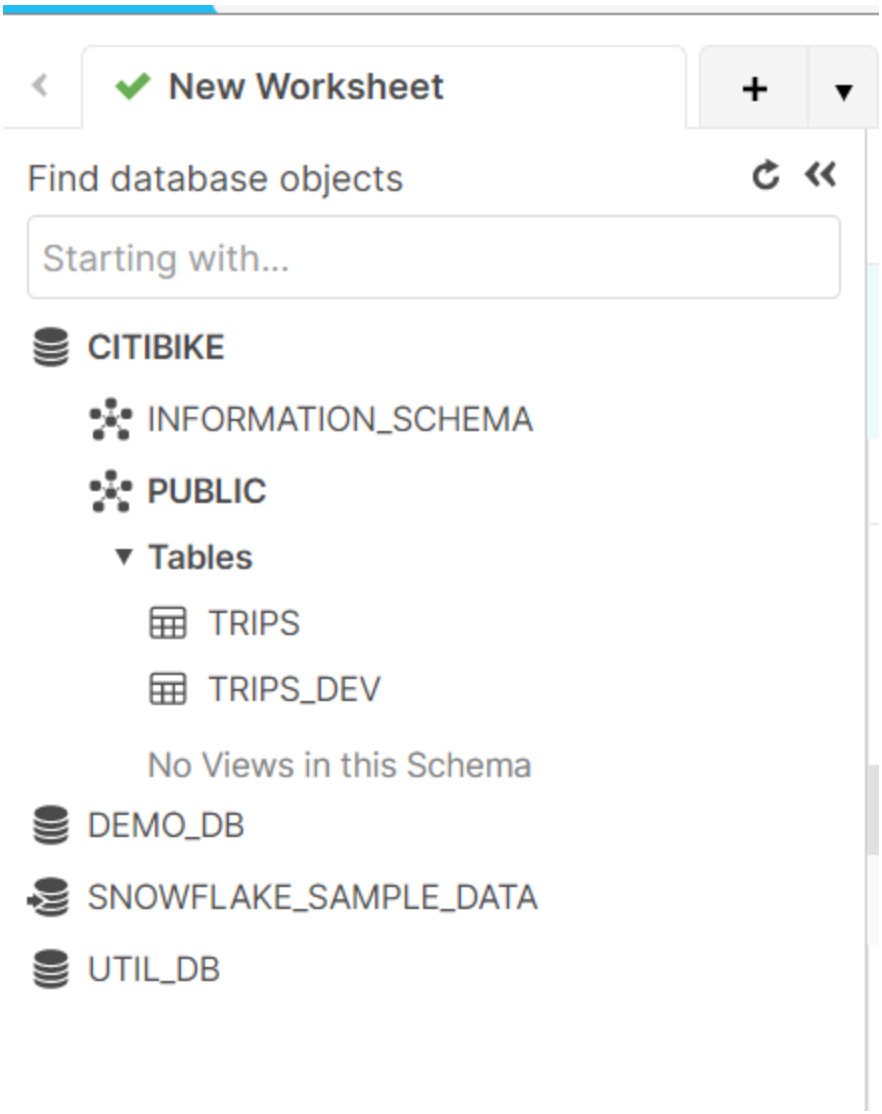
A snapshot of data present in the source object is taken when the clone is created, and is made available to the cloned object.

The cloned object is writable and independent of the clone source.

Creating a development environment so that we can test the data without aking ay changes to prod data and prod data can remain secure .



New cloned table is present under CITIBIKE database



Working with Semi-Structured Data, Views, JOIN

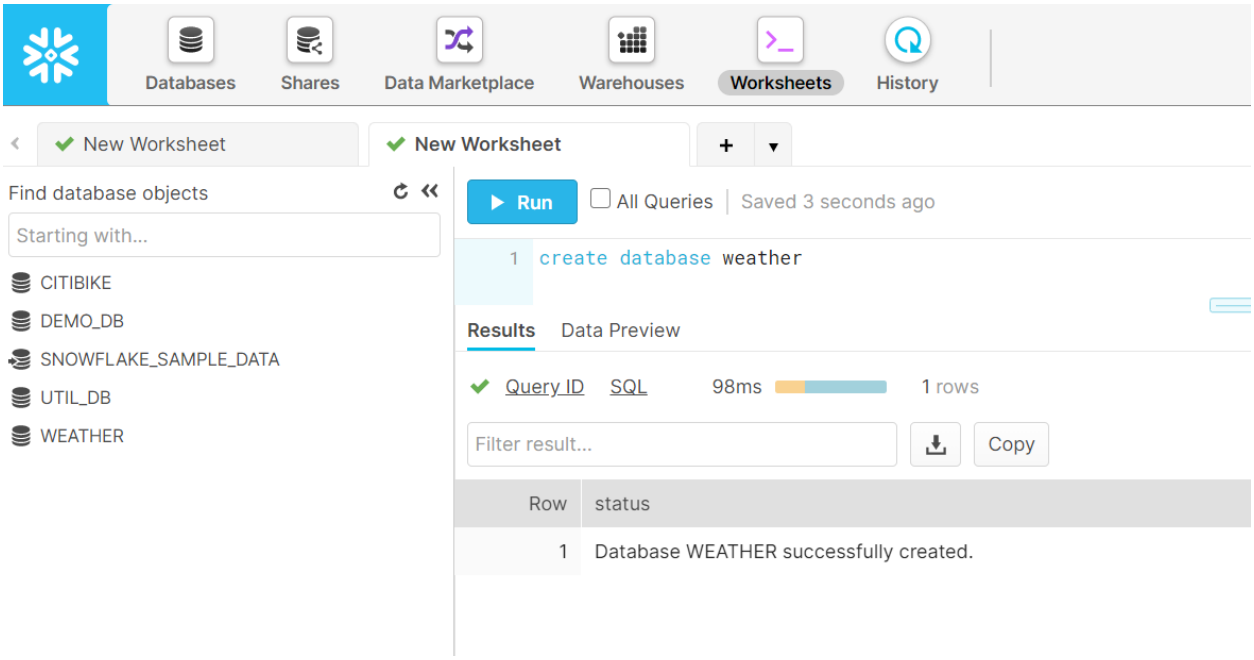
The semi structured weather data is staged in AWS S3. Citi bike wants to analyze the impact of weather on the bike rides.

The following 4 steps need to be followed :-

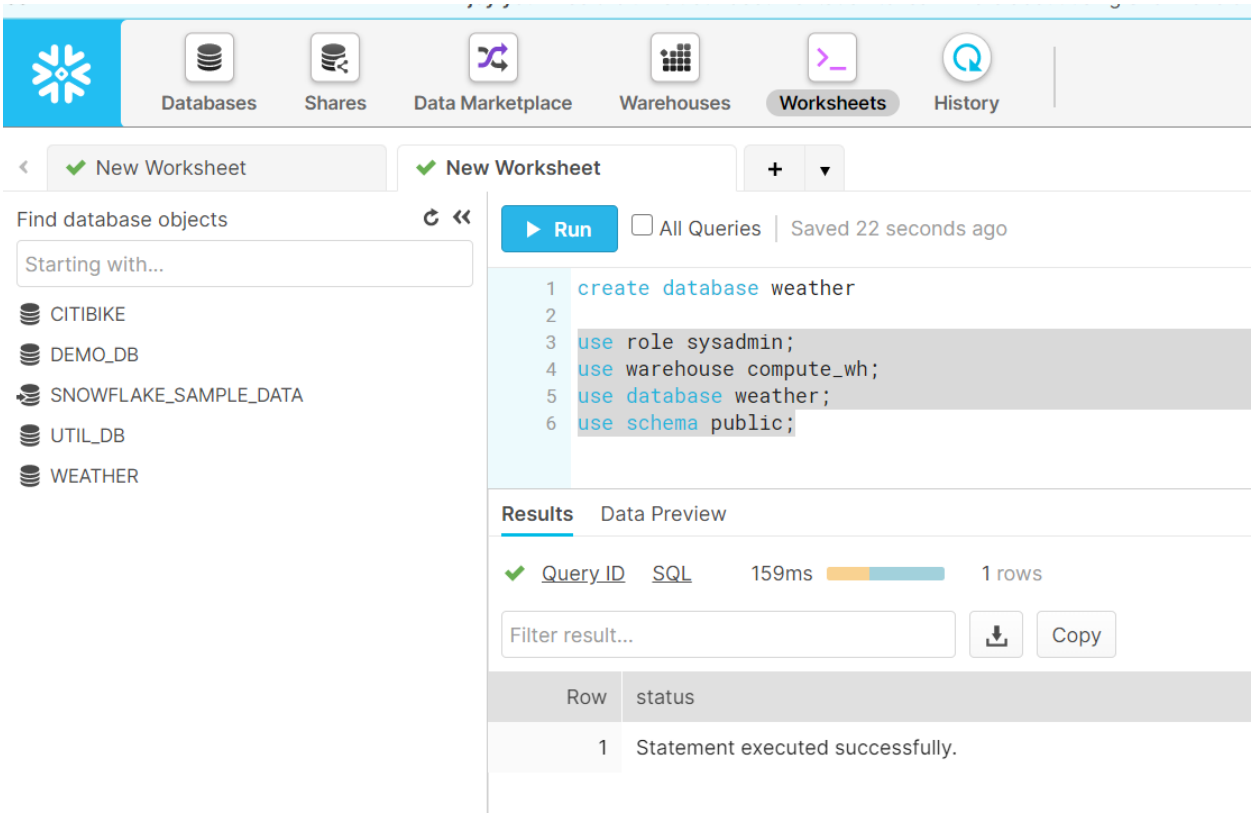
- Load weather data staged in AWS S3 bucket in json format
- Query the data and create view using SQL Dot notation
- Create join on the weather data and trips data
- Analyze the weather and ride count data to determine their relationship

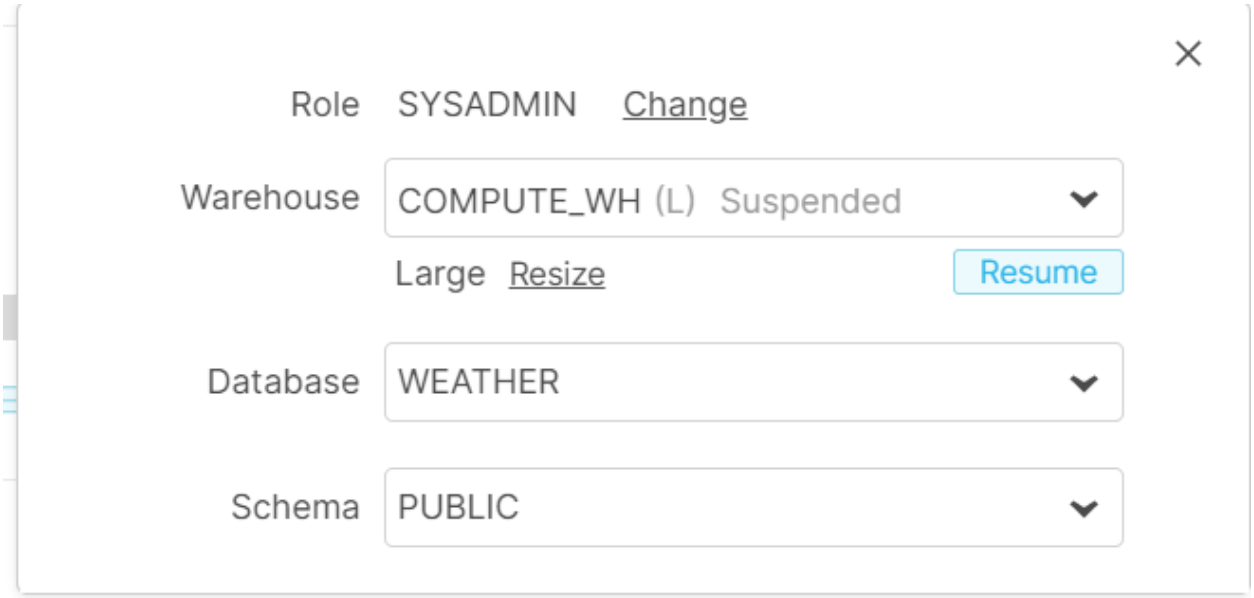
Create a Database and Table

Create weather database to store the semi structured data

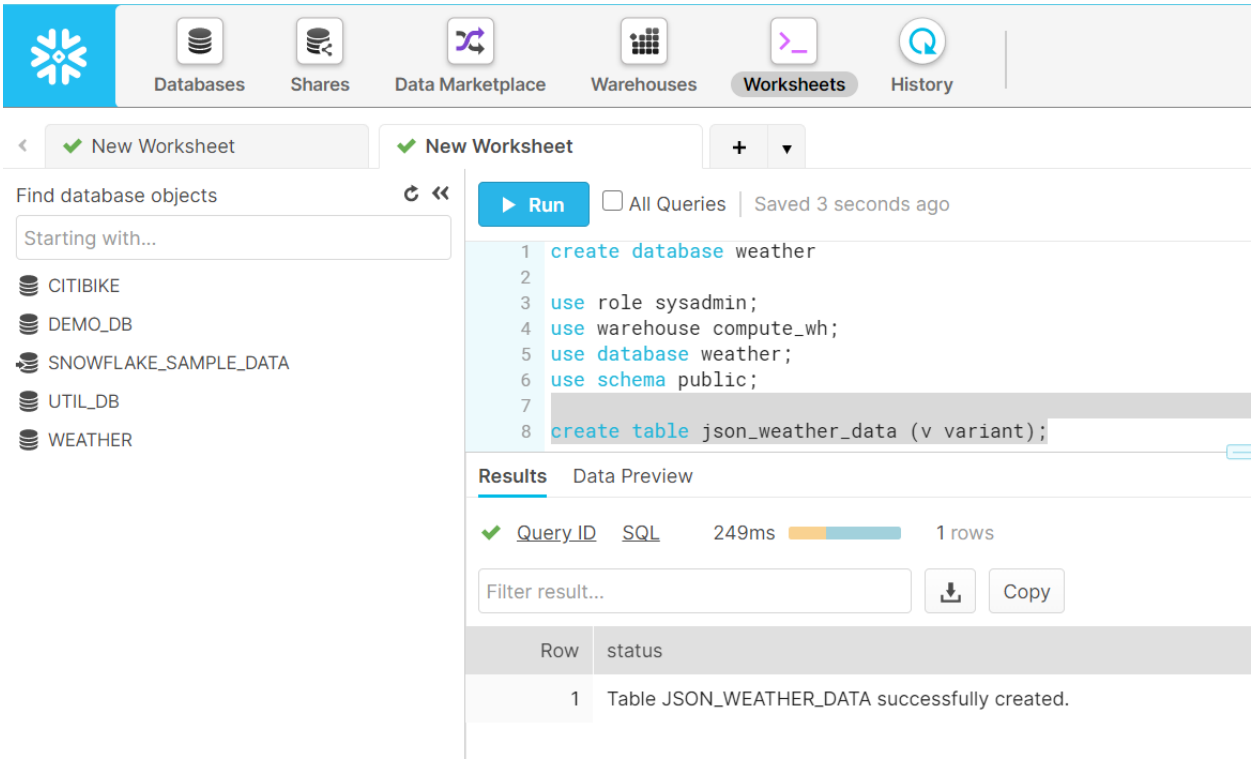


Setting the context

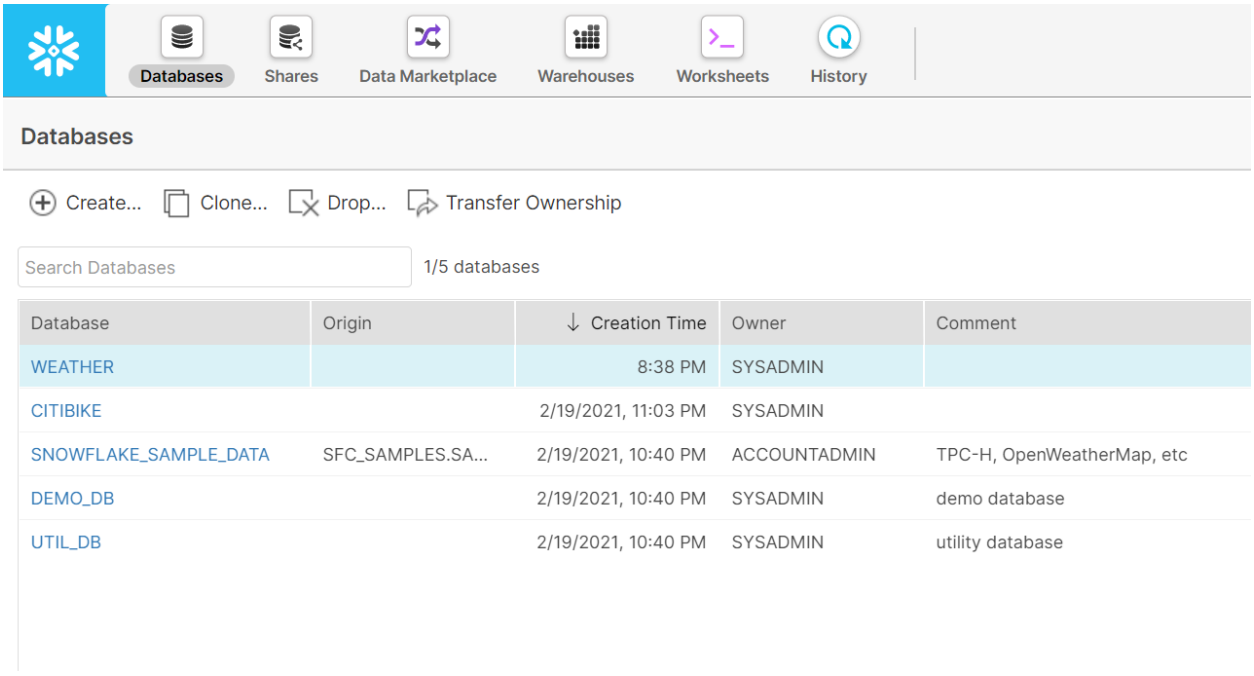




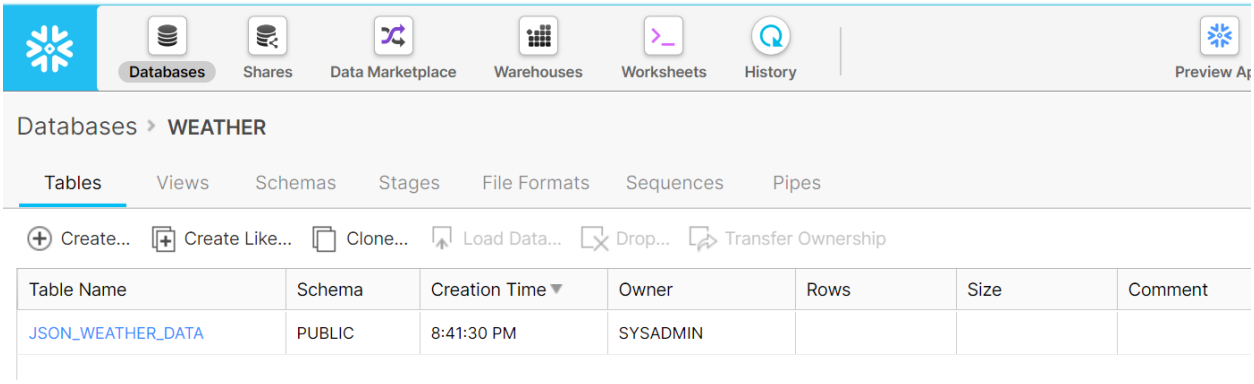
Creating table to store the JSON data which will be stored in Snowflake's variant column type.



Go to Databases -> Click on WEATHER link

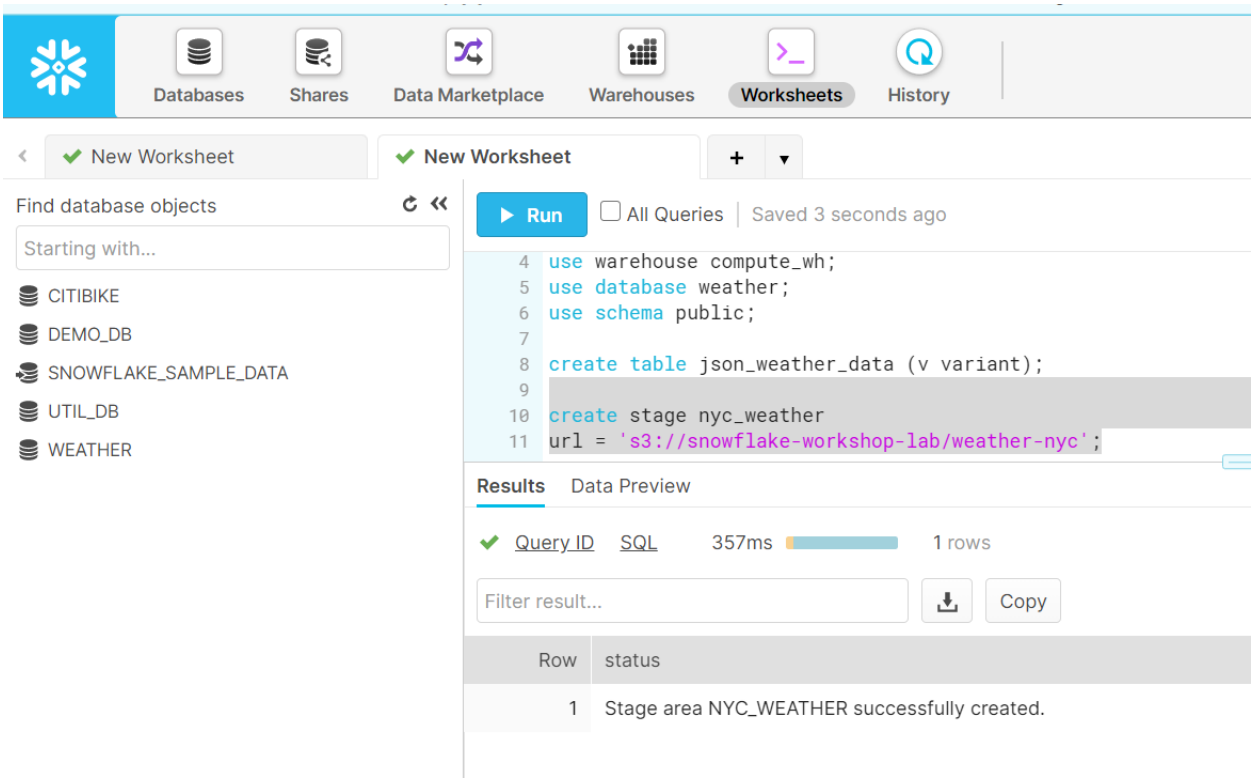


View the newly created table

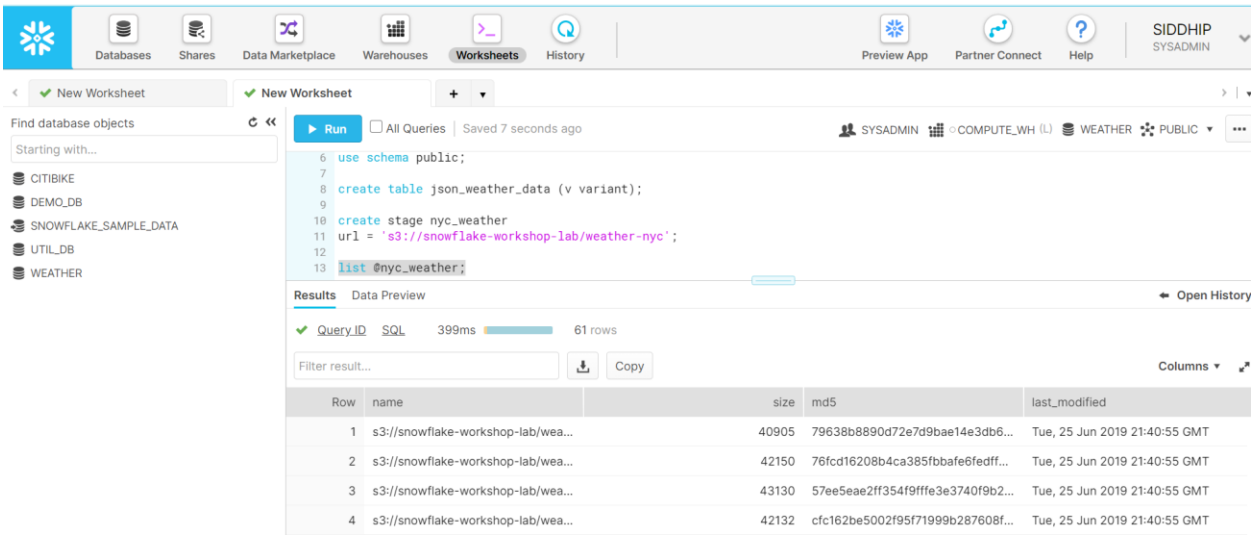


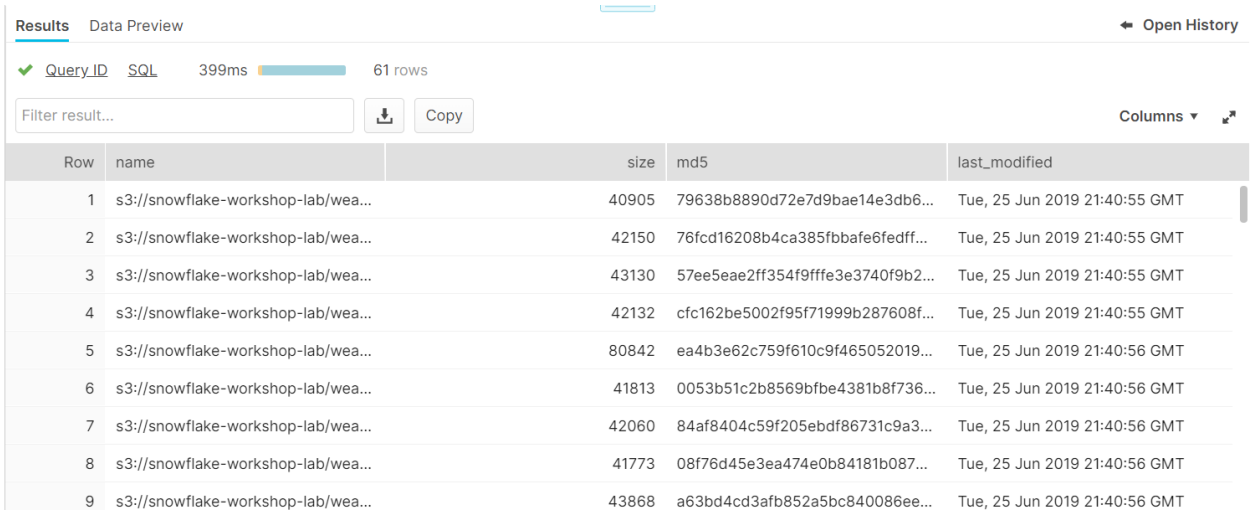
Create an External Stage

Go to worksheet tab -> create a stage from where the unstructured data is stored on AWS S3



Viewing the contents of the nyc_weather stage





Results | Data Preview | Open History

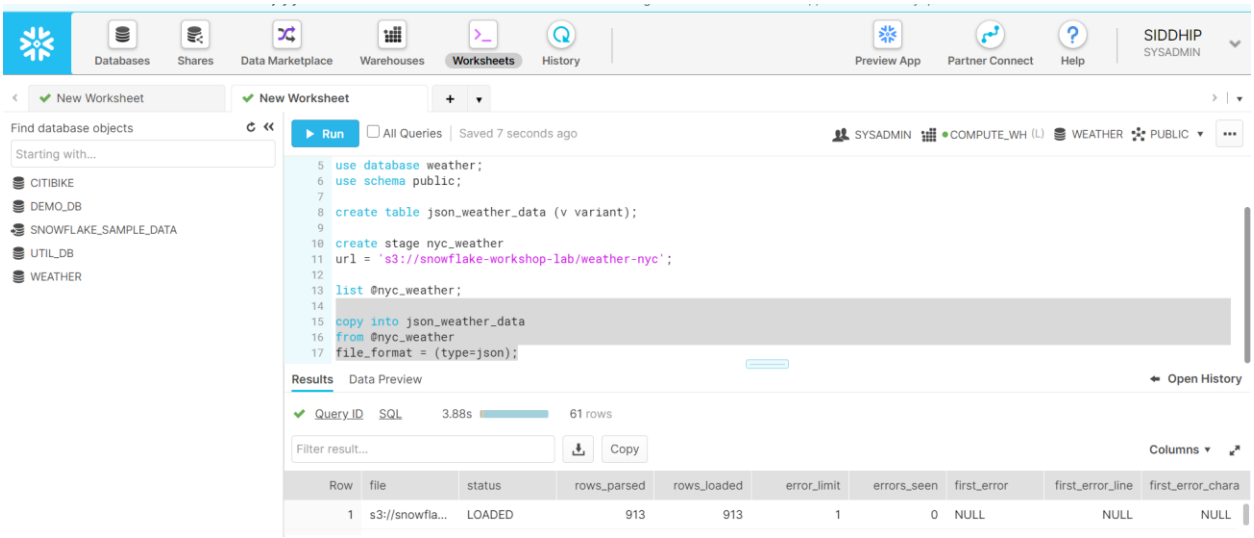
Query ID | SQL | 399ms | 61 rows

Filter result... | Copy | Columns

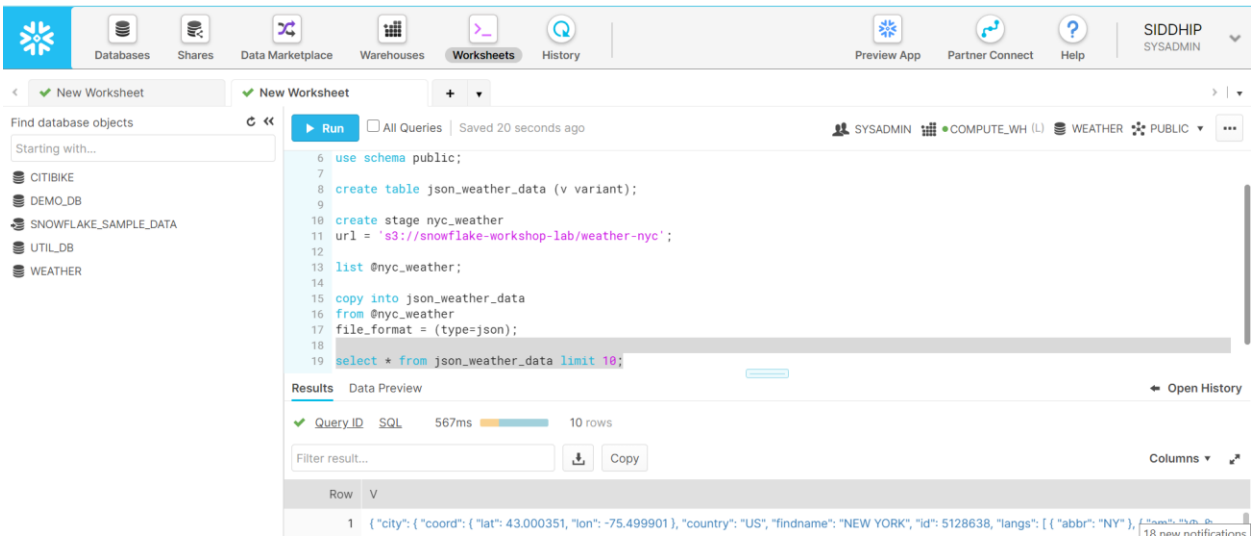
Row	name	size	md5	last_modified
1	s3://snowflake-workshop-lab/wea...	40905	79638b8890d72e7d9bae14e3db6...	Tue, 25 Jun 2019 21:40:55 GMT
2	s3://snowflake-workshop-lab/wea...	42150	76fcd16208b4ca385fbbafe6fedff...	Tue, 25 Jun 2019 21:40:55 GMT
3	s3://snowflake-workshop-lab/wea...	43130	57ee5eae2ff354f9fffe3e3740f9b2...	Tue, 25 Jun 2019 21:40:55 GMT
4	s3://snowflake-workshop-lab/wea...	42132	cfc162be5002f95f71999b287608f...	Tue, 25 Jun 2019 21:40:55 GMT
5	s3://snowflake-workshop-lab/wea...	80842	ea4b3e62c759f610c9f465052019...	Tue, 25 Jun 2019 21:40:56 GMT
6	s3://snowflake-workshop-lab/wea...	41813	0053b51c2b8569bfbe4381b8f736...	Tue, 25 Jun 2019 21:40:56 GMT
7	s3://snowflake-workshop-lab/wea...	42060	84af8404c59f205ebdf86731c9a3...	Tue, 25 Jun 2019 21:40:56 GMT
8	s3://snowflake-workshop-lab/wea...	41773	08f76d45e3ea474e0b84181b087...	Tue, 25 Jun 2019 21:40:56 GMT
9	s3://snowflake-workshop-lab/wea...	43868	a63bd4cd3afb852a5bc840086ee...	Tue, 25 Jun 2019 21:40:56 GMT

Loading and Verifying the Unstructured Data

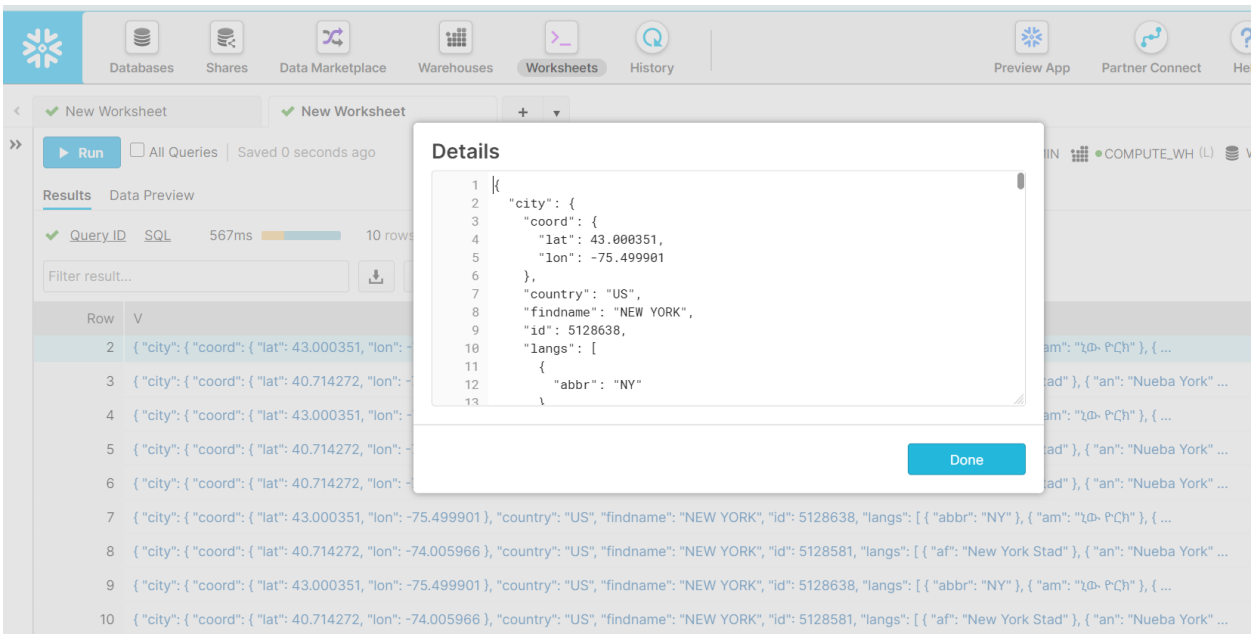
Using warehouse loading data into snowflake table from s3 bucket



Viewing the data that has been loaded

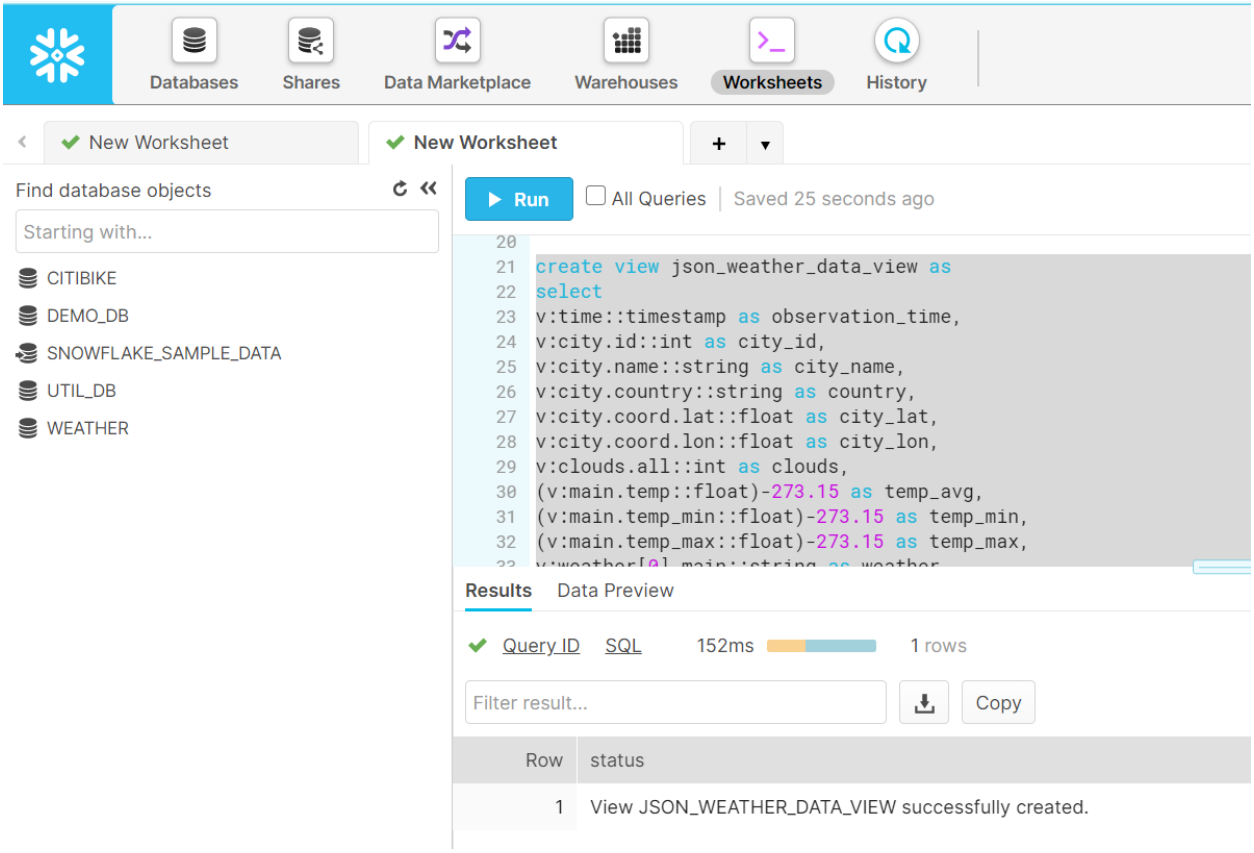


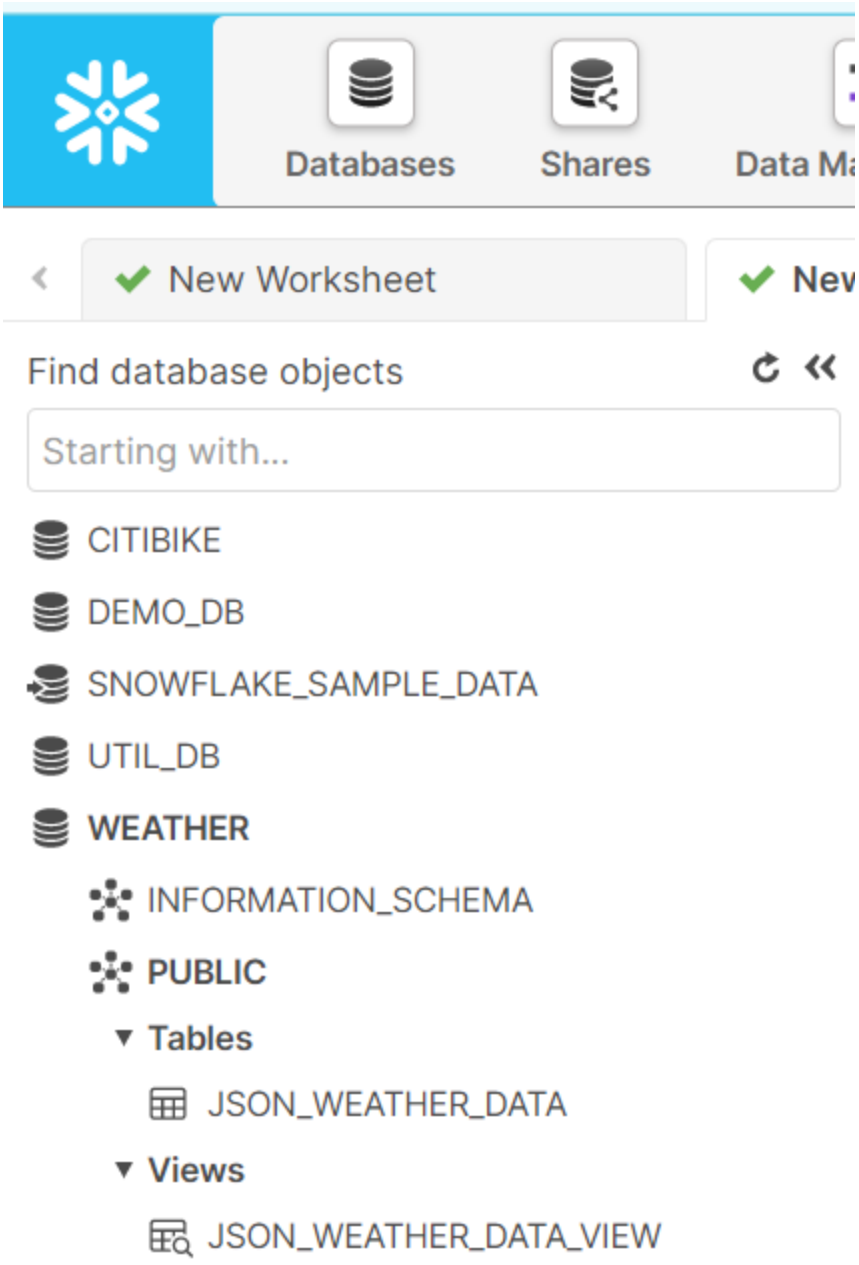
Clicking on one of the data -> view data stored in raw json format

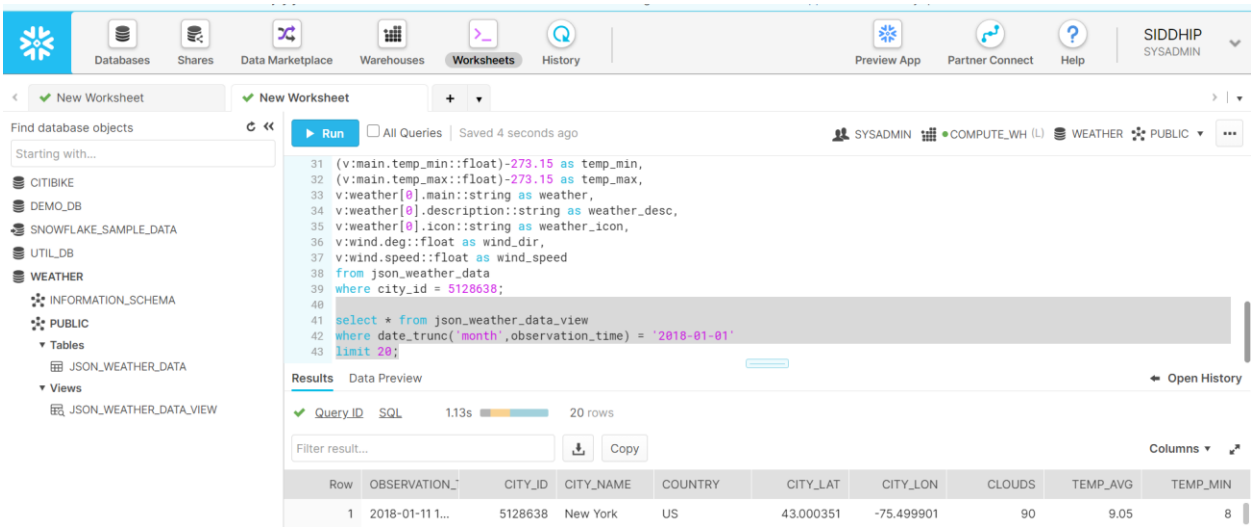

Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Preview App
Partner Connect
New Worksheet
New Worksheet
Run
All Queries
Saved 0 seconds ago
COMPUTE_WH (L)
Results
Data Preview
Query ID
SQL
567ms
10 rows
Filter result...
Row
V
Details
{
"city": {
"coord": {
"lat": 43.000351,
"lon": -75.499901
},
"country": "US",
"findname": "NEW YORK",
"id": 5128638,
"langs": [
{
"abbr": "NY"
Done

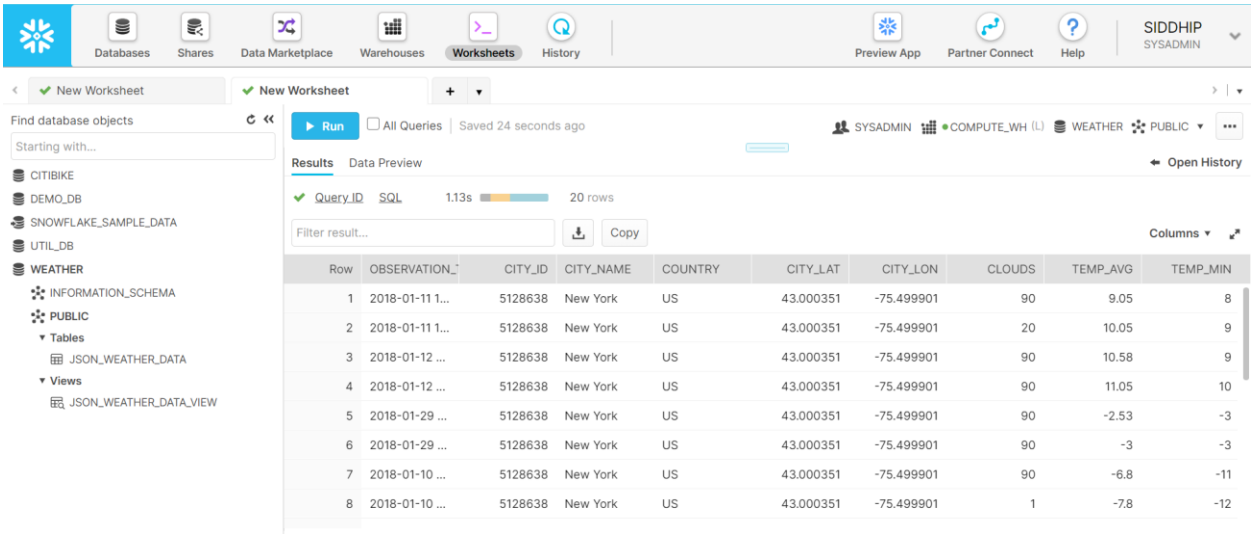
Create a View and Query Semi-Structured Data

SQL dot notation **v.city.coord.lat** is used in this command to pull out values at lower levels within the JSON hierarchy. This allows us to treat each field as if it were a column in a relational table.



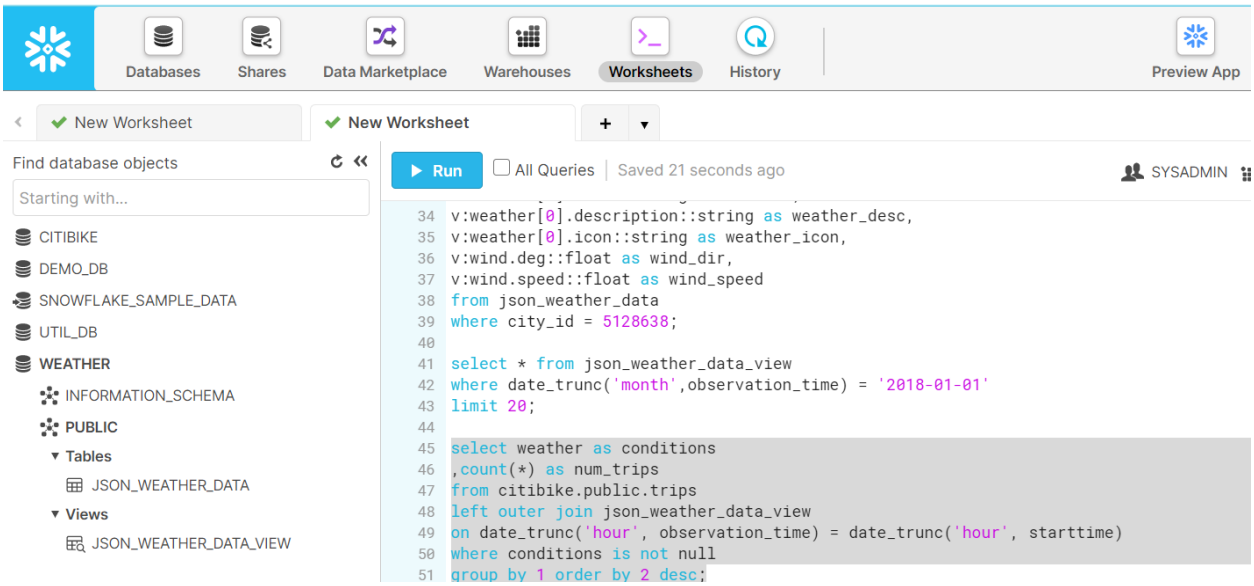

Databases
Shares
Data Ma
New Worksheet
Nev
Find database objects
Starting with...
CITIBIKE
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
UTIL_DB
WEATHER
INFORMATION_SCHEMA
PUBLIC
Tables
JSON_WEATHER_DATA
Views
JSON_WEATHER_DATA_VIEW



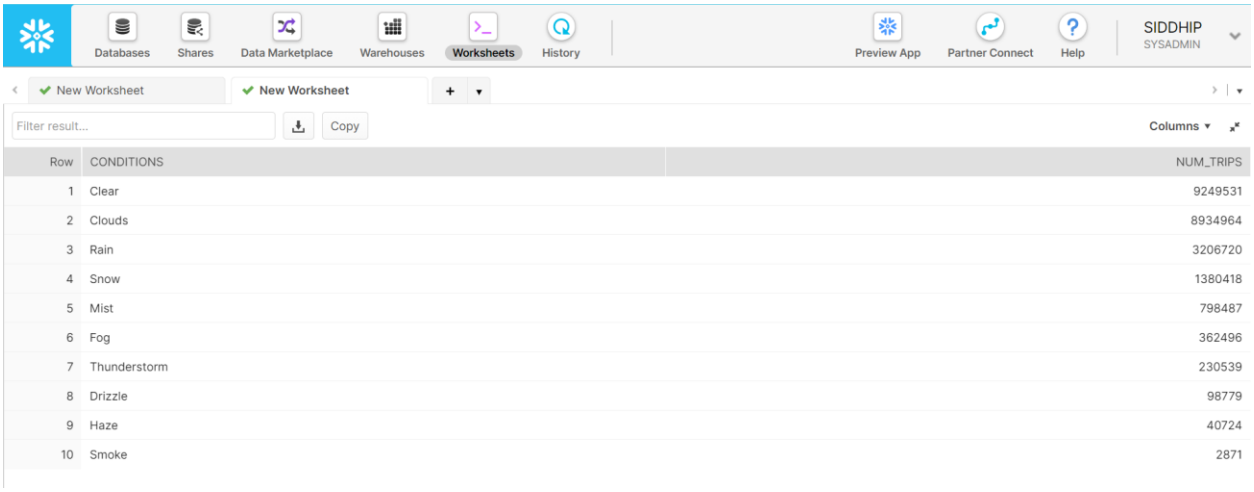


Use a Join Operation to Correlate Against Data Sets

Joining weather and trips data to analyze the impact of weather on the bike rides by counting number of trips associated with certain weather conditions.



So, in the output below we can observe that the number of bike rides is quite higher when the climate is good.



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

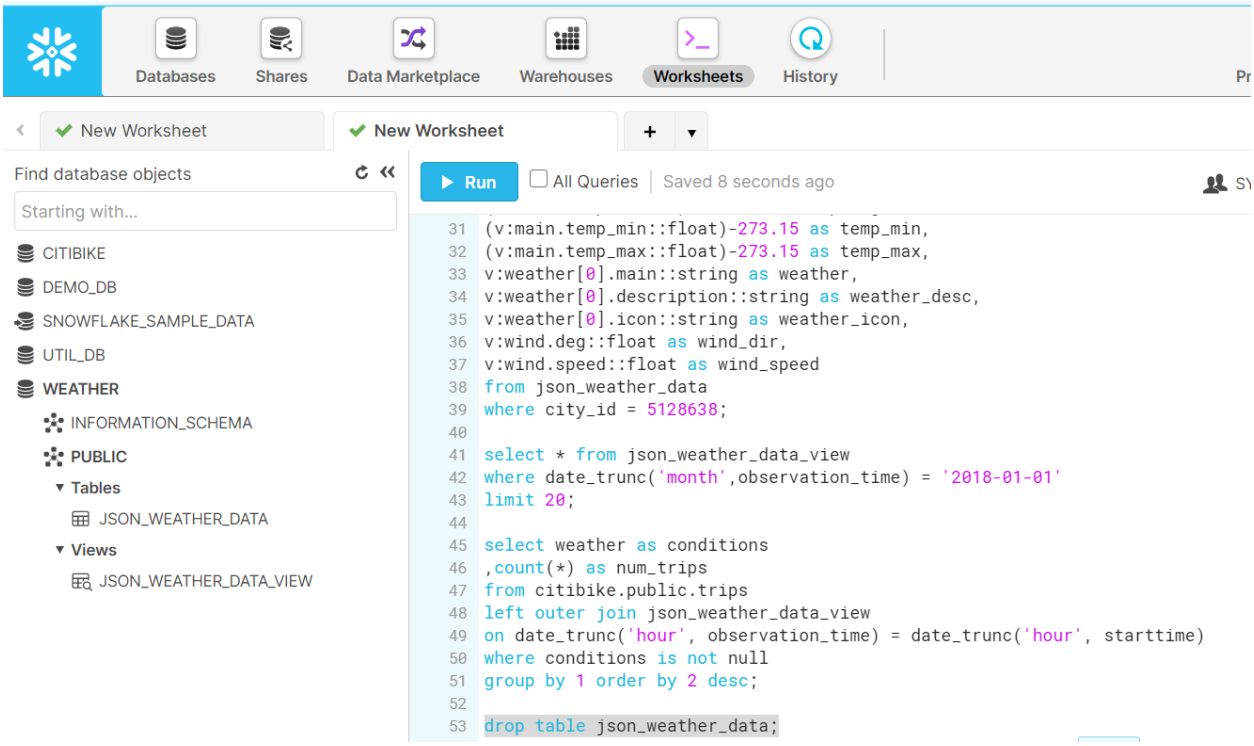
Using Time Travel

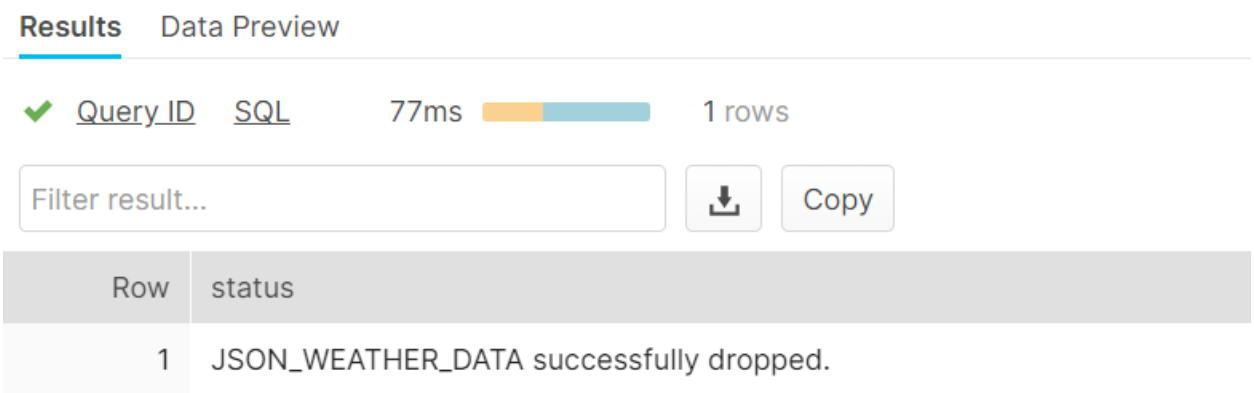
This feature allows us to access the historical data. Below are the applications of time travel in snowflake.

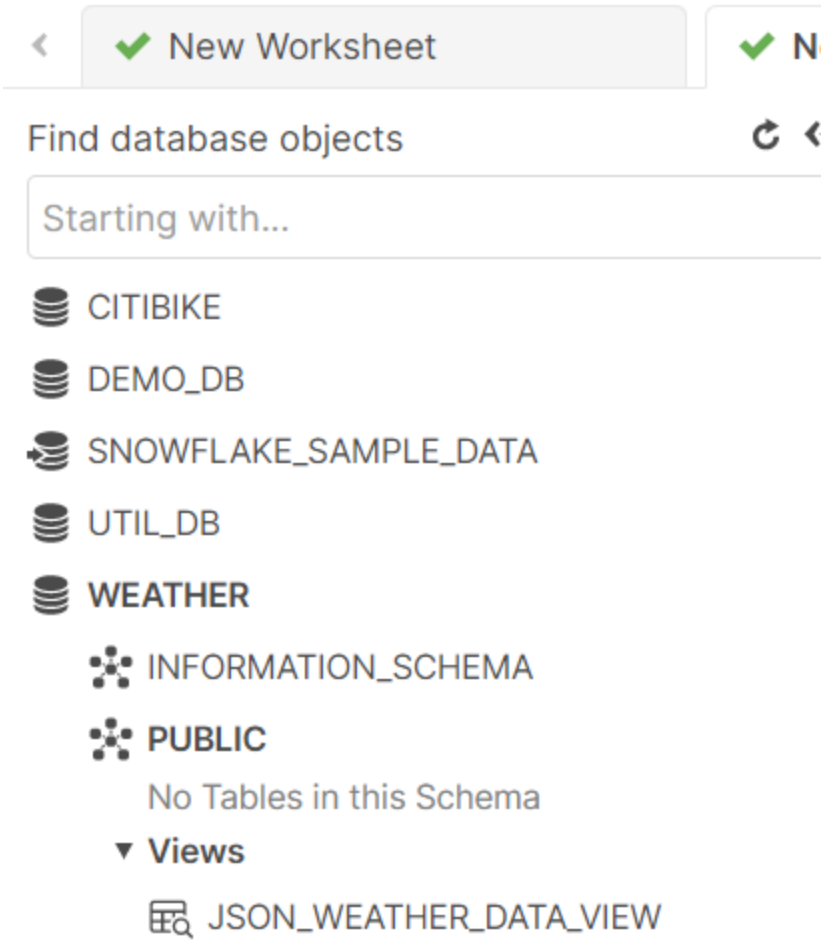
- Restoring data-related objects such as tables, schemas, and databases that may have been deleted
- Duplicating and backing up data from key points in the past
- Analyzing data usage and manipulation over specified periods of time

Drop and Undrop a Table

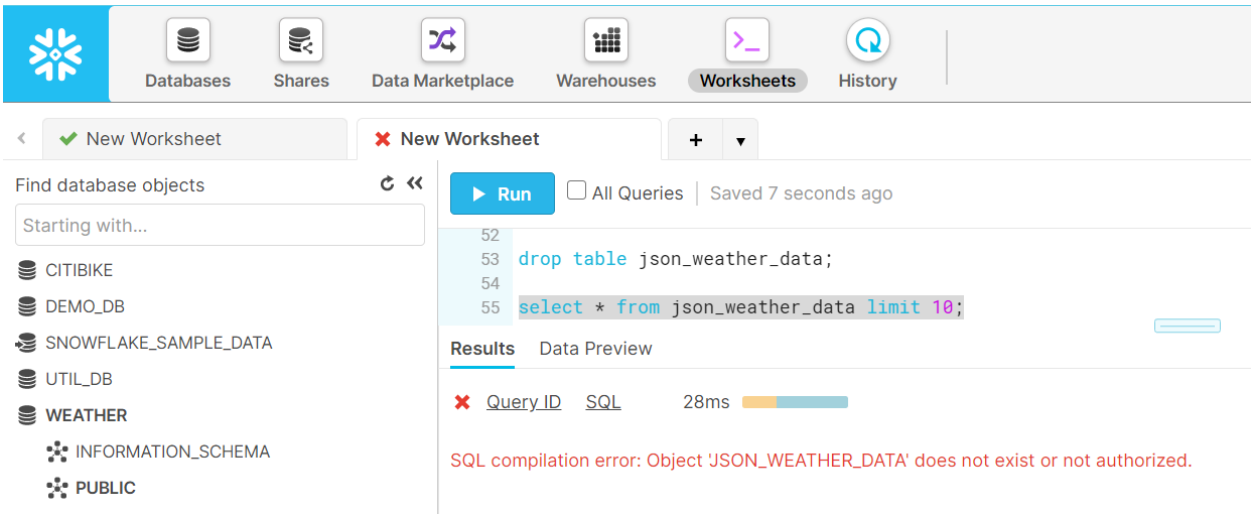
Recover data that has been accidently dropped.



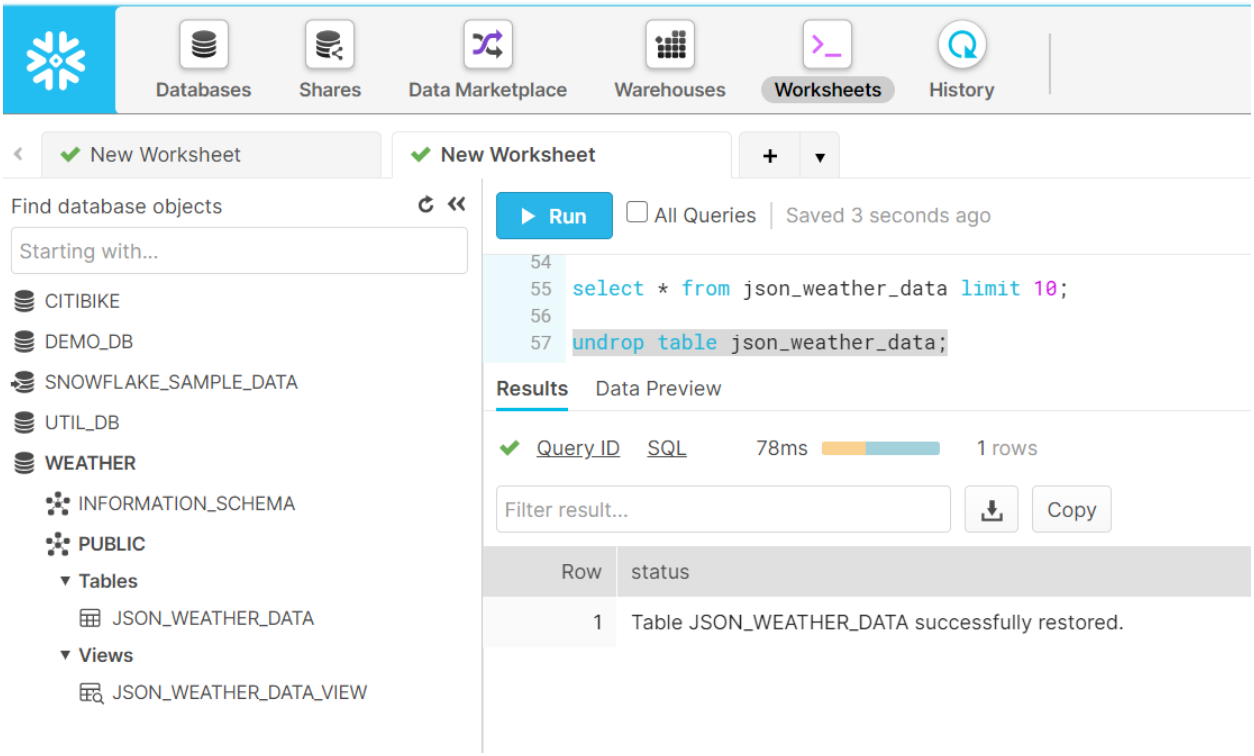


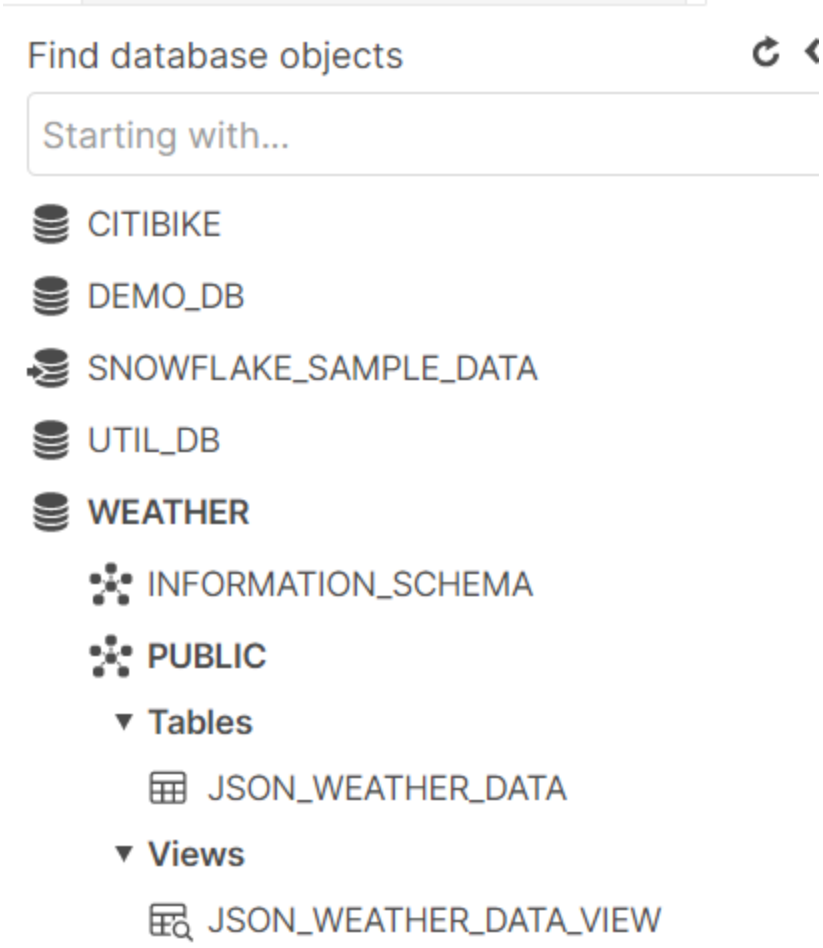


Verify the table has been dropped successfully by running the select command



Restoring the dropped table.

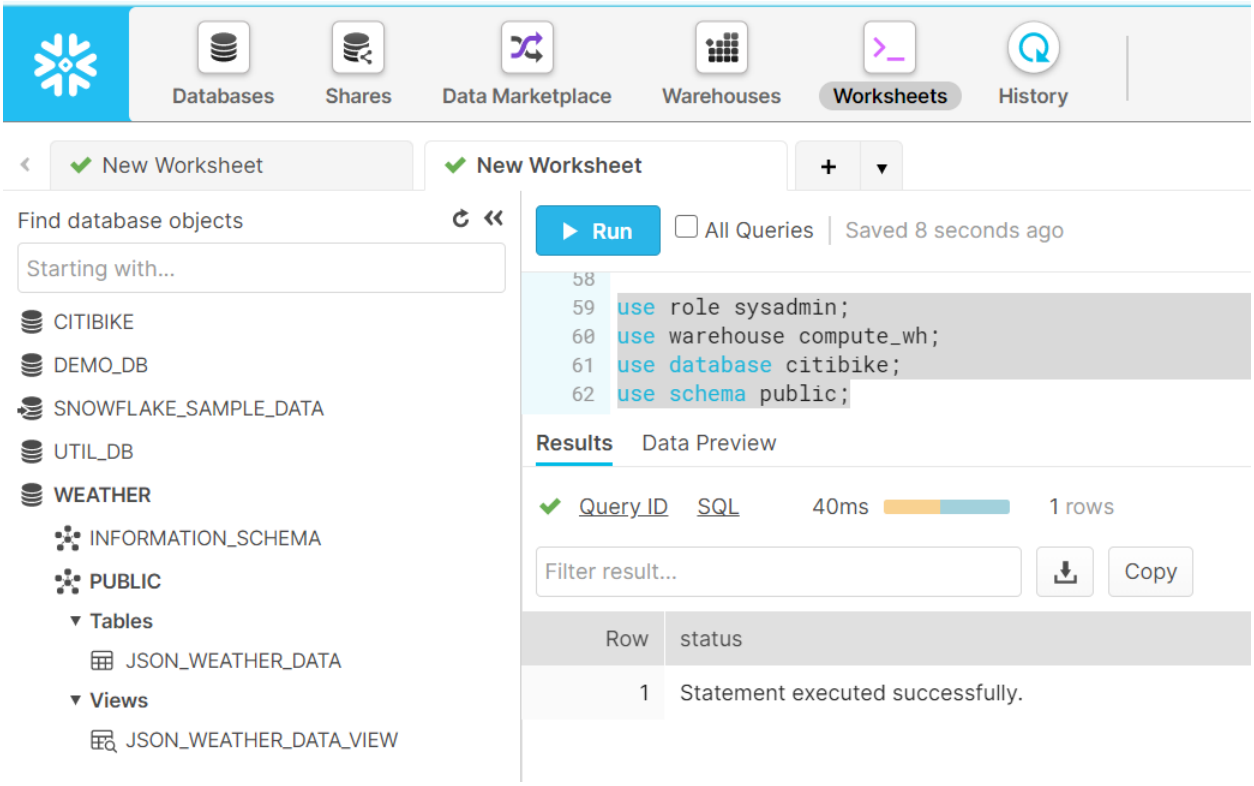


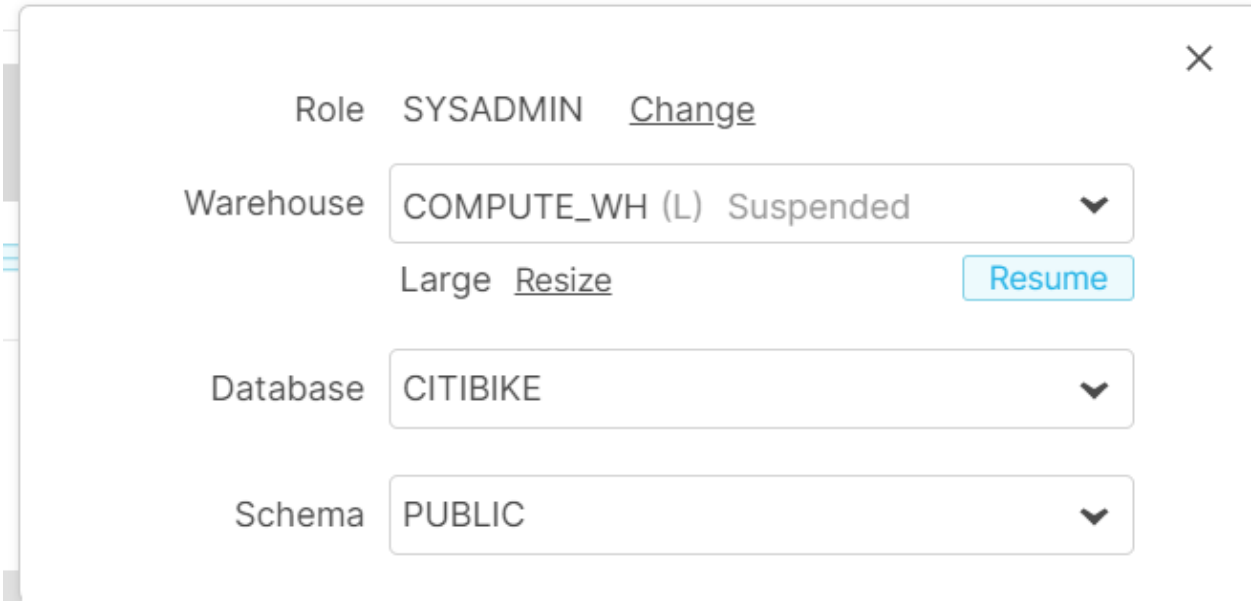

Find database objects
Starting with...
CITIBIKE
DEMO_DB
SNOWFLAKE_SAMPLE_DATA
UTIL_DB
WEATHER
INFORMATION_SCHEMA
PUBLIC
Tables
JSON_WEATHER_DATA
Views
JSON_WEATHER_DATA_VIEW

Roll Back a Table

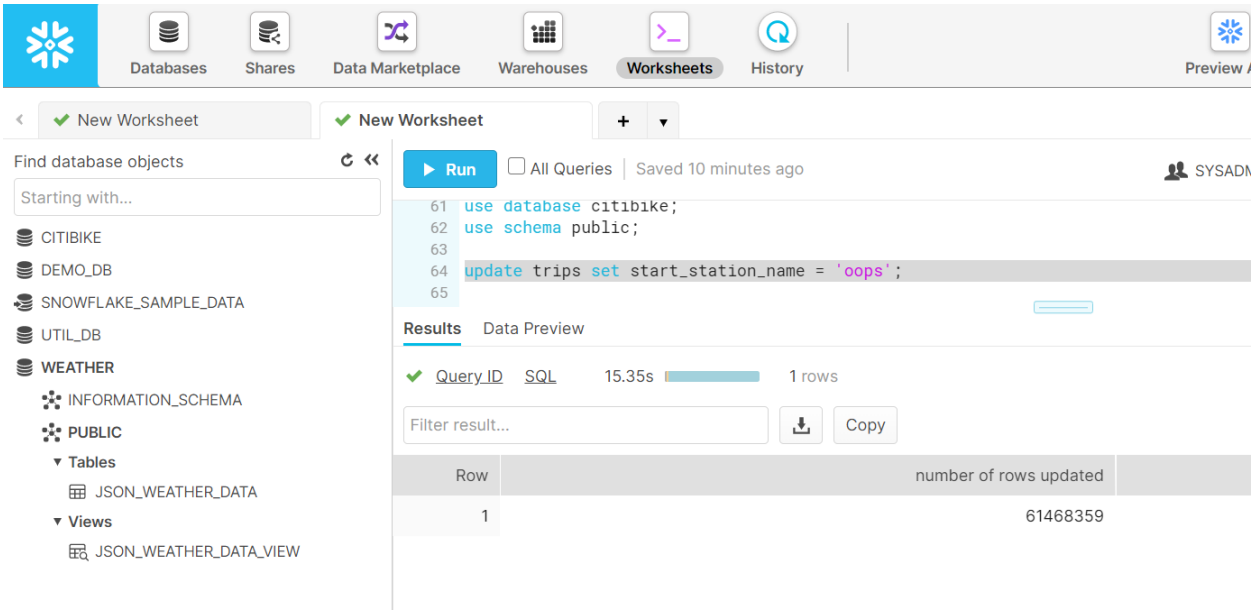
Roll back a table to a previous state to fix an unintentional DML error that replaces all the station names in the CITIBIKE database's TRIPS table with the word "oops."

Verify the context of the worksheet.

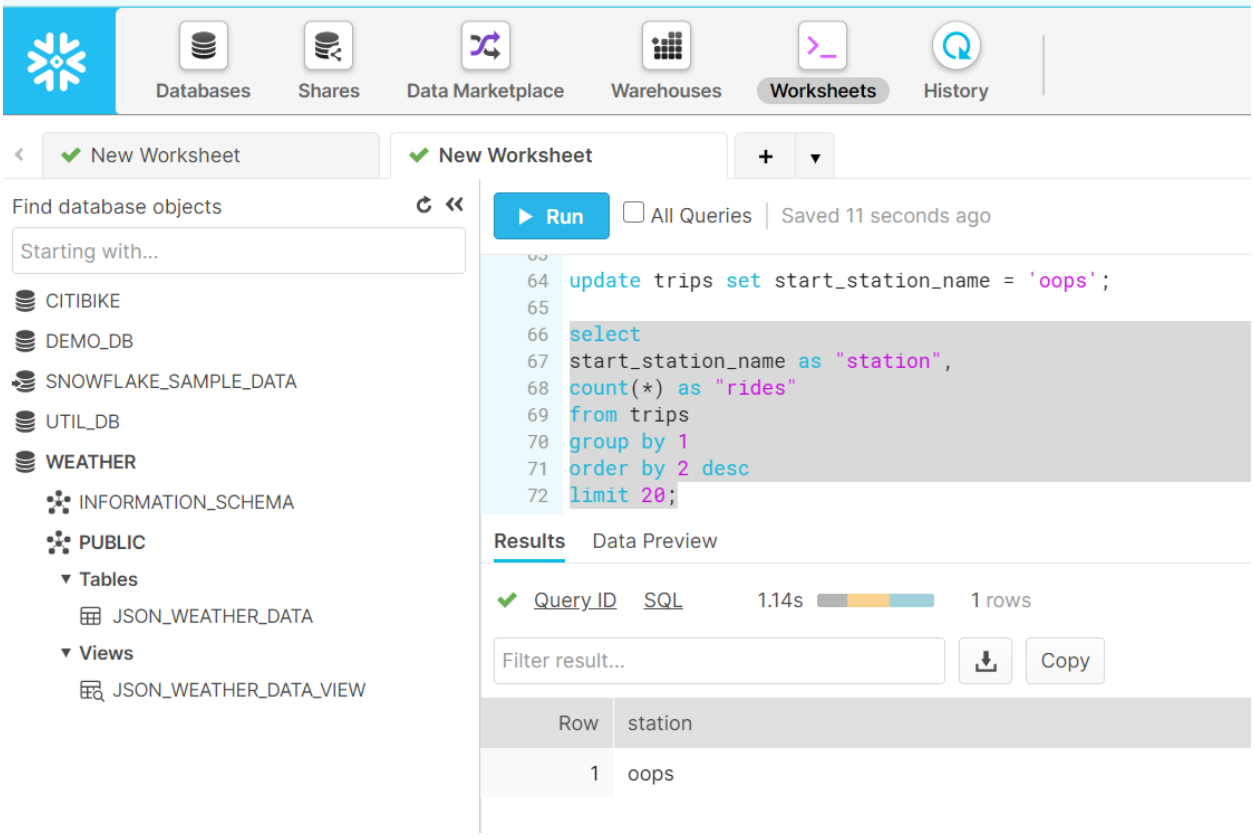




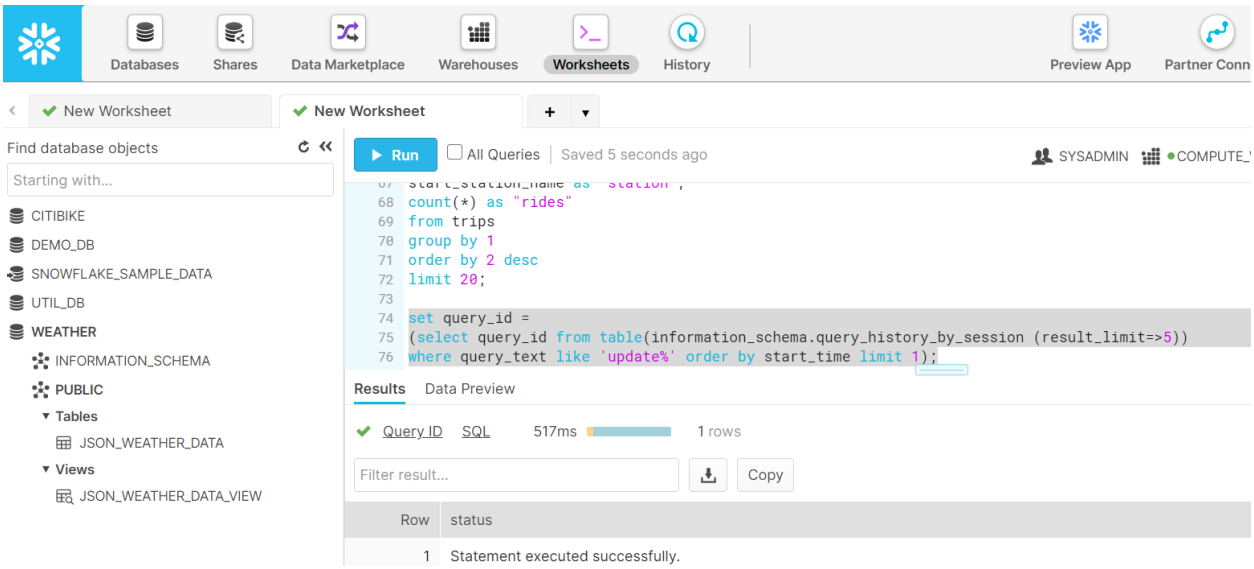
Run the following command to replace all of the station names in the table with the word "oops".



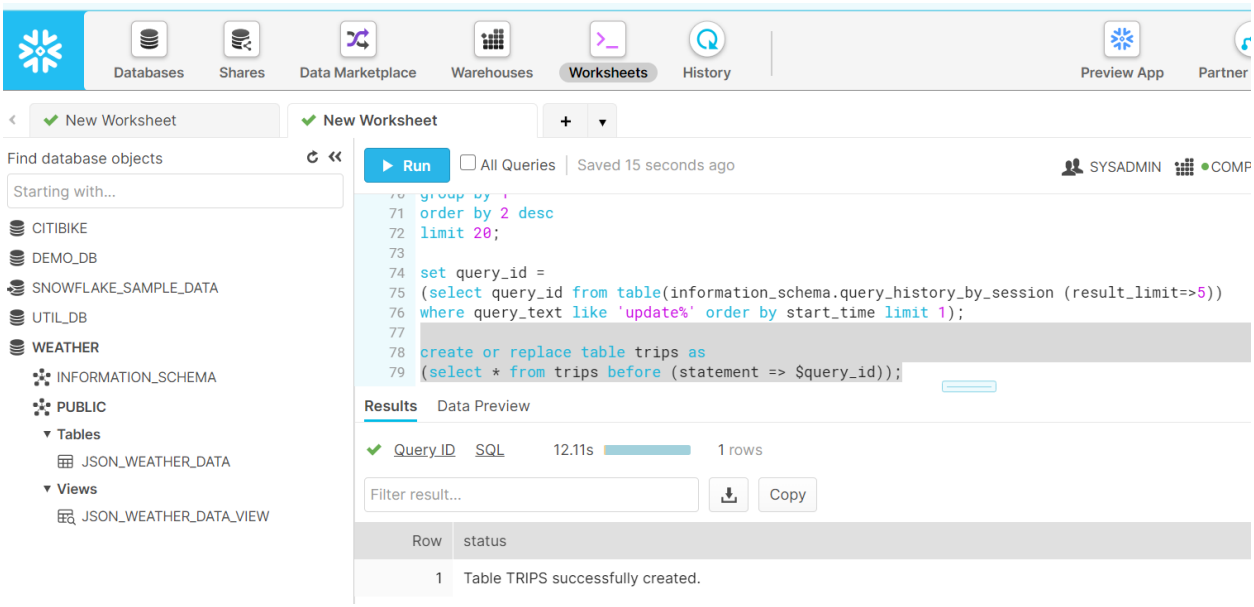
Run a query that returns the top 20 stations by number of rides. Notice that the station names result is only one row.



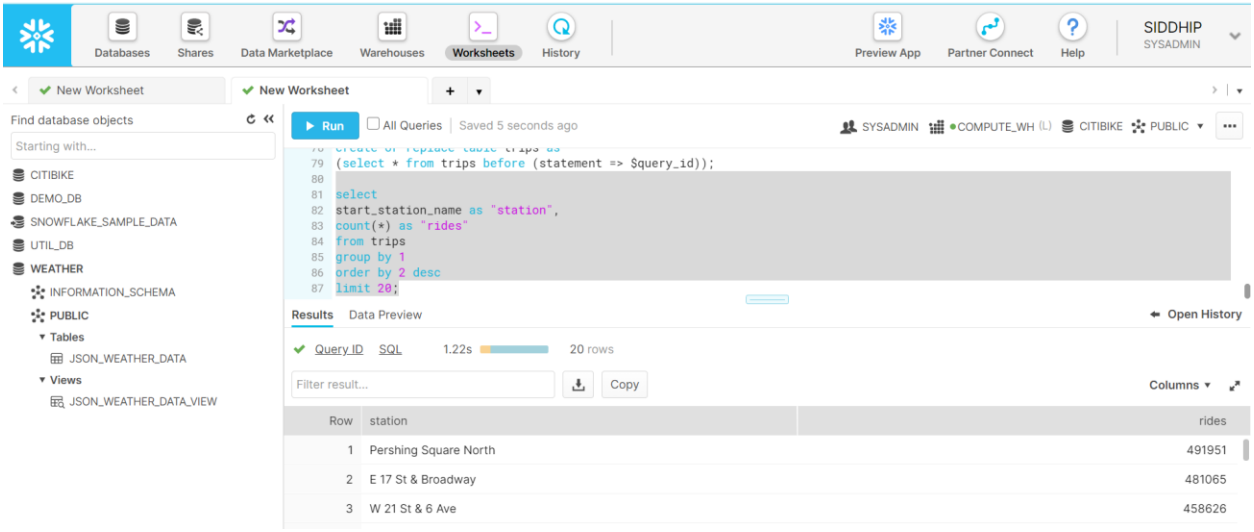
In Snowflake, we can simply run commands to find the query ID of the last UPDATE command and store it in a variable called $QUERY_ID.

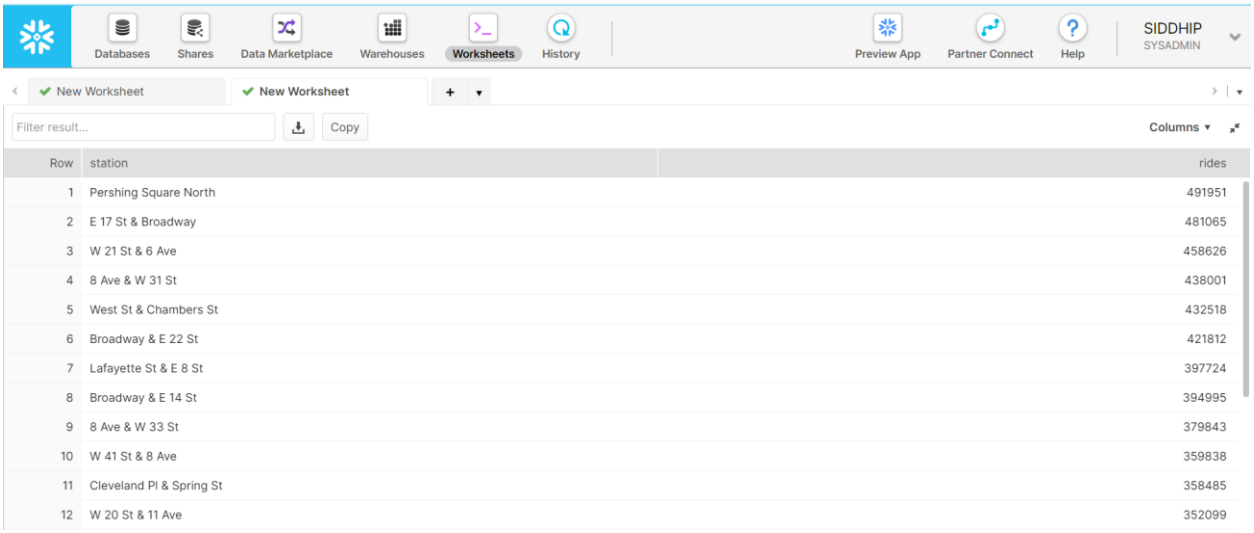


Re-create the table with the correct station names:



Run the SELECT statement again to verify that the station names have been restored:





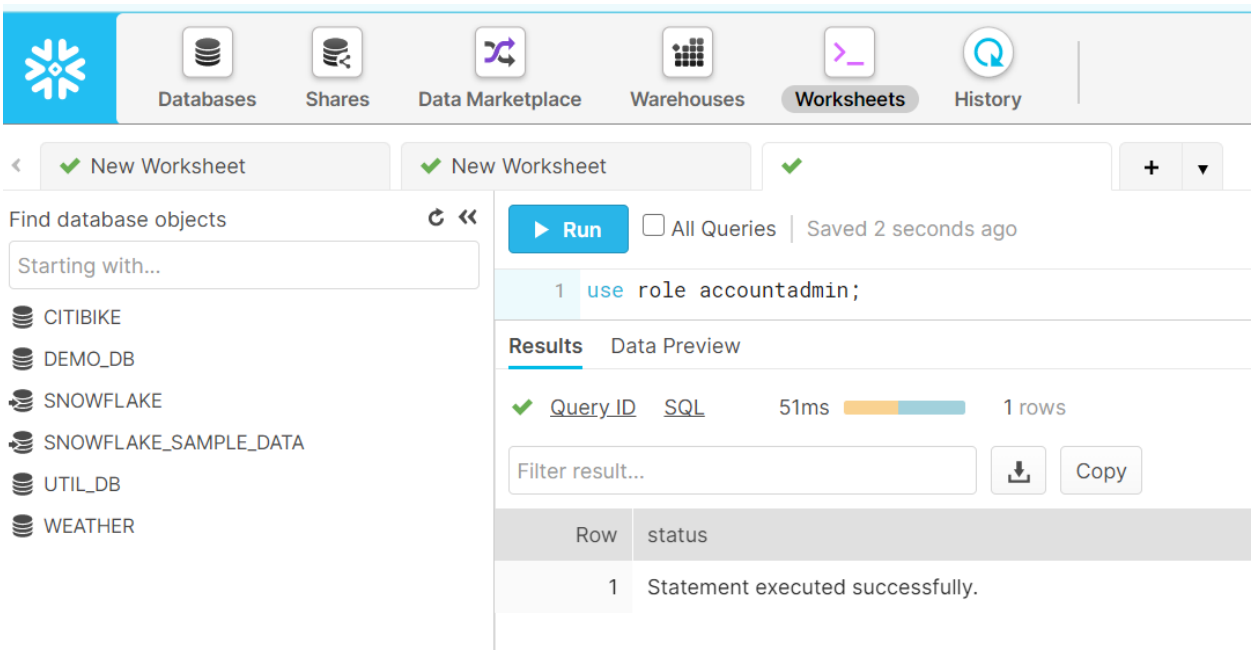
Role-Based Access Controls, Account Usage, and Account Admin

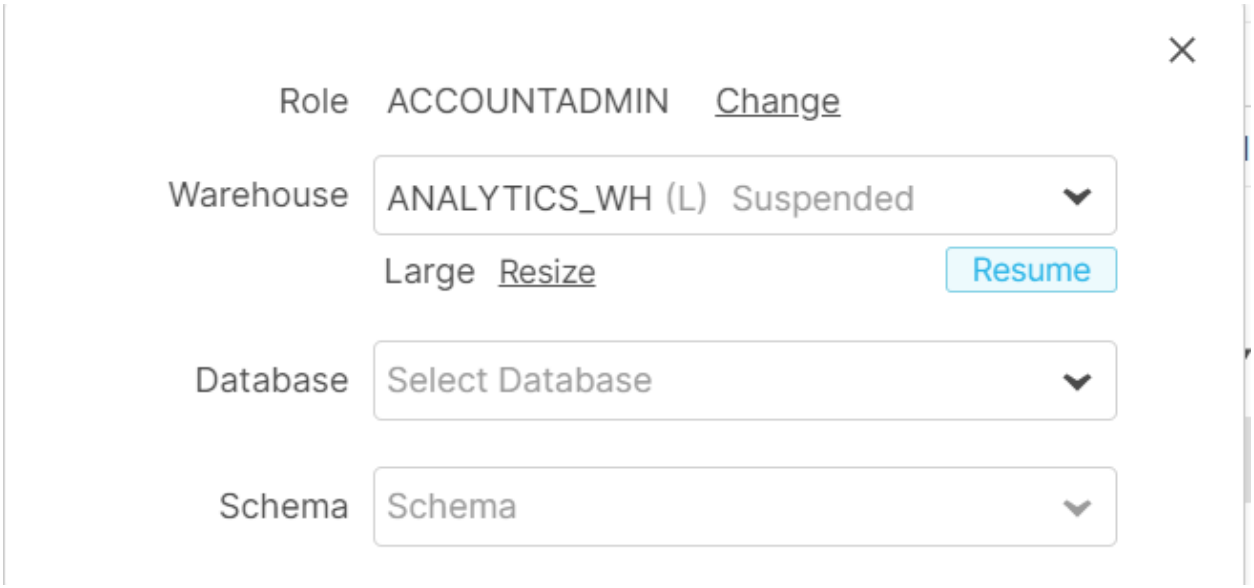
Create a New Role and Add a User

Go to the worksheet -> change to ACCOUNTADMIN role to create a new role.

ACCOUNTADMIN encapsulates the SYSADMIN and SECURITYADMIN system-defined roles. It is the top-level role in the system and should be granted only to a limited number of users in your account

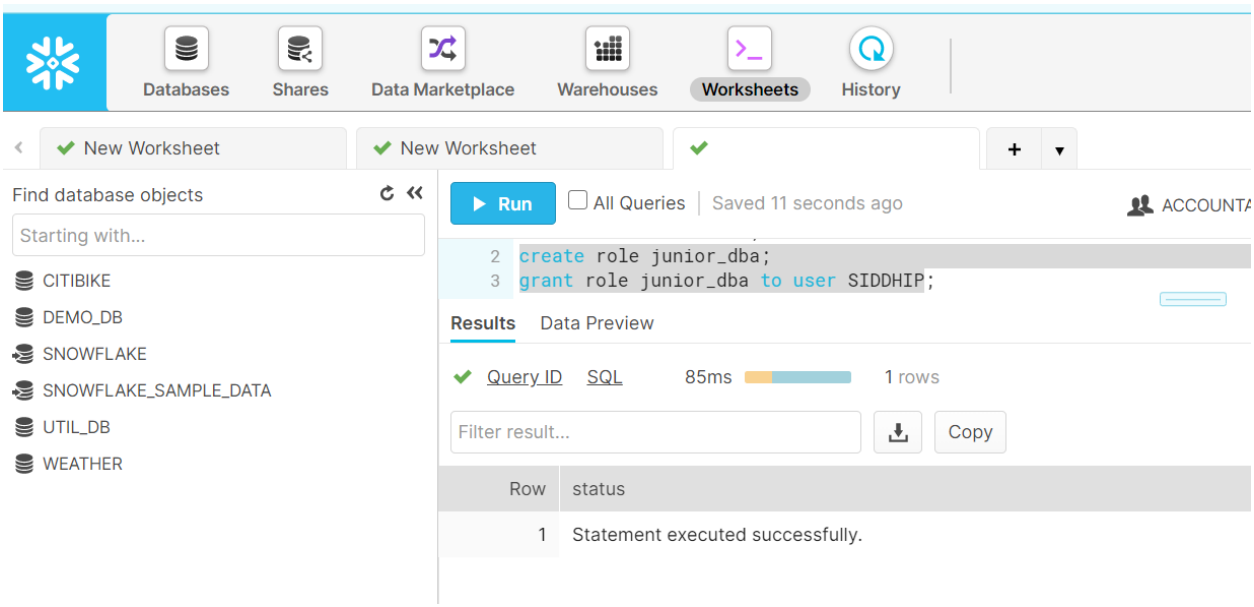
SYSADMIN role cannot create any new roles and hence if we try to create new role while being in SYSADMIN role we will get an error so we need to switch to ACCOUNTADMIN role.





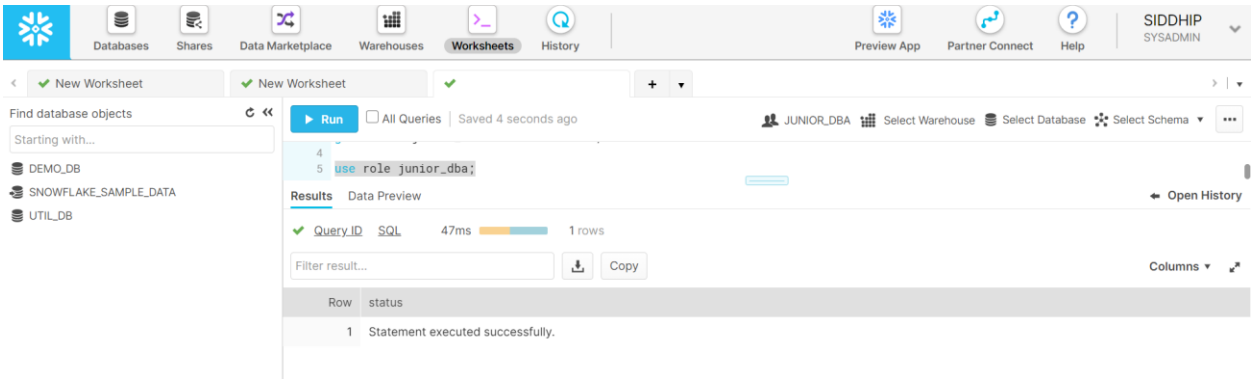
Assigning my username to the newly created user role - junior_dba

Create the role and add your username to it:

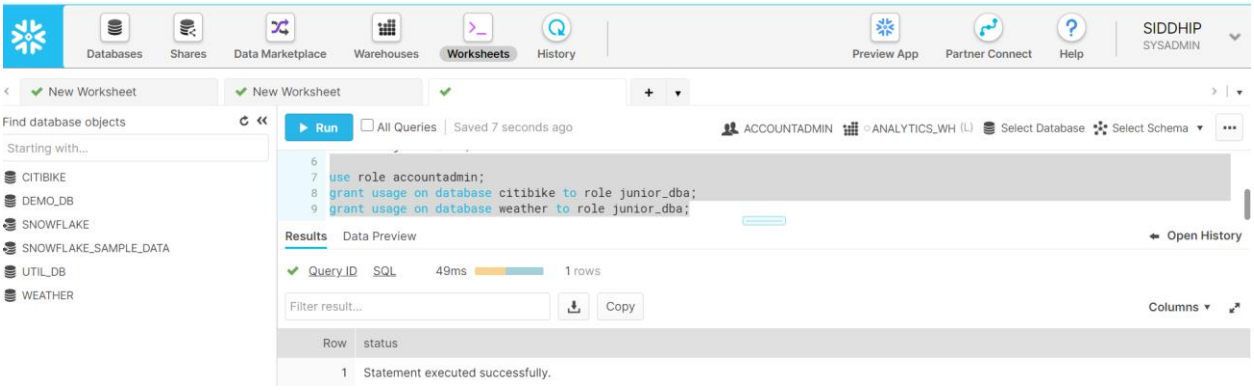


Changing worksheet context to new Junior DBA role.

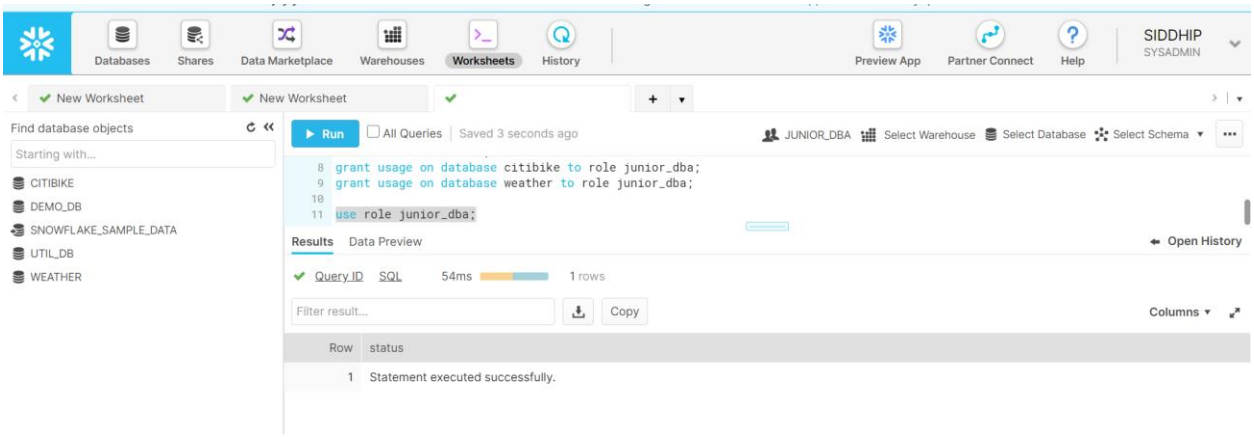
Also, in the left pane CITIBIKE and WEATHER databases no longer appear. This is because the junior_dba role does not have access to view them.



Switch back to the ACCOUNTADMIN role and grant the junior_dba the ability to view and use the CITIBIKE and WEATHER databases:

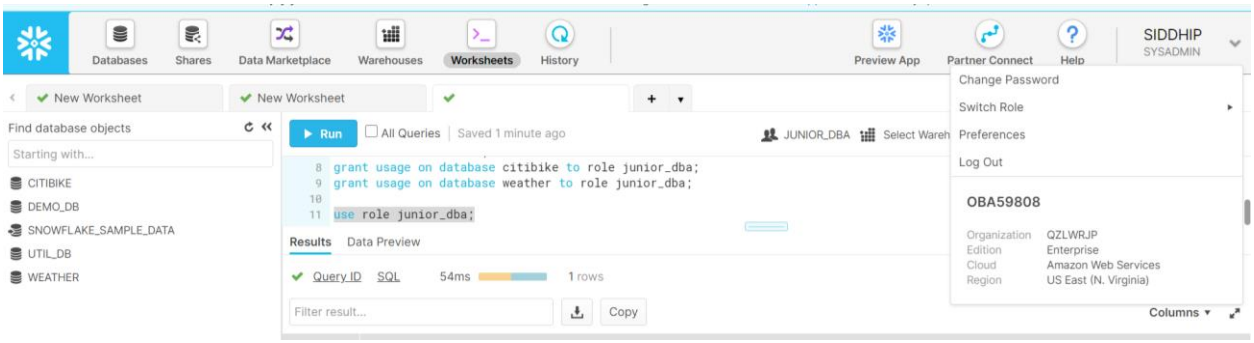


Switching to junior admin role, now we can view CITIBIKE and WEATHER databases on the left side pane.

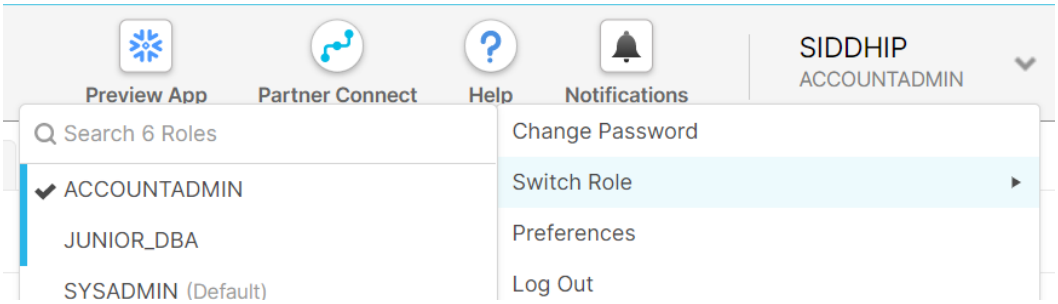


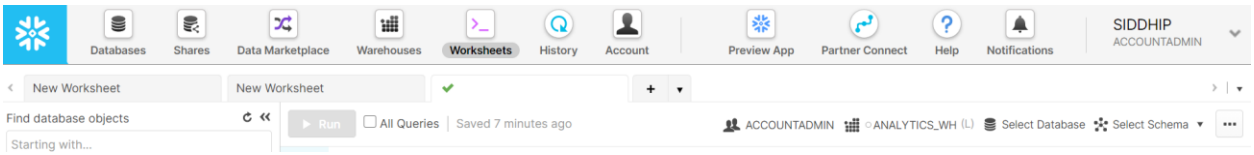
Account Administrator View

In the top right corner of the UI, click on your username to show the User Preferences menu. Go to Switch Role, then select the ACCOUNTADMIN role.



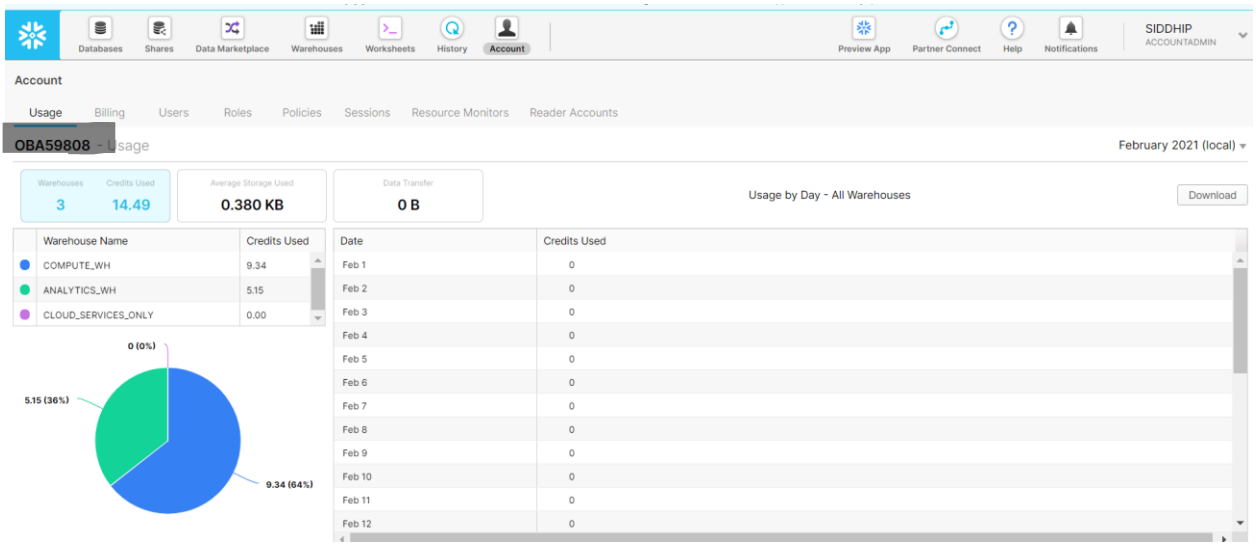
Roles in user preference can be simultaneously different from the role used in the worksheets. Also, we can see that a new tab – 'ACCOUNT' has been added at the top which is only added for ACCOUTADMIN user.





Click on Account tab -> Click on Usage -> Information on credits, storage, and daily or hourly usage for each warehouse, including cloud services. Select a day to review its usage.

Information on Users, Roles, and Resource Monitors is also present. The latter set limits on your account's credit consumption so you can appropriately monitor and manage your credits.



Secure Data Sharing & Data Marketplace

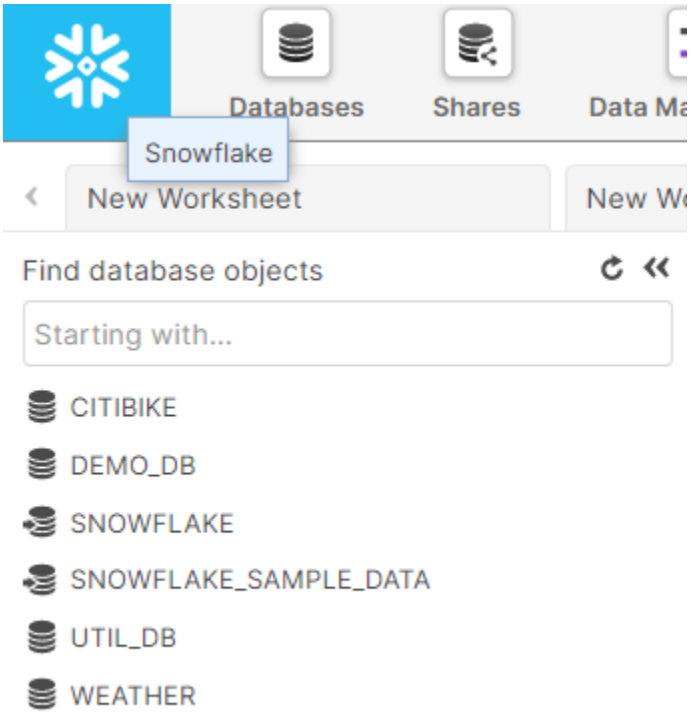
We can access data as shares in snowflake. Provider provides data as shares which consumers(external entity or different internal business unit) import. There can be snowflake reader account or consumer's own account to import data.

With Secure Data Sharing:

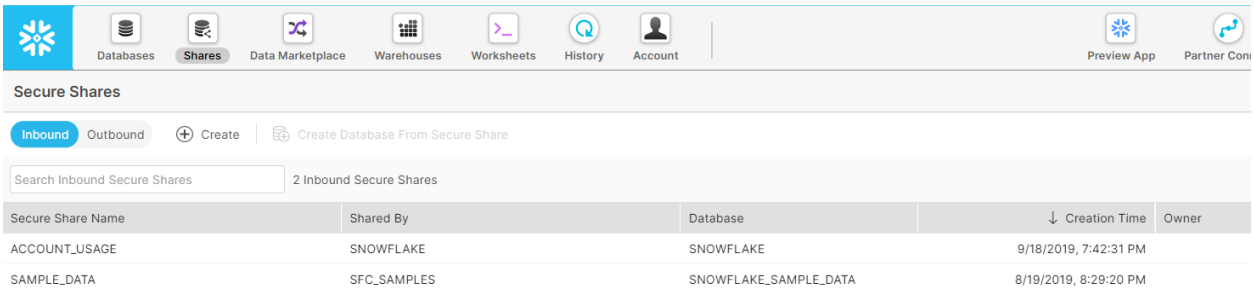
- There is only one copy of the data that lives in the data provider's account
- Shared data is always live, real-time, and immediately available to consumers
- Providers can establish revocable, fine-grained access to shares
- Data sharing is simple and safe, especially compared to older data sharing methods which were often manual and insecure, involving the transfer of large .csv files across the internet

See Existing Shares

Click on the blue Snowflake logo -> On the left side of the page in the database object browser, notice the database SNOWFLAKE_SAMPLE_DATA. The small arrow on the database icon indicates this is a share.



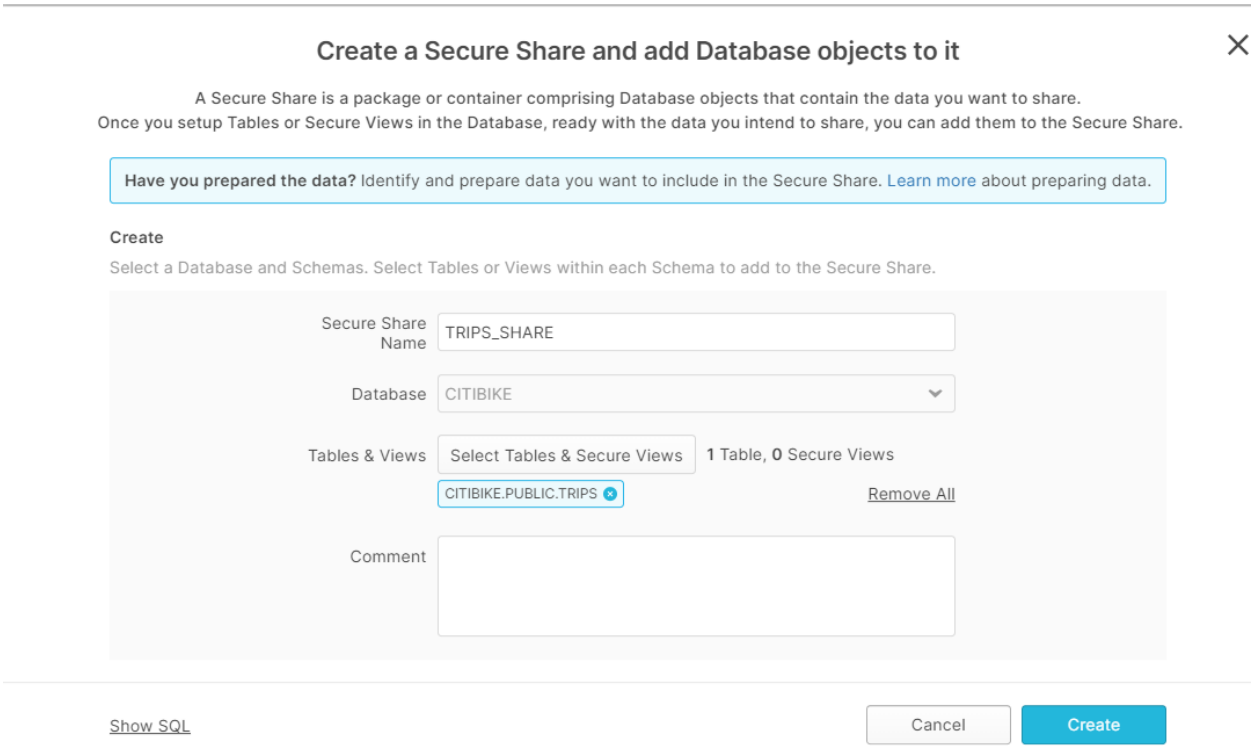
We can go to shares tab to view the existing shares. Our account acts as consumer of these shares.



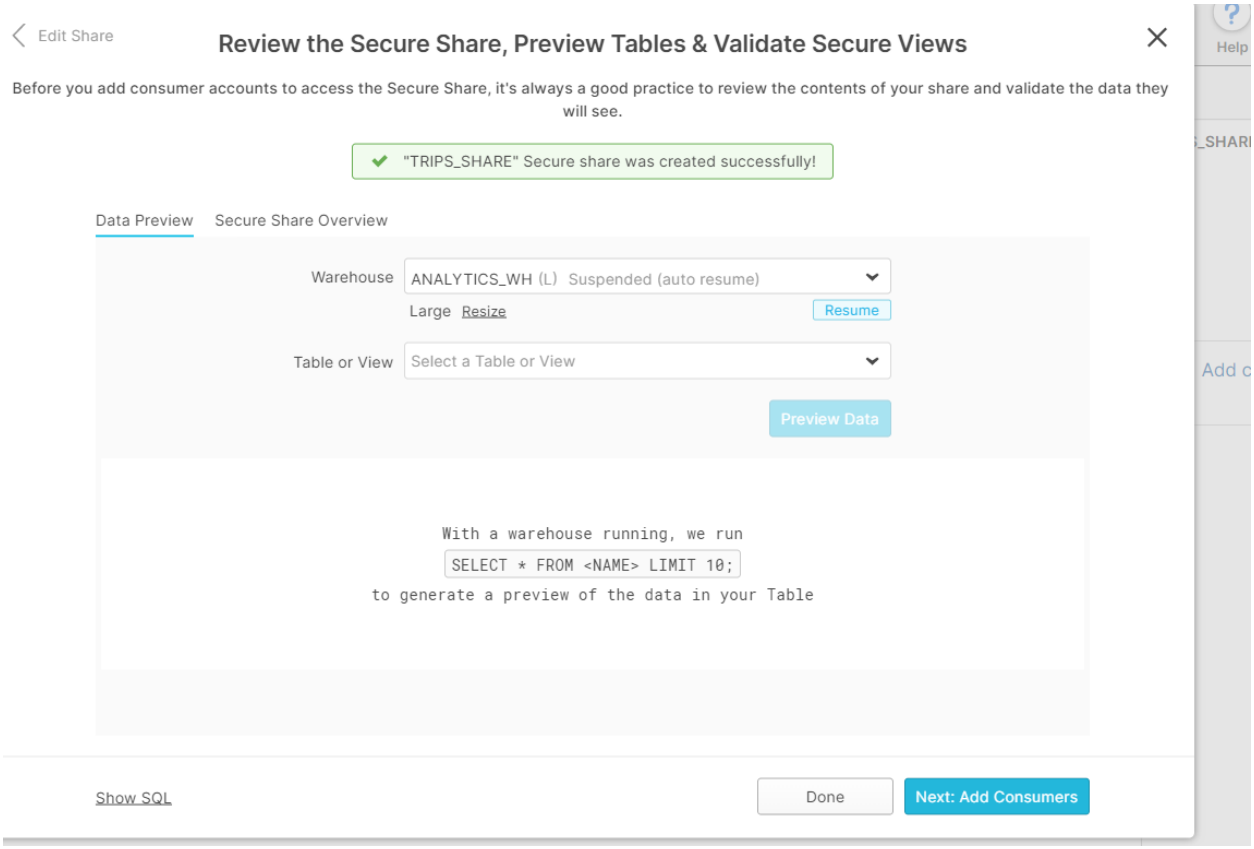
Create an Outbound Share

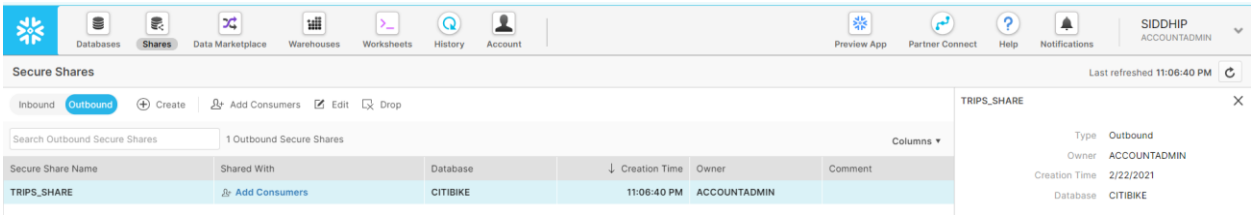
Create an outbound share to give access to your company's trusted partner in order to analyze the data. Also, not extra copies of the data was created. One secure data can be accessed by multiple people.

We can also share UDF's, views and secure joins. We can share data by preventing access to sensitive information as well.



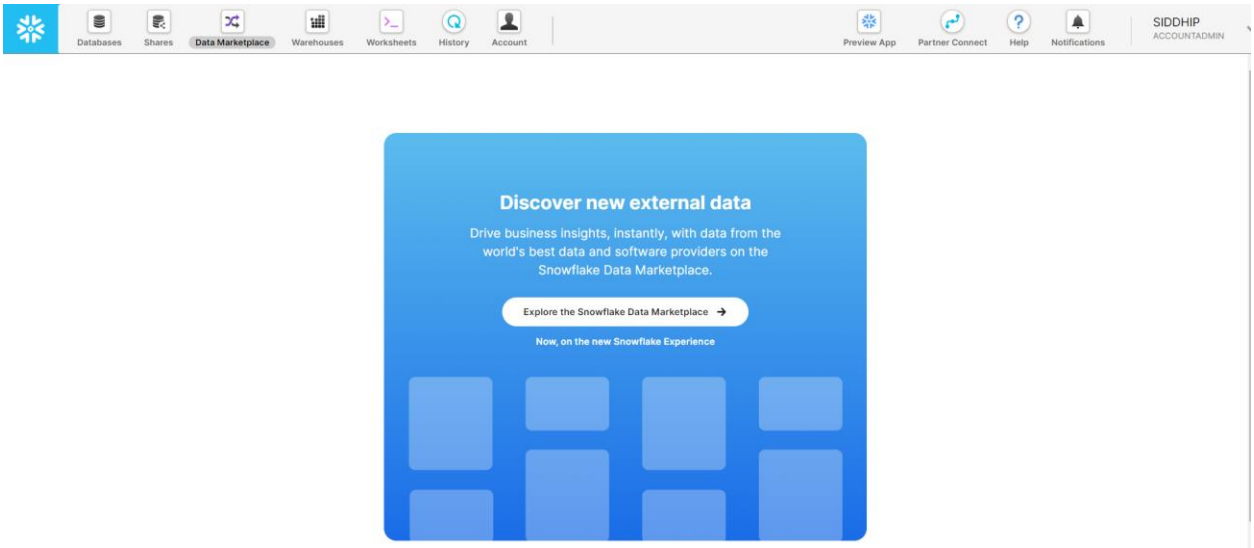
Secure share was created successfully. Account Administrator would click on the Next: Add Consumers button to input their partner's Snowflake account name and type.

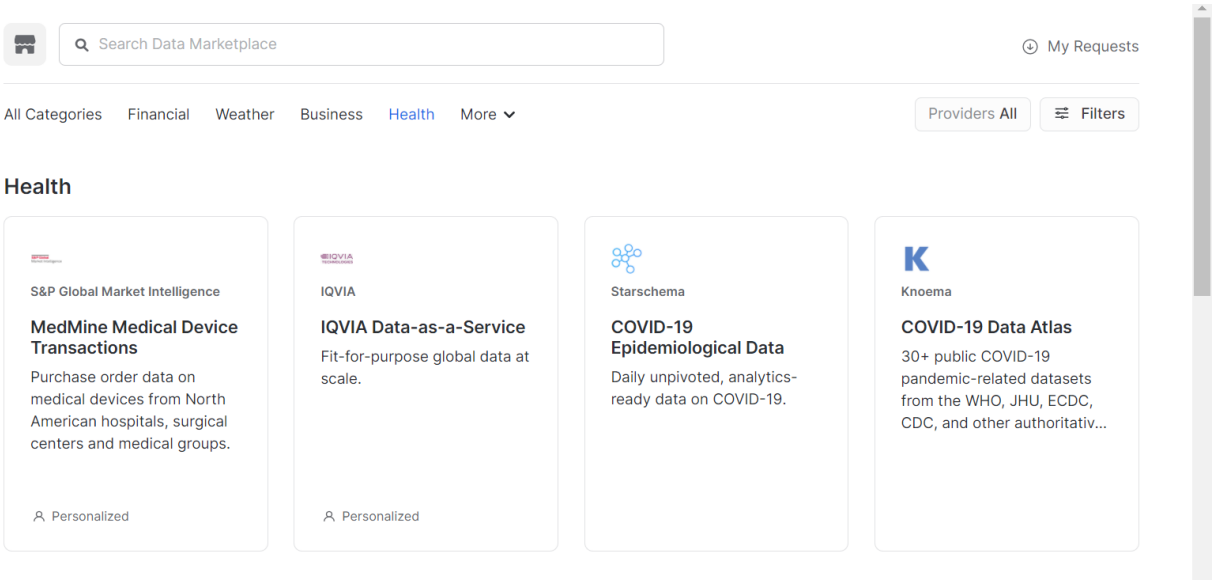

Edit Share
Review the Secure Share, Preview Tables & Validate Secure Views
Before you add consumer accounts to access the Secure Share, it's always a good practice to review the contents of your share and validate the data they will see.
"TRIPS_SHARE" Secure share was created successfully!
Data Preview
Secure Share Overview
Warehouse
ANALYTICS_WH (L) Suspended (auto resume)
Large Resize
Resume
Table or View
Select a Table or View
Preview Data
With a warehouse running, we run
SELECT * FROM <NAME> LIMIT 10;
to generate a preview of the data in your Table
Show SQL
Done
Next: Add Consumers
Help


Databases
Shares
Data Marketplace
Warehouses
Worksheets
History
Account
Preview App
Partner Connect
Help
Notifications
SIDDHIP
ACCOUNTADMIN
Secure Shares
Last refreshed 11:06:40 PM
Inbound
Outbound
Create
Add Consumers
Edit
Drop
Search Outbound Secure Shares
1 Outbound Secure Shares
Columns
Secure Share Name
Shared With
Database
Creation Time
Owner
Comment
TRIPS_SHARE
Add Consumers
CITIBIKE
11:06:40 PM
ACCOUNTADMIN
TRIPS_SHARE
Type Outbound
Owner ACCOUNTADMIN
Creation Time 2/22/2021
Database CITIBIKE

Snowflake Data Marketplace

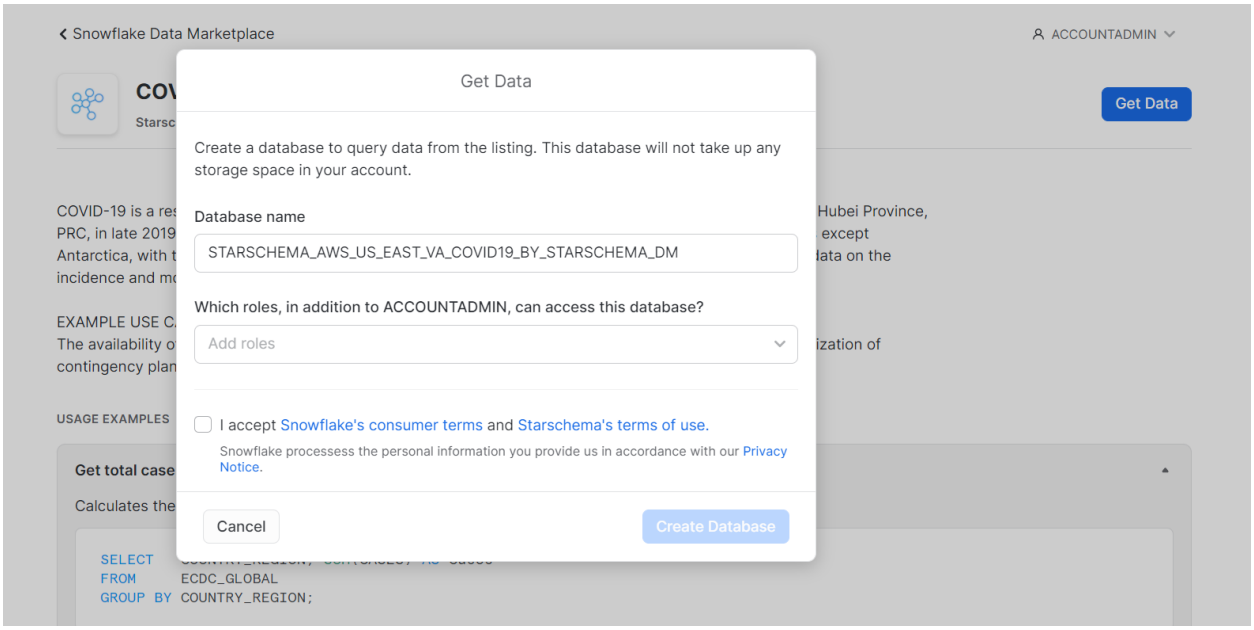
Go to Data Marketplace tab -> Enter your credentials to login -> view categories of data available on top -> click on ‘Health’



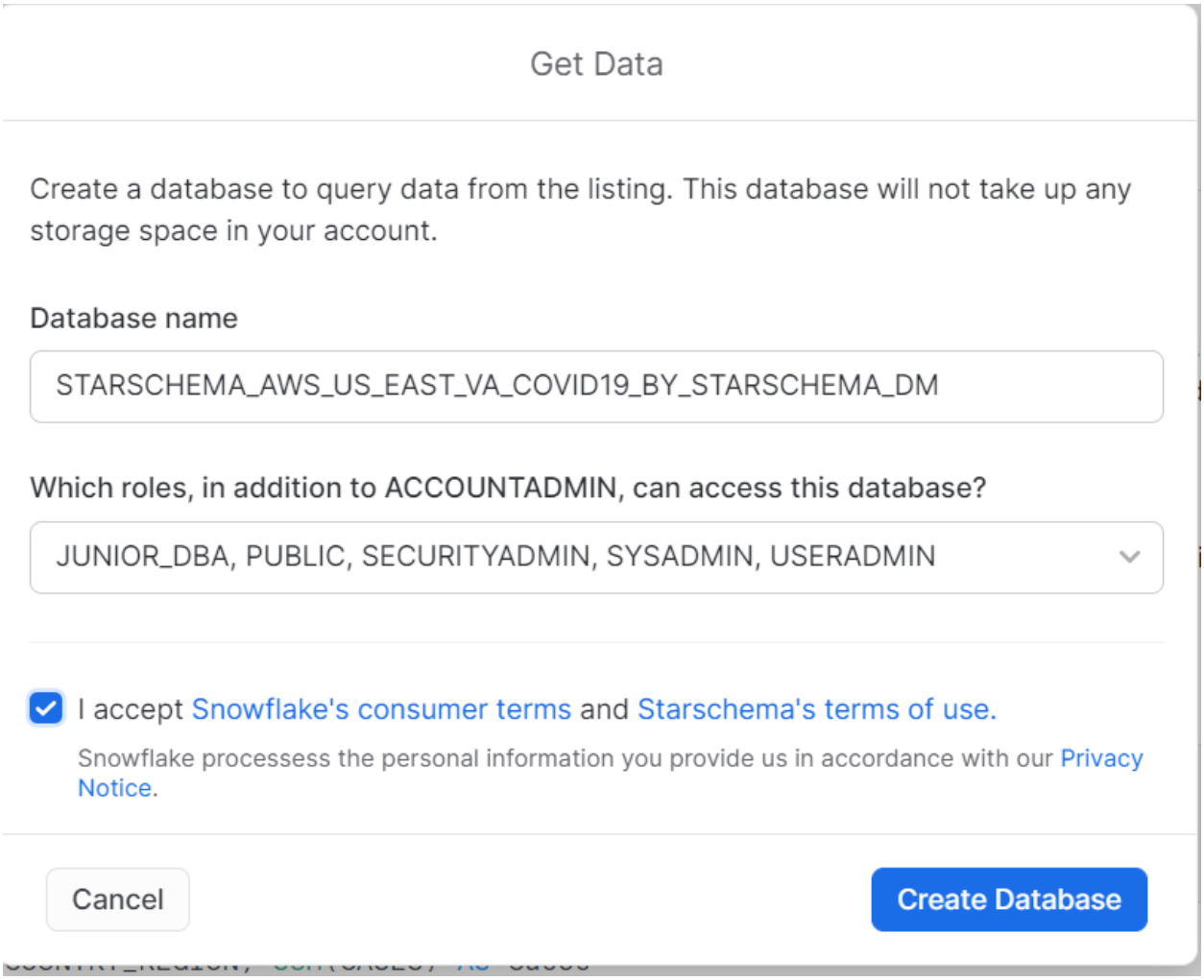


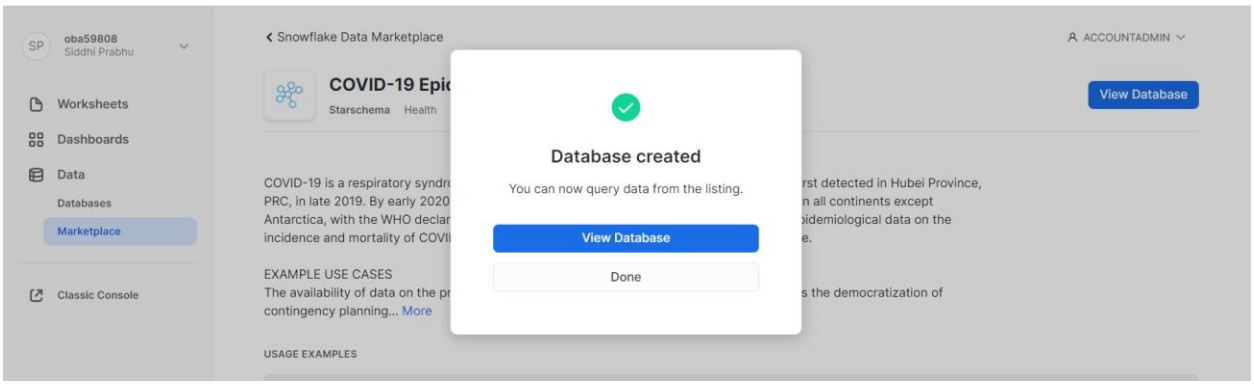
We will use Starschema's COVID-19 Epidemiological Data. Make sure you are in the ACCOUNTADMIN role.

select the data and click on the Get Data button to access this information within your Snowflake Account.



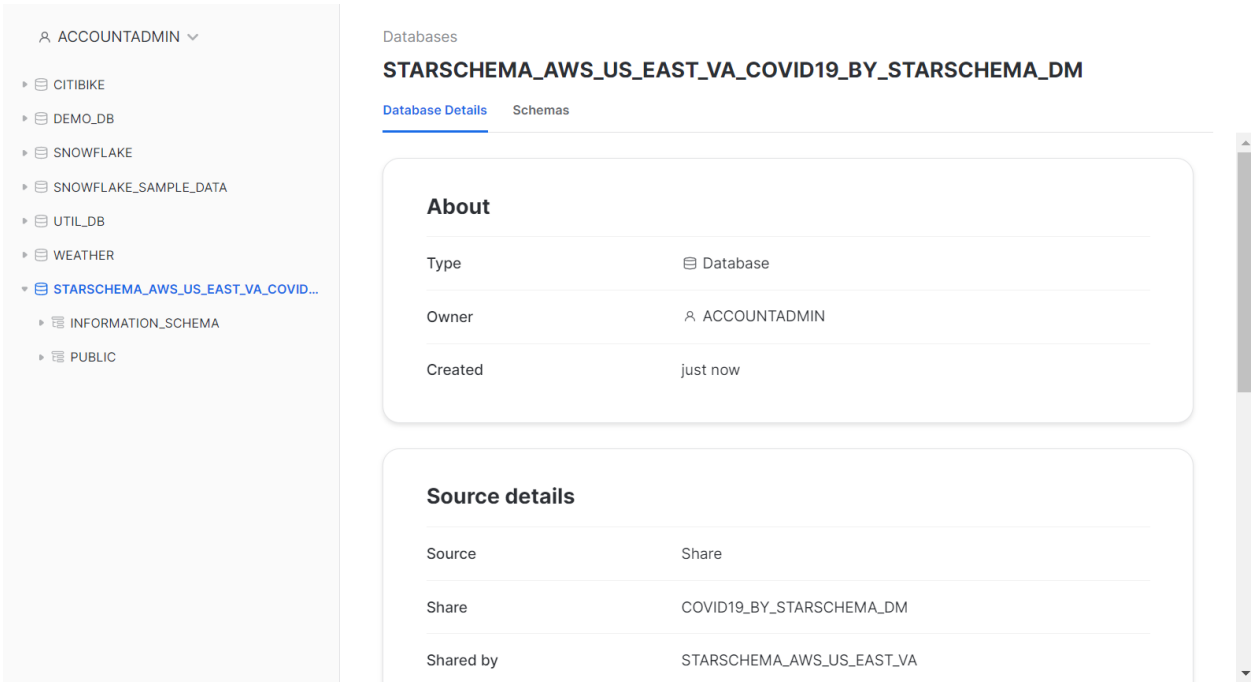
Select all roles and click on terms -> click on create database





We can view Database Details including the owner of the share, number of tables and views in this database, source, share name, and data provider.

View the schemas and details. Select PUBLIC Schema to view available tables.



Databases / STARSCHEMA_AWS_US_EAST_VA_COVID19_BY_STARSCHEMA_DM

PUBLIC

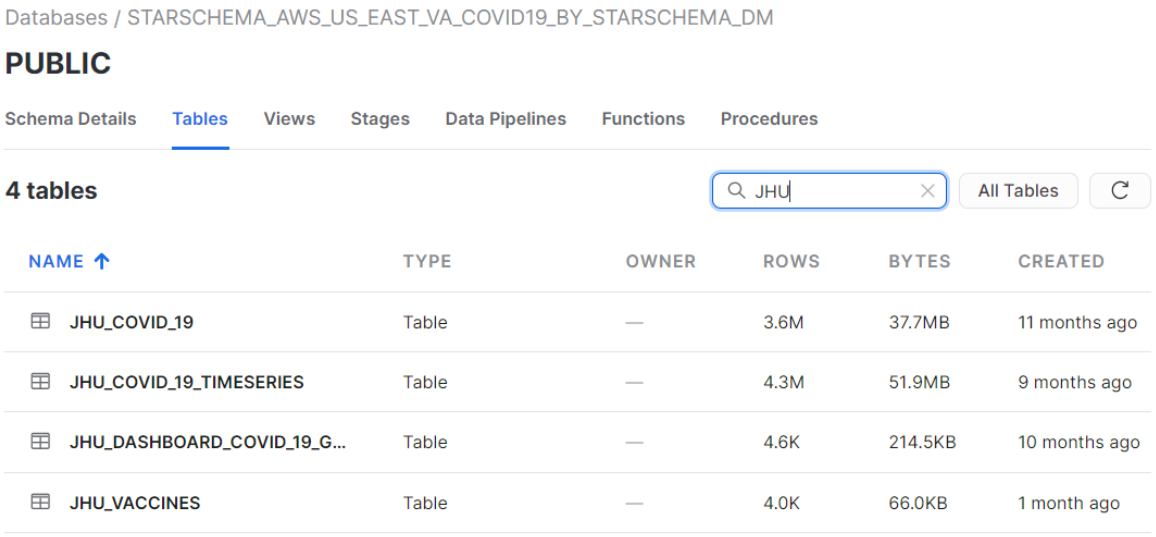
Schema Details | Tables | Views | Stages | Data Pipelines | Functions | Procedures

32 tables

Filter | All Tables

NAME	TYPE	OWNER	ROWS	BYTES	CREATED
CT_US_COVID_TESTS	Table	—	20.0K	640.0KB	11 months ago
DATABANK_DEMOGRAPHICS	Table	—	216	9.0KB	10 months ago
DEMOGRAPHICS	Table	—	3.1K	124.0KB	11 months ago
ECDC_GLOBAL	Table	—	61.9K	639.5KB	9 months ago
GOOG_GLOBAL_MOBILITY_RE...	Table	—	4.3M	39.4MB	10 months ago
HDX_ACAPS	Table	—	23.9K	2.8MB	10 months ago
HS_BULK_DATA	Table	—	727.8K	18.8MB	11 months ago
HUM_RESTRICTIONS_AIRLINE	Table	—	651	64.0KB	10 months ago
HUM_RESTRICTIONS_COUNT...	Table	—	235	601.5KB	10 months ago
IHME_COVID_19	Table	—	189.2K	31.9MB	6 months ago

Type JHU in the search bar to view all tables with data sourced from John Hopkins University. We have now successfully subscribed to the COVID-19 dataset from StarSchema which is updated daily with global COVID data.



Resetting Your Snowflake Environment

First set the worksheet context:

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

Results Data Preview

Query ID SQL 38ms 1 rows

Filter result... Copy

Row	status
1	Statement executed successfully.

Then run this SQL to drop all the objects we created in the lab: