Experiment: Data Engineering Pipeline

Overview

This experiment demonstrates Snowflake features specifically aligned to “Data Engineering” workload to build modern data pipelines. Data pipelines automate many of the manual steps involved in transforming and optimizing continuous data loads. Frequently, the “raw” data is firstloaded temporarily into a staging table used for interim storage and then transformed using a series of SQL statements before it is inserted into the destination reporting tables. The most efficient workflow for this process involves transforming only data that is new or modified.

Experiment 23: Streams

23.1 Creating Trips and Stations Streams on the Raw Data

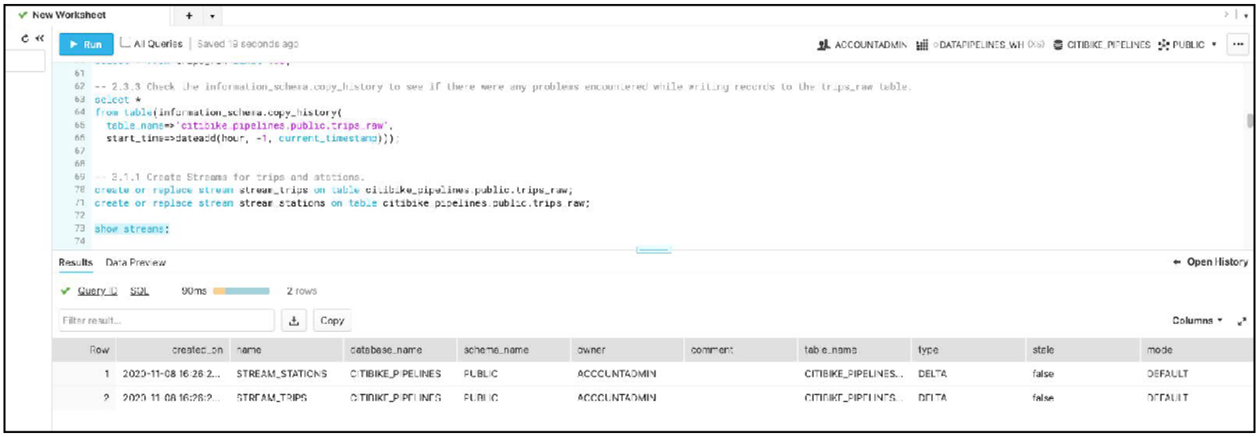
23.1.1 Having built continuous data loading using Snowpipe, the next step is to create multiple Streams on the trips_raw table to track new trips and stations records. A stream object records the delta of change data capture (CDC) information for a table (such as a staging table), including inserts and other data manipulation language (DML) changes.

In a continuous data pipeline, table streams record when staging tables and any downstream tables are populated with data from business applications using continuous data loading and are ready for further processing using SQL statements

Streams provide a convenient way to ensure that you only process new records in the table each time.

```
create or replace stream stream_trips
on table
citibike_pipelines.public.trips_raw;
create or replace stream stream_stations
on table
citibike_pipelines.public.trips_raw;
```

```
show streams;
```



```
-- 2.3.3 Check the information_schema.copy_history to see if there were any problems encountered while writing records to the trips_raw table.
select *
from table(information_schema.copy_history(
  table_name=>'citibike_pipelines.public.trips_raw',
  start_time=>dateadd(hour, -1, current_timestamp)));


-- 3.1.1 Create Streams for trips and stations.
create or replace stream stream_trips on table citibike_pipelines.public.trips_raw;
create or replace stream stream_stations on table citibike_pipelines.public.trips_raw;

show streams;
```

Row	created_on	name	database_name	schema_name	owner	comment	table_name	type	stale	mode
1	2020-11-08 16:26:2...	STREAM_STATIONS	CITIBIKE_PIPELINES	PUBLIC	ACCOUNTADMIN		CITIBIKE_PIPELINES...	DELTA	false	DEFAULT
2	2020-11-08 16:26:2...	STREAM_TRIPS	CITIBIKE_PIPELINES	PUBLIC	ACCOUNTADMIN		CITIBIKE_PIPELINES...	DELTA	false	DEFAULT

23.1.2 Load 1 day of data to test the streams.

```
call stream_data('2018-01-02', '2018-01-02');
```

23.1.3 Show the contents of the stage

```
list @streaming_data;
select $1 from @streaming_data limit 100;
```

23.1.4 Check the status of the pipe and watch for file create events

```
select system$pipe_status('trips_pipe');
```

23.1.5 Snowpipe copies the data into our raw table…

```
select count(*) from
citibike_pipelines.public.trips_raw; select *
from citibike_pipelines.public.trips_raw limit
100;
```

and the insertions are tracked in the stream.

```
select count(*) from
stream_trips; select * from
stream_trips limit 100;
```

23.1.6 Clean up the stage by calling the purge_files function.

```
call purge_files('trips_raw', '@streaming_data/');
```

23.1.7 The data engineering process will process the streaming trips data, so next create tables to store those results.

```
create or replace table
  bike_trips (tripduration
  integer,
  starttime
  timestamp_ntz,
  stoptime
  timestamp_ntz,
  start_station_id
  integer,
  end_station_id
  integer, bikeid
  integer,
  usertype string
);

create or replace table
  bike_stations (station_id
  integer,
  station_name
  string,
  station_latitude
  float,
  station_longitude
  float,
  station_comment
  string
);
```

The process flow will be:
Stream stream_trips -> table bike_trips (convert to structured, append new records)
Stream stream_stations -> table bike_stations (build a dimension table, merge new start and end stations)