Analyzing Trip Data in Snowflake

Shweta Ajay Shinde

Masters in Data Analytics, San Jose State University

Data 226: Data warehousing

Instructor: Keeyong Han

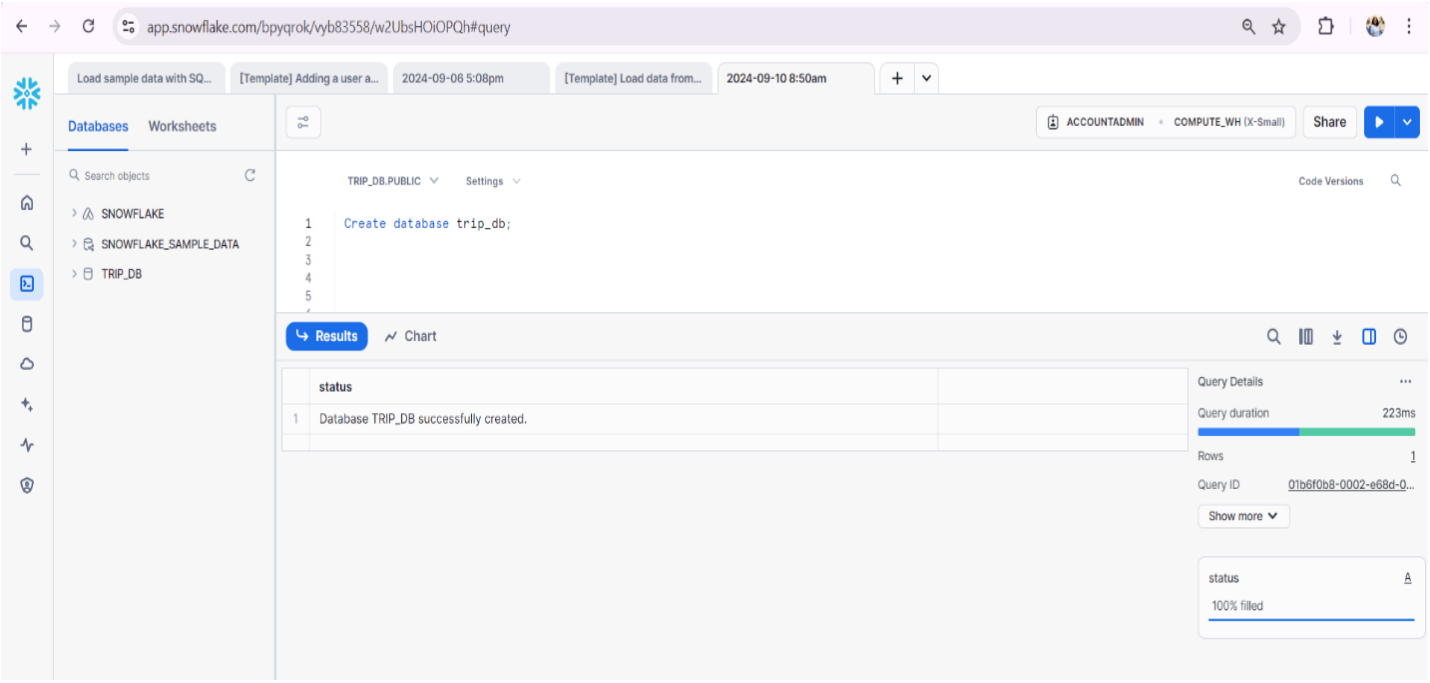
9th Sept 2024

Analyzing Trip Data in Snowflake

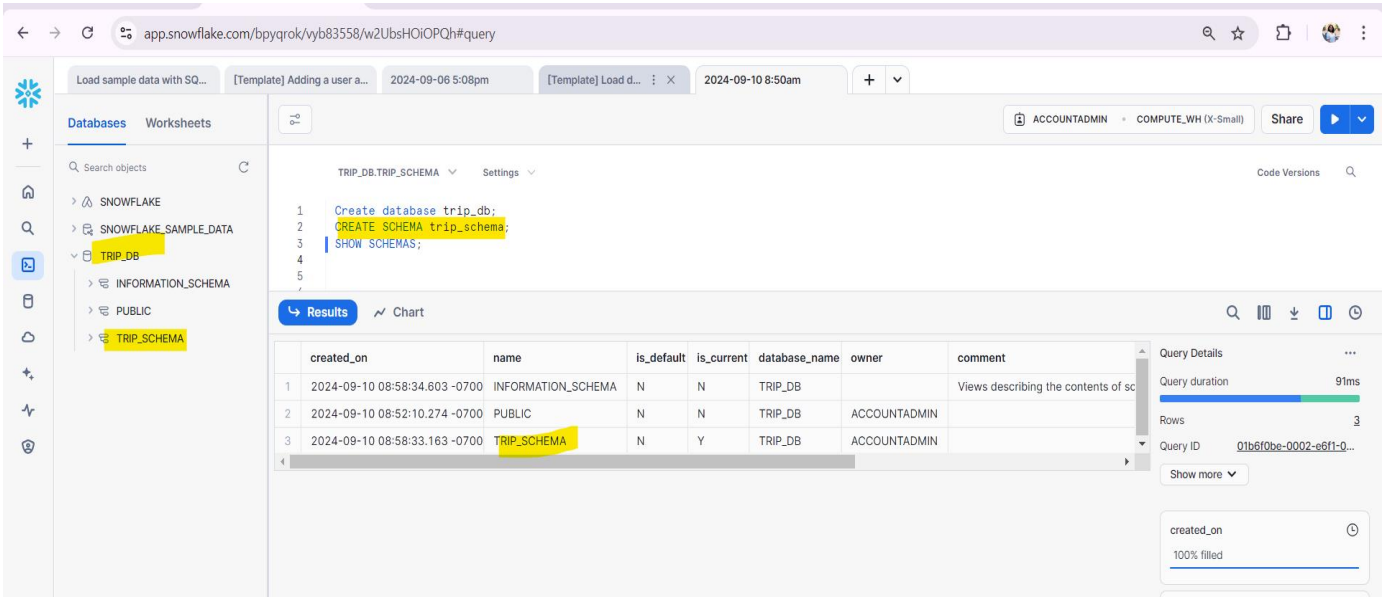
1. Creating objects like Database, Schema, and Table in Snowflake.

   We are using a **COMPUTE_WH (X-Small)** warehouse for the following steps.

   - SQL query to create database:- **CREATE DATABASE trip_db;**



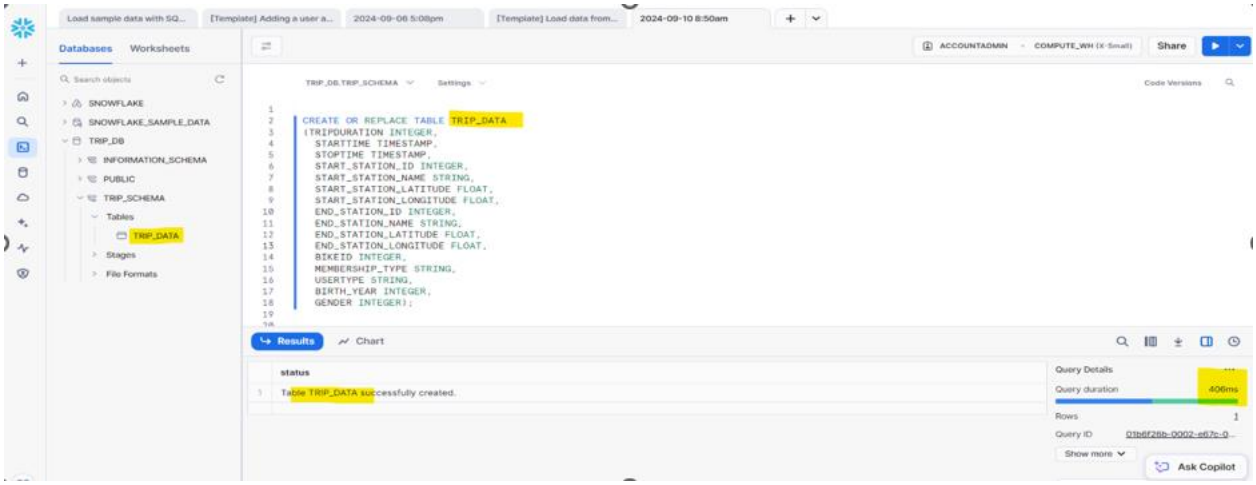
   - SQL query to create schema:- **CREATE SCHEMA trip_schema;**



Analyzing Trip Data in Snowflake

- Query to create table:-

```
CREATE OR REPLACE TABLE TRIP_DATA
(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);
```



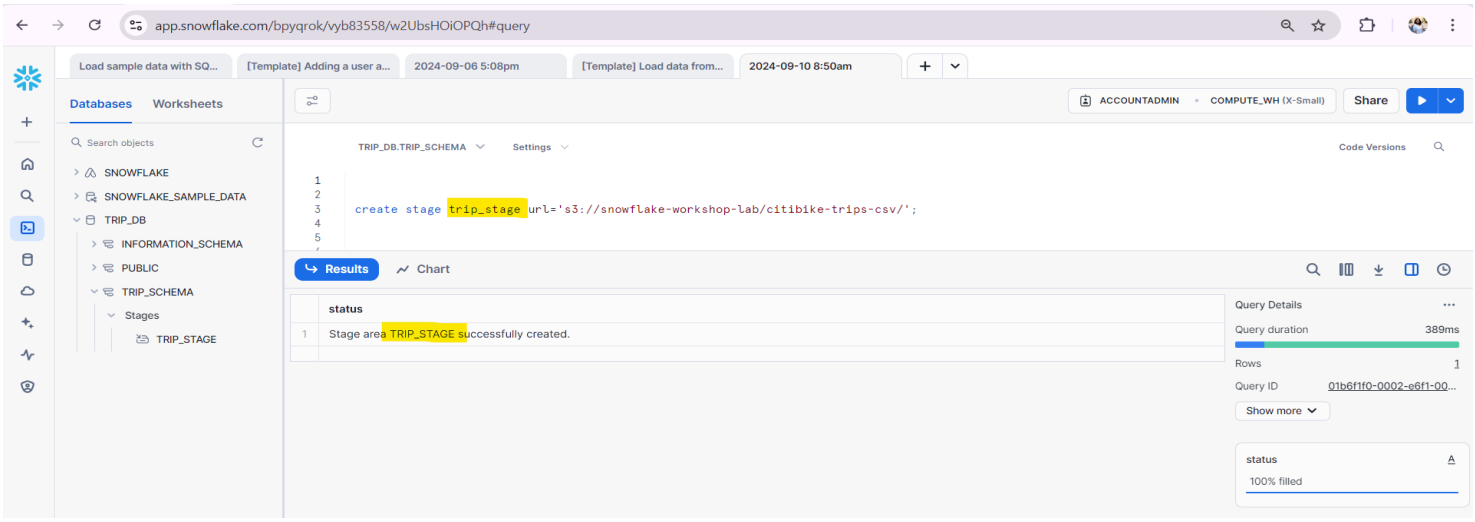
2. Create Stage 'trip_stage' in Snowflake

   Next, create a stage to hold the files. Here's how you can create an external stage:

- SQL query to create stag

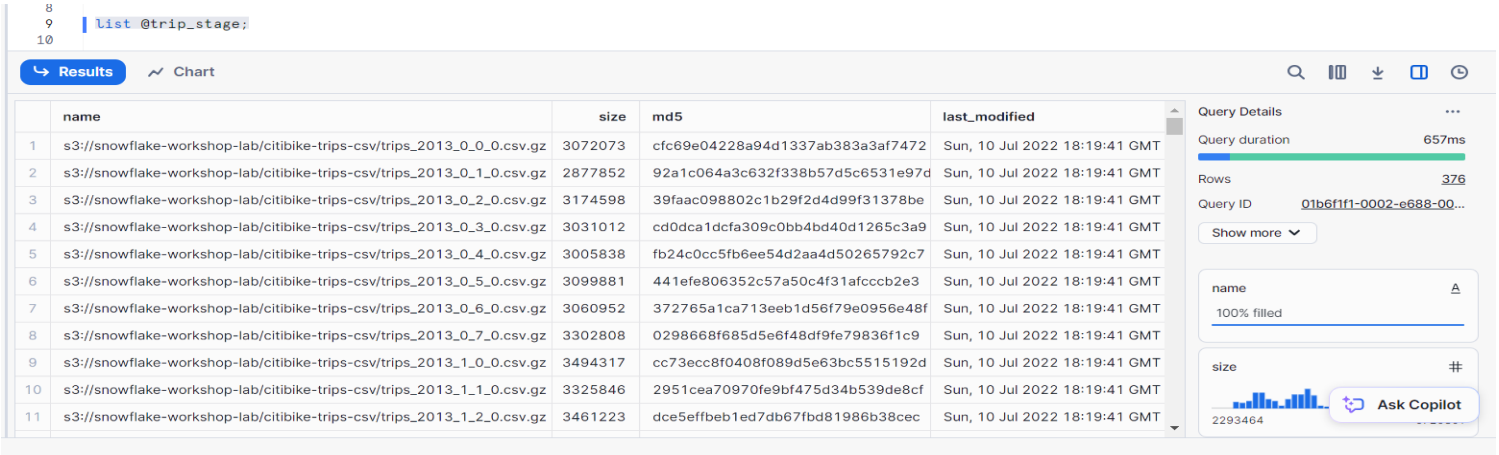
  create stage trip_stage url='s3://snowflake-workshop-lab/citibike-trips-csv/';

Analyzing Trip Data in Snowflake



- Query to list the stage

  list @trip_stage;

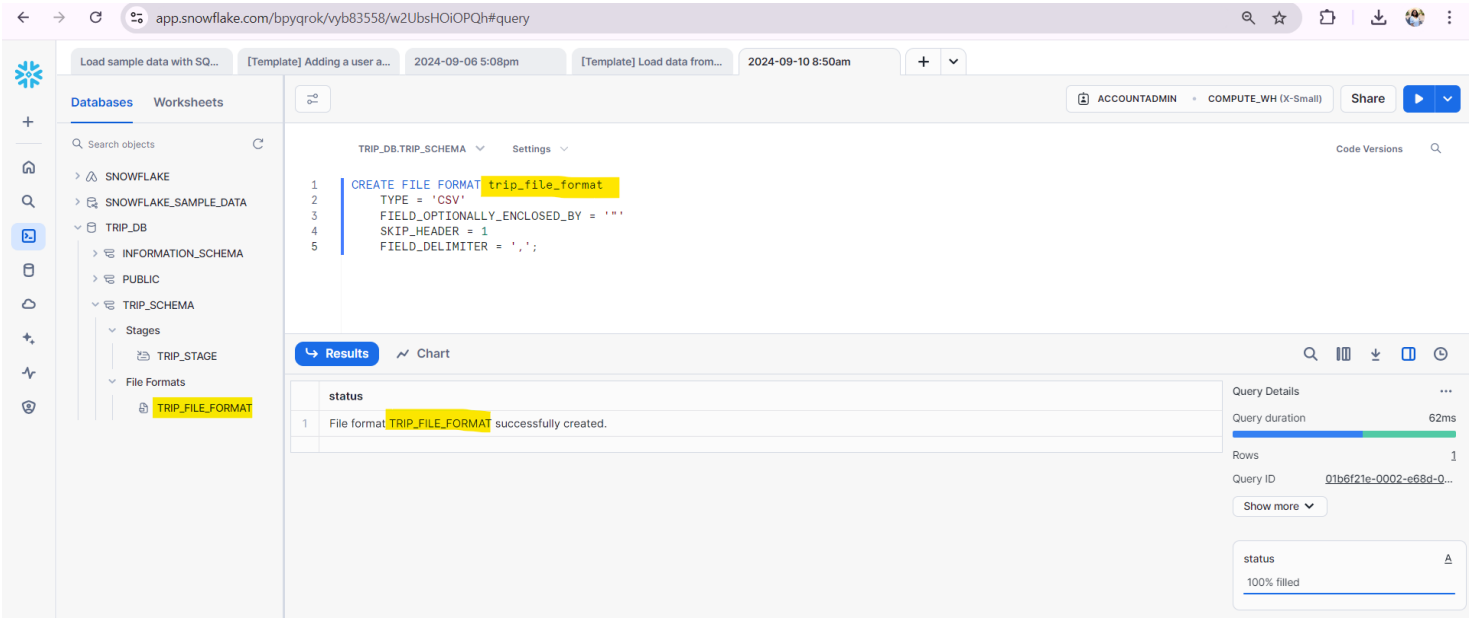


3. Create File Format

Create a file format to read data from CSV file.

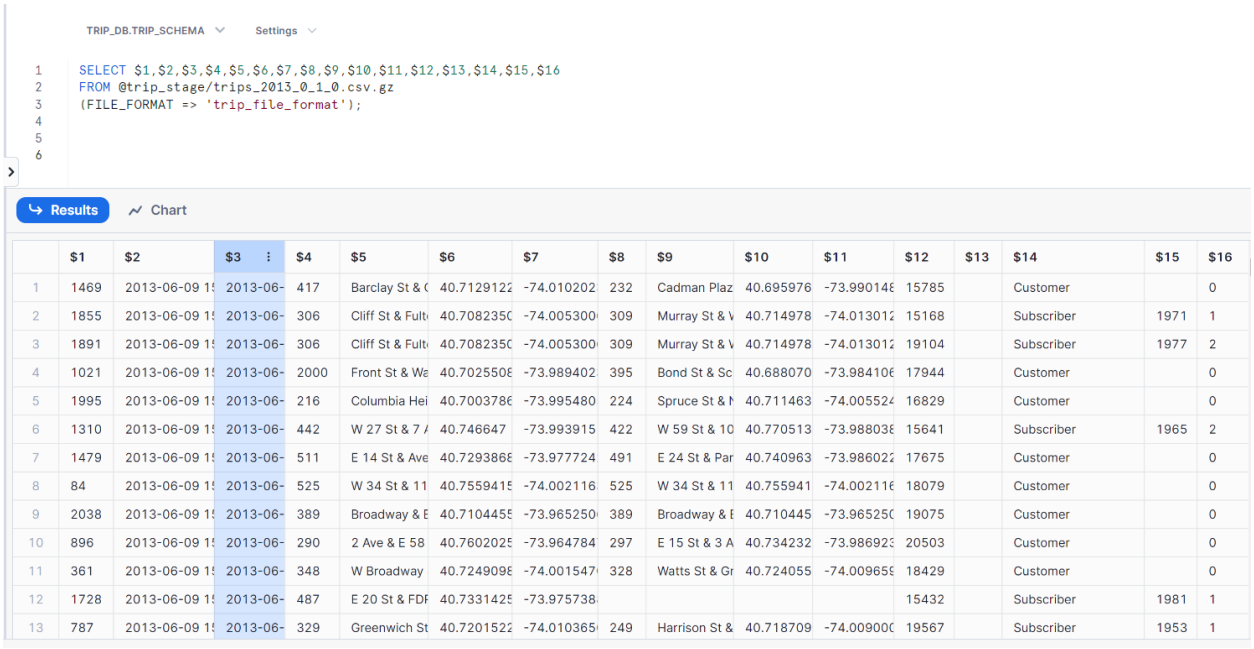
- Query to create file format
  CREATE FILE FORMAT TRIP_FILE_FORMAT TYPE = 'CSV'
  FIELD_OPTIONALLY_ENCLOSED_BY = '"'
  SKIP_HEADER = 1
  FIELD_DELIMITER = ',' ;

Analyzing Trip Data in Snowflake



- Query to view the data in file using file format.

SELECT $1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,$15,$16
FROM @trip_stage/trips_2013_0_1_0.csv.gz
(FILE_FORMAT => 'trip_file_format');



	$1	$2	$3	$4	$5	$6	$7	$8	$9	$10	$11	$12	$13	$14	$15	$16
1	1469	2013-06-09 1	2013-06-	417	Barclay St & (	40.7129122	-74.010202	232	Cadman Plaz	40.695976	-73.990148	15785		Customer		0
2	1855	2013-06-09 1	2013-06-	306	Cliff St & Fult	40.708235(	-74.005300	309	Murray St & V	40.714978	-74.013012	15168		Subscriber	1971	1
3	1891	2013-06-09 1	2013-06-	306	Cliff St & Fult	40.708235(	-74.005300	309	Murray St & V	40.714978	-74.013012	19104		Subscriber	1977	2
4	1021	2013-06-09 1	2013-06-	2000	Front St & We	40.7025508	-73.989402	395	Bond St & Sc	40.688070	-73.984106	17944		Customer		0
5	1995	2013-06-09 1	2013-06-	216	Columbia Hei	40.7003786	-73.995480	224	Spruce St & M	40.711463	-74.005524	16829		Customer		0
6	1310	2013-06-09 1	2013-06-	442	W 27 St & 7 A	40.746647	-73.993915	422	W 59 St & 10	40.770513	-73.988038	15641		Subscriber	1965	2
7	1479	2013-06-09 1	2013-06-	511	E 14 St & Ave	40.7293868	-73.977724	491	E 24 St & Par	40.740963	-73.986022	17675		Customer		0
8	84	2013-06-09 1	2013-06-	525	W 34 St & 11	40.7559415	-74.002116	525	W 34 St & 11	40.755941	-74.002116	18079		Customer		0
9	2038	2013-06-09 1	2013-06-	389	Broadway & E	40.7104455	-73.965250	389	Broadway & E	40.710445	-73.965250	19075		Customer		0
10	896	2013-06-09 1	2013-06-	290	2 Ave & E 58	40.7602025	-73.964784	297	E 15 St & 3 A	40.734232	-73.986923	20503		Customer		0
11	361	2013-06-09 1	2013-06-	348	W Broadway	40.7249098	-74.001547	328	Watts St & Gr	40.724055	-74.009659	18429		Customer		0
12	1728	2013-06-09 1	2013-06-	487	E 20 St & FDF	40.7331425	-73.975738					15432		Subscriber	1981	1
13	787	2013-06-09 1	2013-06-	329	Greenwich St	40.7201522	-74.010365	249	Harrison St &	40.718709	-74.009000	19567		Subscriber	1953	1

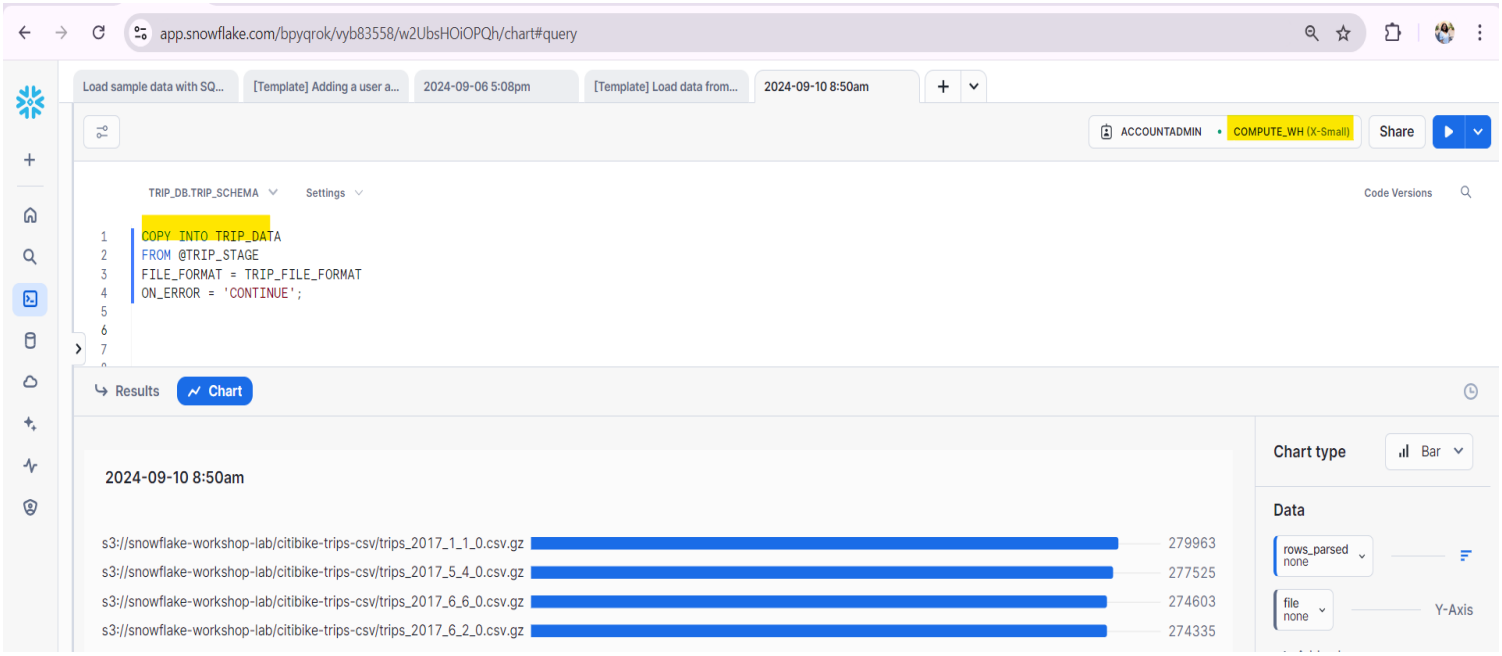
Analyzing Trip Data in Snowflake

4. Load Data into the Table

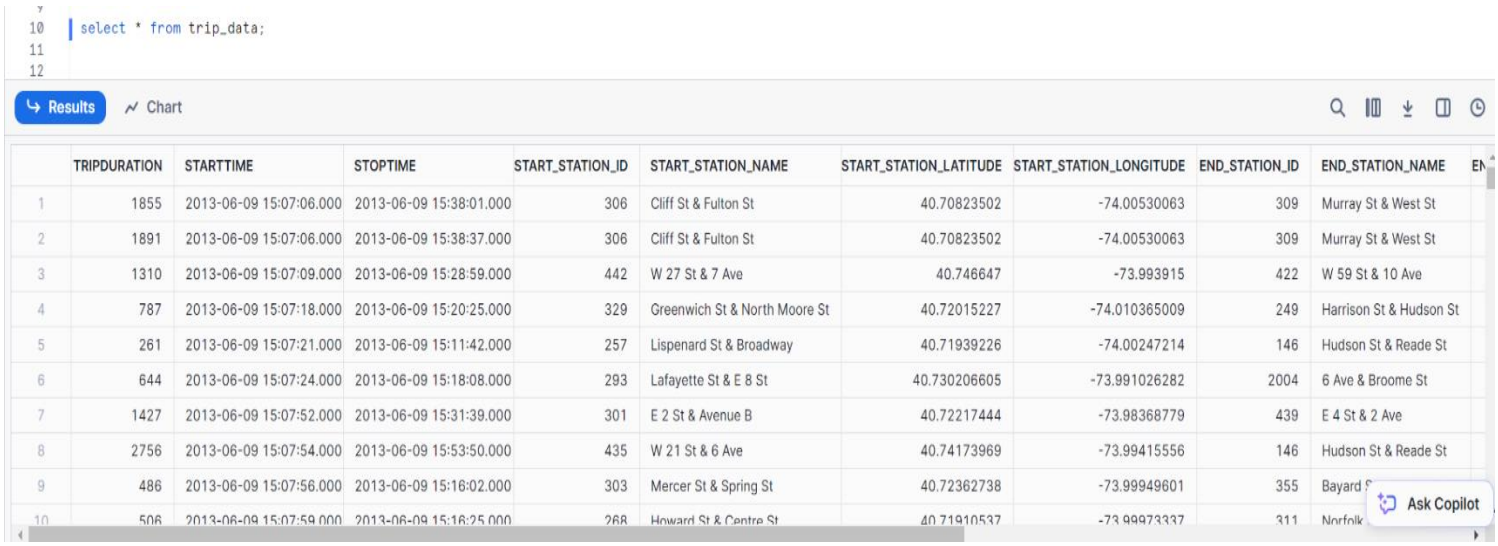
After data file is uploaded to the stage, you can now load it into the trip_data table:

- Query to copy data into table

```
COPY INTO TRIP_DATA
 FROM @TRIP_STAGE
 FILE_FORMAT = TRIP_FILE_FORMAT
 ON_ERROR = 'CONTINUE';
```



select * from trip_data;



	TRIPDURATION	STARTTIME	STOPTIME	START_STATION_ID	START_STATION_NAME	START_STATION_LATITUDE	START_STATION_LONGITUDE	END_STATION_ID	END_STATION_NAME
1	1855	2013-06-09 15:07:06.000	2013-06-09 15:38:01.000	306	Cliff St & Fulton St	40.70823502	-74.00530063	309	Murray St & West St
2	1891	2013-06-09 15:07:06.000	2013-06-09 15:38:37.000	306	Cliff St & Fulton St	40.70823502	-74.00530063	309	Murray St & West St
3	1310	2013-06-09 15:07:09.000	2013-06-09 15:28:59.000	442	W 27 St & 7 Ave	40.746647	-73.993915	422	W 59 St & 10 Ave
4	787	2013-06-09 15:07:18.000	2013-06-09 15:20:25.000	329	Greenwich St & North Moore St	40.72015227	-74.010365009	249	Harrison St & Hudson St
5	261	2013-06-09 15:07:21.000	2013-06-09 15:11:42.000	257	Lispenard St & Broadway	40.71939226	-74.00247214	146	Hudson St & Reade St
6	644	2013-06-09 15:07:24.000	2013-06-09 15:18:08.000	293	Lafayette St & E 8 St	40.730206605	-73.991026282	2004	6 Ave & Broome St
7	1427	2013-06-09 15:07:52.000	2013-06-09 15:31:39.000	301	E 2 St & Avenue B	40.72217444	-73.98368779	439	E 4 St & 2 Ave
8	2756	2013-06-09 15:07:54.000	2013-06-09 15:53:50.000	435	W 21 St & 6 Ave	40.74173969	-73.99415556	146	Hudson St & Reade St
9	486	2013-06-09 15:07:56.000	2013-06-09 15:16:02.000	303	Mercer St & Spring St	40.72362738	-73.99949601	355	Bayard S
10	506	2013-06-09 15:07:59.000	2013-06-09 15:16:25.000	268	Howard St & Centre St	40.71910537	-73.99973337	311	Norfolk

Analyzing Trip Data in Snowflake

5.Write an SQL query to produce a report that shows, for each hour of the day, the following:

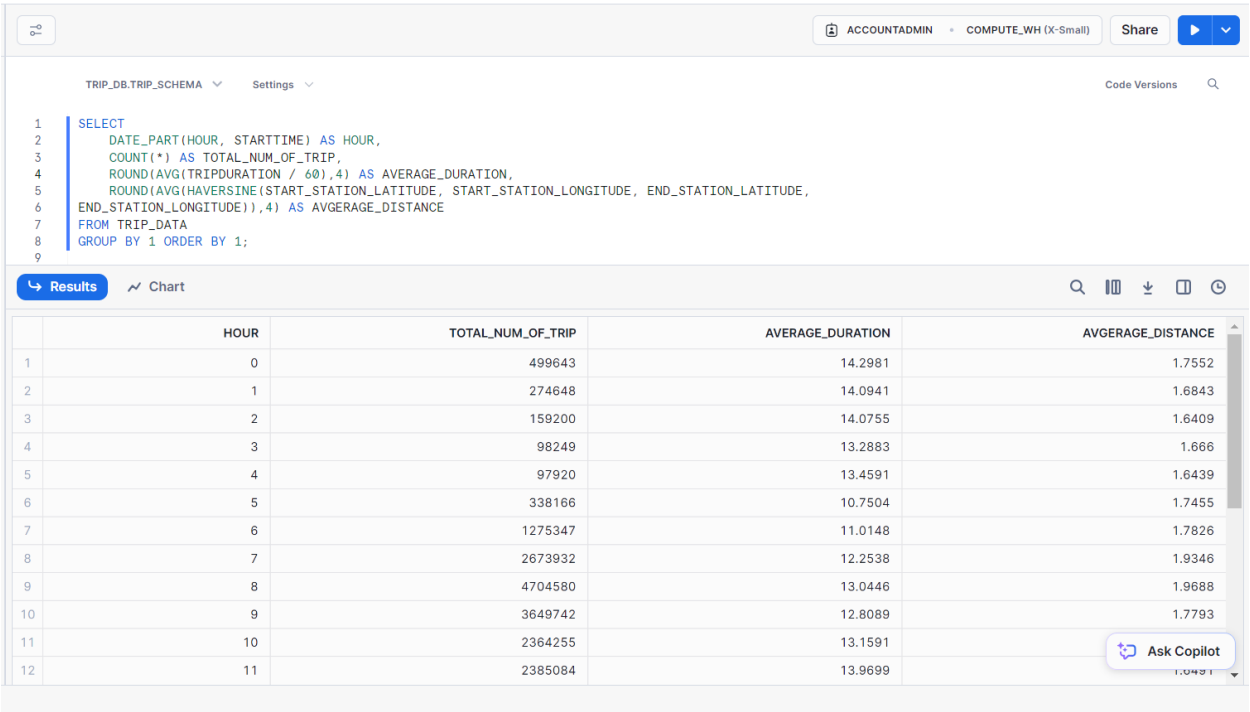
1. The total number of trips that started during that hour.
2. The average duration of these trips in mins.
3. The average distance traveled during these trips in kms.

- SQL query

```
SELECT
  DATE_PART(HOUR, STARTTIME) AS HOUR,
   COUNT(*) AS TOTAL_NUM_OF_TRIP,
   ROUND(AVG(TRIPDURATION / 60),4) AS AVERAGE_DURATION,
   ROUND(AVG(HAVERSINE(START_STATION_LATITUDE,
START_STATION_LONGITUDE, END_STATION_LATITUDE,
END_STATION_LONGITUDE)),4) AS AVGERAGE_DISTANCE
FROM TRIP_DATA
GROUP BY 1 ORDER BY 1;
```

Explanation:

- DATE_PART(HOUR, STARTTIME) extracts the hour from the STARTTIME timestamp.
- COUNT(*) gives the total number of trips for each hour.
- AVG(TRIPDURATION / 60.0) calculates the average trip duration in minutes (assuming TRIPDURATION is stored in seconds).
- HAVERSINE function calculates the distance between the start and end coordinates of each trip using the Haversine formula.
- ROUND() round the number up to the given number, in this case it rounds till 4 places.



	HOUR	TOTAL_NUM_OF_TRIP	AVERAGE_DURATION	AVGERAGE_DISTANCE
1	0	499643	14.2981	1.7552
2	1	274648	14.0941	1.6843
3	2	159200	14.0755	1.6409
4	3	98249	13.2883	1.666
5	4	97920	13.4591	1.6439
6	5	338166	10.7504	1.7455
7	6	1275347	11.0148	1.7826
8	7	2673932	12.2538	1.9346
9	8	4704580	13.0446	1.9688
10	9	3649742	12.8089	1.7793
11	10	2364255	13.1591	
12	11	2385084	13.9699	

Analyzing Trip Data in Snowflake

Data Result

HOUR	TOTAL_NUM_OF_TRIP	AVERAGE_DURATION	AVERAGE_DISTANCE
0	499643	14.2981	1.7552
1	274648	14.0941	1.6843
2	159200	14.0755	1.6409
3	98249	13.2883	1.666
4	97920	13.4591	1.6439
5	338166	10.7504	1.7455
6	1275347	11.0148	1.7826
7	2673932	12.2538	1.9346
8	4704580	13.0446	1.9688
9	3649742	12.8089	1.7793
10	2364255	13.1591	1.6973
11	2385084	13.9699	1.6491
12	2771192	13.974	1.6038
13	2899224	14.0809	1.5985
14	2969842	14.7	1.6678
15	3130067	14.9695	1.7113
16	3767896	14.8513	1.7962
17	5457805	14.7013	1.9442
18	5295007	14.7997	1.9237
19	3675781	14.5261	1.7884
20	2461409	14.5986	1.737
21	1716035	14.4014	1.7365
22	1283920	14.5091	1.7496
23	854615	14.4283	1.7302

6. Create X-LARGE Warehouse
Now, create a new warehouse TRIP_XLARGE_WH of size X-LARGE.

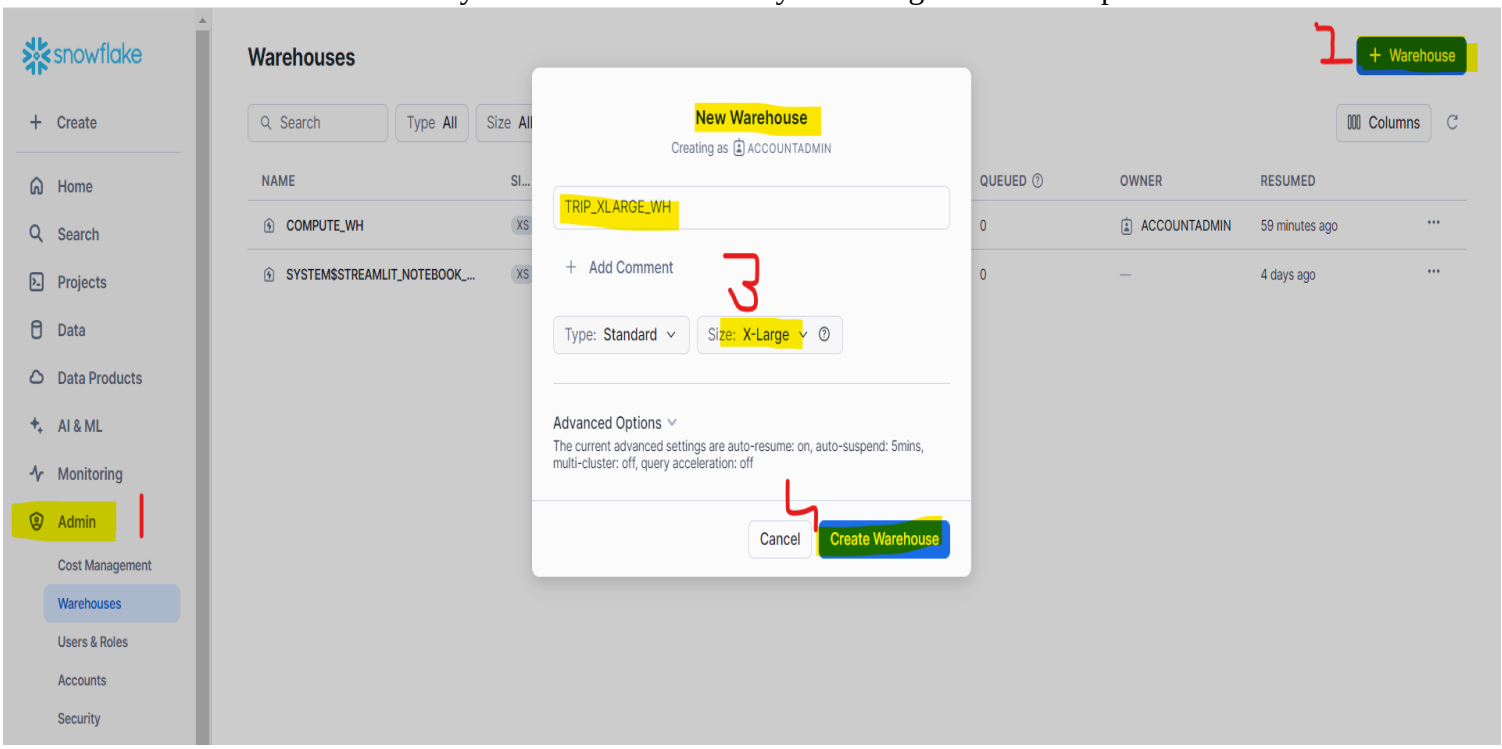
- SQL query to create X-LARGE WH

```
CREATE WAREHOUSE TRIP_XLARGE_WH
 WAREHOUSE_SIZE = 'XLARGE'
AUTO_SUSPEND = 300
AUTO_RESUME = TRUE;
```

Or

Analyzing Trip Data in Snowflake

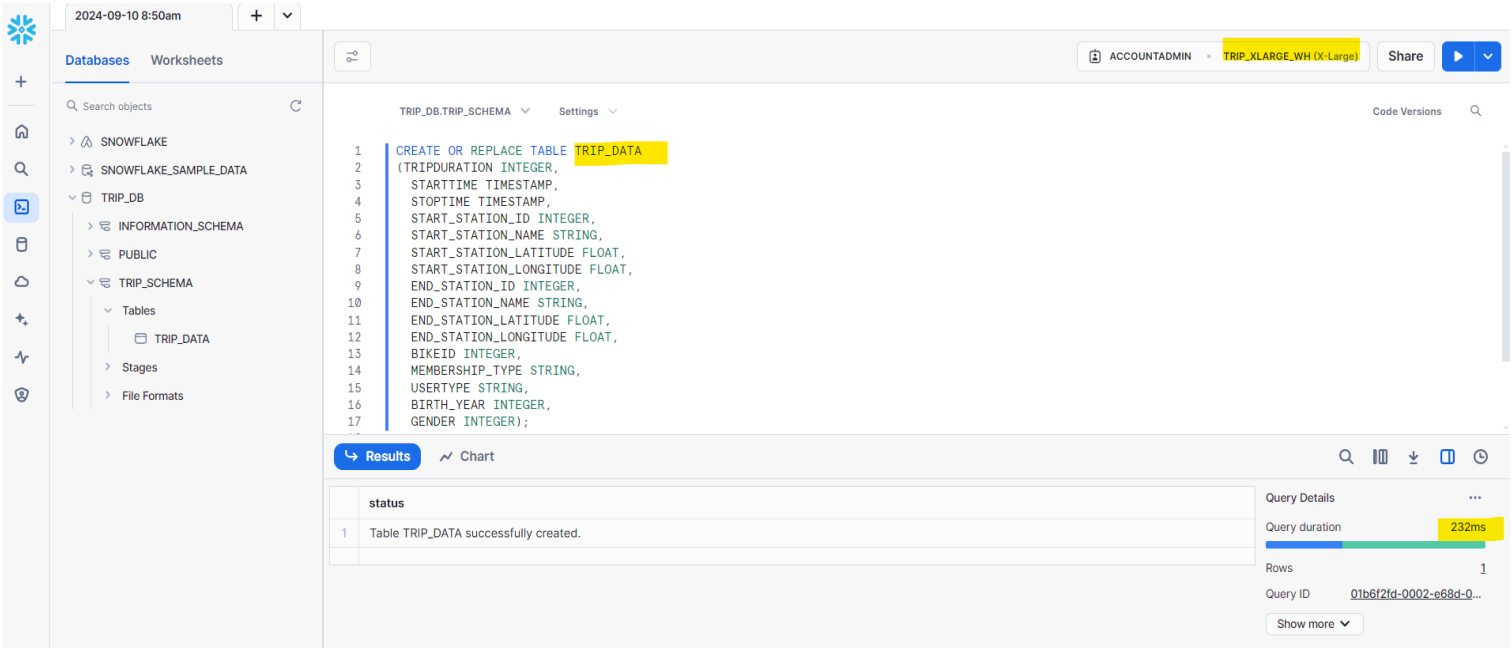
We can manually create the warehouse by following the below steps.



After dropping and recreating the data table, we will repeat steps 4 and 5 to analyze the performance of both the X-Small and X-Large warehouses.

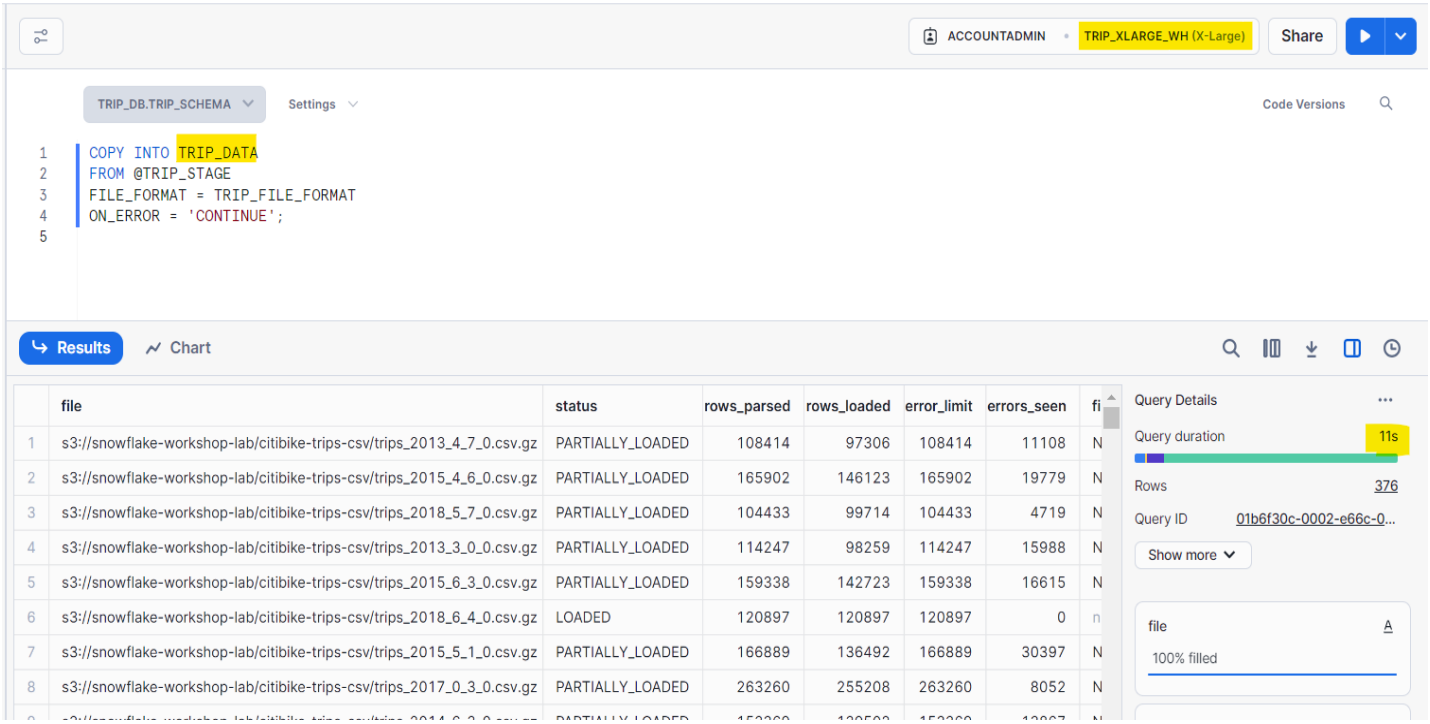
drop table trip_data;	01b6f2fc-0002-e67c-0000-l	Success	SHWETASHINDE

4. Create trip_data using X-Large Warehouse



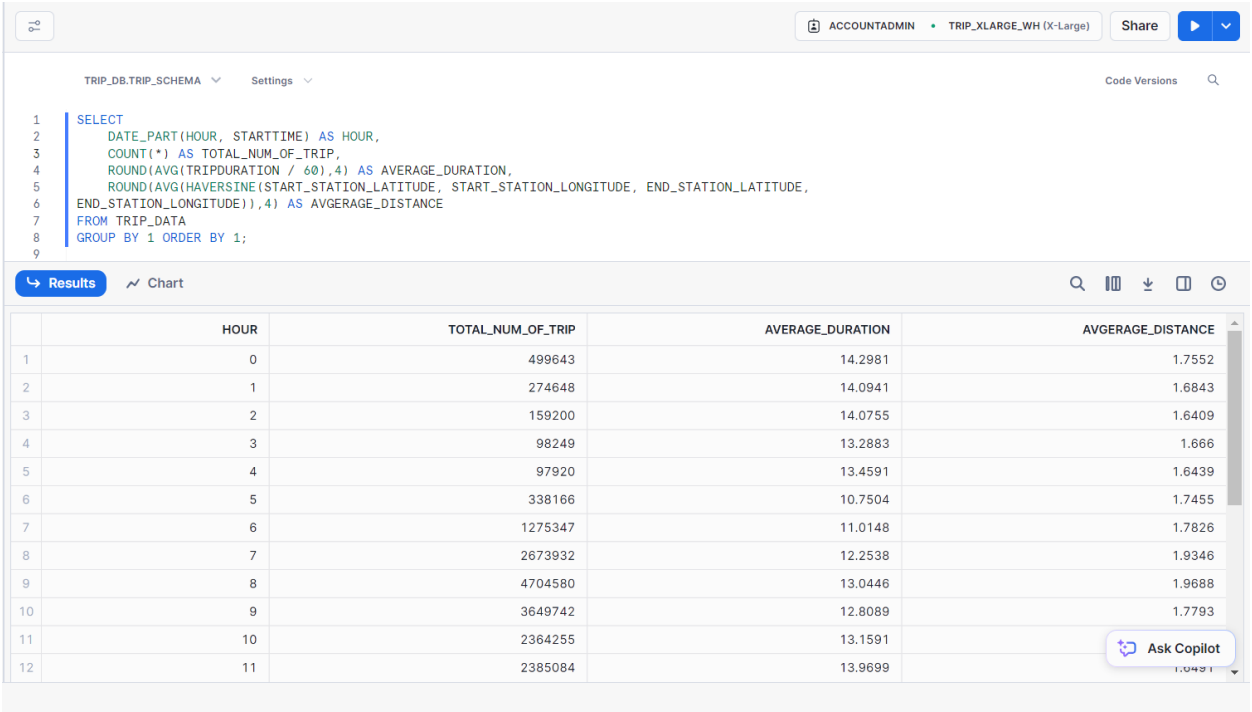
Analyzing Trip Data in Snowflake

We are load data in the table using X-Large Warehouse



	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	fi
1	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2013_4_7_0.csv.gz	PARTIALLY_LOADED	108414	97306	108414	11108	N
2	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2015_4_6_0.csv.gz	PARTIALLY_LOADED	165902	146123	165902	19779	N
3	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2018_5_7_0.csv.gz	PARTIALLY_LOADED	104433	99714	104433	4719	N
4	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2013_3_0_0.csv.gz	PARTIALLY_LOADED	114247	98259	114247	15988	N
5	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2015_6_3_0.csv.gz	PARTIALLY_LOADED	159338	142723	159338	16615	N
6	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2018_6_4_0.csv.gz	LOADED	120897	120897	120897	0	n
7	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2015_5_1_0.csv.gz	PARTIALLY_LOADED	166889	136492	166889	30397	N
8	s3://snowflake-workshop-lab/citibike-trips-csv/trips_2017_0_3_0.csv.gz	PARTIALLY_LOADED	263260	255208	263260	8052	N

5.We will run the SQL query to produce a report that shows, for each hour of the day using X-Large Warehouse



	HOUR	TOTAL_NUM_OF_TRIP	AVERAGE_DURATION	AVGERAGE_DISTANCE
1	0	499643	14.2981	1.7552
2	1	274648	14.0941	1.6843
3	2	159200	14.0755	1.6409
4	3	98249	13.2883	1.666
5	4	97920	13.4591	1.6439
6	5	338166	10.7504	1.7455
7	6	1275347	11.0148	1.7826
8	7	2673932	12.2538	1.9346
9	8	4704580	13.0446	1.9688
10	9	3649742	12.8089	1.7793
11	10	2364255	13.1591	
12	11	2385084	13.9699	

Analyzing Trip Data in Snowflake

Analyze the performance of the node upon changing configurations from X-Small to X-Large in Snowflake.

In the analysis, we observed that the X-Small warehouse took longer to execute the query compared to the X-Large warehouse. Specifically:

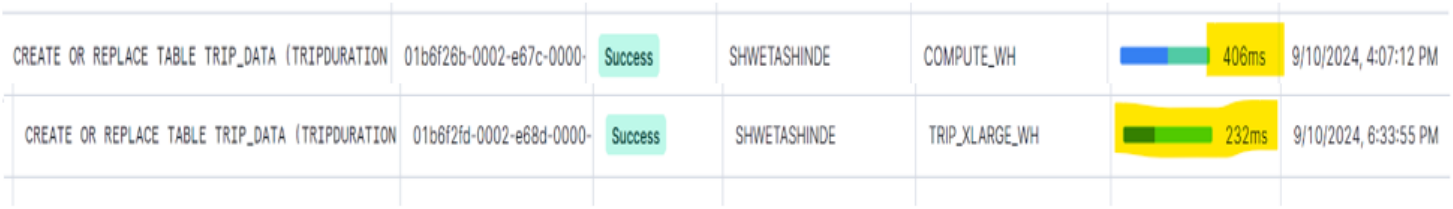
- The **X-Large** Warehouse (*TRIP_XLARGE_WH*) delivers much faster query completion, indicating a higher level of performance and capacity compared to the X-Small Warehouse. The X-Large Warehouse offers **better performance,** but this comes with a **higher cost**. Its advanced capabilities are reflected in the expense of using this option.
- In contrast, the **X-Small** Warehouse (*COMPUTE_WH*) demonstrates **slower performance**, taking more time to execute similar operations. This slower speed suggests potential limitations in handling large volumes of data or complex queries effectively. The X-Small Warehouse is more **budget-friendly**, but it comes with slower performance. This lower cost may result in reduced efficiency and longer processing times.

Please find below the screenshots comparing both Warehouses in terms of Efficiency and cost.

<u>Analysis of Efficiency:-</u>

- **CREATE** table TRIP_DATA

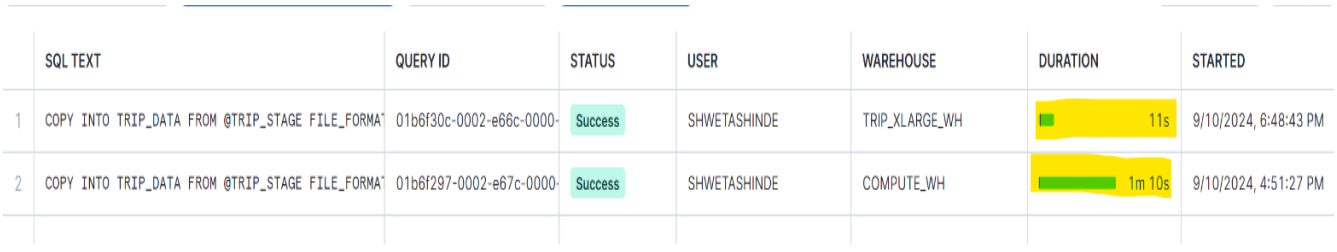
The **X-Large** Warehouse took 232ms to create the table, while the **X-Small** Warehouse took 406ms for the same task.



CREATE OR REPLACE TABLE TRIP_DATA (TRIPDURATION	01b6f26b-0002-e67c-0000-	Success	SHWETASHINDE	COMPUTE_WH	406ms	9/10/2024, 4:07:12 PM
CREATE OR REPLACE TABLE TRIP_DATA (TRIPDURATION	01b6f2fd-0002-e68d-0000-	Success	SHWETASHINDE	TRIP_XLARGE_WH	232ms	9/10/2024, 6:33:55 PM

- **COPY** query to load data into TRIP_DATA

The **X-Large** Warehouse took 11 seconds to load 54,803,559 rows into the table, while the **X-Small** Warehouse took 1 minute and 10 seconds for the same task.

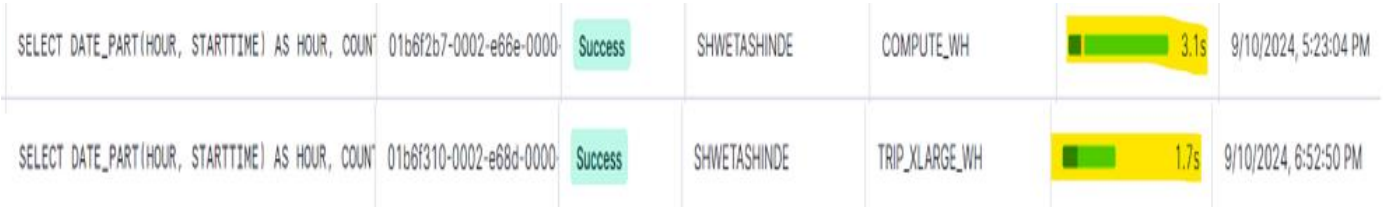


	SQL TEXT	QUERY ID	STATUS	USER	WAREHOUSE	DURATION	STARTED
1	COPY INTO TRIP_DATA FROM @TRIP_STAGE FILE_FORMA	01b6f30c-0002-e66c-0000-	Success	SHWETASHINDE	TRIP_XLARGE_WH	11s	9/10/2024, 6:48:43 PM
2	COPY INTO TRIP_DATA FROM @TRIP_STAGE FILE_FORMA	01b6f297-0002-e67c-0000-	Success	SHWETASHINDE	COMPUTE_WH	1m 10s	9/10/2024, 4:51:27 PM

Analyzing Trip Data in Snowflake

- **SELECT** query on TRIP_DATA

The **X-Large** Warehouse took 1.7 seconds to select data from table , while the **X-Small** Warehouse took 3.1 seconds for the same task.



Analysis of Cost:-

- Cost of both the Warehouse

The **X-Large** Warehouse has an expense of $5.25 to execute 3 to 4 queries , while the **X-Small** Warehouse costs $1.92 for handling more than 10 queries.

Conclusion

For much larger datasets, the X-Large Warehouse will likely perform even better. It will take less time to load data compared to the X-Small Warehouse. As the amount of data grows, the time difference between the two warehouses will increase. The X-Large Warehouse will handle large tasks more efficiently, significantly cutting down the time needed for loading and processing data.

While the X-Large Warehouse comes with a higher cost, its performance justifies the expense for handling large-scale operations. In contrast, the X-Small Warehouse is more budget-friendly, making it ideal for smaller datasets or less complex tasks where high performance is less critical. It is cost-effective for development, testing, small-scale operations or small dataset.