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 25: Pipeline Validation

That’s it! Let’s kick this off and observe the data now flowing through the pipeline end to end for a couple of days worth of data..

25.1 Data Pipeline Demonstration

25.1.1 Call the stream_data procedure to drop files into the STAGE which will then get auto loaded into the raw.trips_raw table. Here we’ll load a couple of days worth of data.

```
call stream_data('2018-01-03', '2018-01-04');
```

25.1.2 The following command will show us the details of each of our tasks over the last five minutes. You will notice the STATE for some of these show ‘SKIPPED’ because no data has arrived in the STREAM.

```
select * from
  table(information_schema.task_history()) where
  scheduled_time > dateadd(minute, -5,
  current_time()) and state <> 'SCHEDULED'
  order by completed_time desc;
```

25.1.3 How long until the next task runs?

```
select timestampdiff(second, current_timestamp,
scheduled_time) next_run, scheduled_time, name, state
  from table(information_schema.task_history())
  where state = 'SCHEDULED' order by completed_time desc;
```

25.1.4 How many files have been processed by the pipeline in the last hour?

```
select count(*)
```

```
from table(information_schema.copy_history(
  table_name=>'citibike_pipelines.public.tri
  ps_raw',start_time=>dateadd(hour, -1,
  current_timestamp)));
```

25.1.5 Here's a convenient query to get an overview of the pipeline including: time to next task run, the number of files in the bucket, number of files pending for loading, total number of files processed in the last hour, and record count metrics across all the tables referenced in the solution.

```
select
  (select min(timestampdiff(second, current_timestamp,
    scheduled_time)) from
    table(information_schema.task_history())
    where state = 'SCHEDULED' order by completed_time desc)
    time_to_next_pulse,
  (select count(distinct metadata$filename) from
@streaming_data/) files_in_bucket,
  (select
parse_json(system$pipe_status('citibike_pipelines.public.trips_pi
pe')):pendingFileCount::number) pending_file_count,
  (select count(*)
    from table(information_schema.copy_history(
    table_name=>'citibike_pipelines.public.trips_raw',
    start_time=>dateadd(hour, -1, current_timestamp))))
    files_processed,
  (select count(*) from citibike_pipelines.public.trips_raw)
  trips_raw, (select count(*) from
  citibike_pipelines.public.stream_trips) recs_in_stream,(select
  count(*) from citibike_pipelines.public.bike_trips)
  bike_trips, (select count(*) from
  citibike_pipelines.public.bike_stations) bike_stations,(select
  max(starttime) from citibike_pipelines.public.trips) max_date;
```

25.1.6 Suspend the tasks using the below queries.

```
alter task push_stations
suspend;

alter task push_trips
suspend;

alter task purge_files
suspend;
```

Congratulations, you are now done with this lab! Let's wrap things up in the next, and final, section.

Summary & Next Steps

This lab has walked you step by step through the elements that comprise data engineering using Snowflake data pipelines. You created a Stage object and configured it to invoke Snowpipe when files arrive. This allows for automatic ingestion of streaming data sources. Next you created streams on top of the staging table that Snowpipe loads data into. Streams provide a way to continually process data as it arrives but only focus that processing on new data and avoid reprocessing old data. Lastly, you created tasks for scheduled execution of logic to move data from the streams into the final tables of the overall data model. Along the way you also created handy procedures to mimic streaming data and clean up data files that were loaded.

It's easy to see how powerful these tools are and how you can apply them to workloads in your Snowflake account to achieve automated data engineering.

25.2 Resetting Your Snowflake Environment

Lastly, if you would like to reset your environment by deleting all the objects created as part of this lab, run the SQL below in a worksheet (ModernDataPipelines_Experiment Cleanup.sql).

```
-- Switch Context
USE ROLE ACCOUNTADMIN;
USE CITIBIKE_PIPELINES.PUBLIC;


USE WAREHOUSE DATAPIPELINES_WH;

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

drop database if exists CITIBIKE_PIPELINES;
drop warehouse if exists DATAPIPELINES_WH;
```

In AWS - delete the S3 bucket/subfolder, IAM role and IAM policies created as part of this lab.