# Experiment: Data Engineering Pipeline

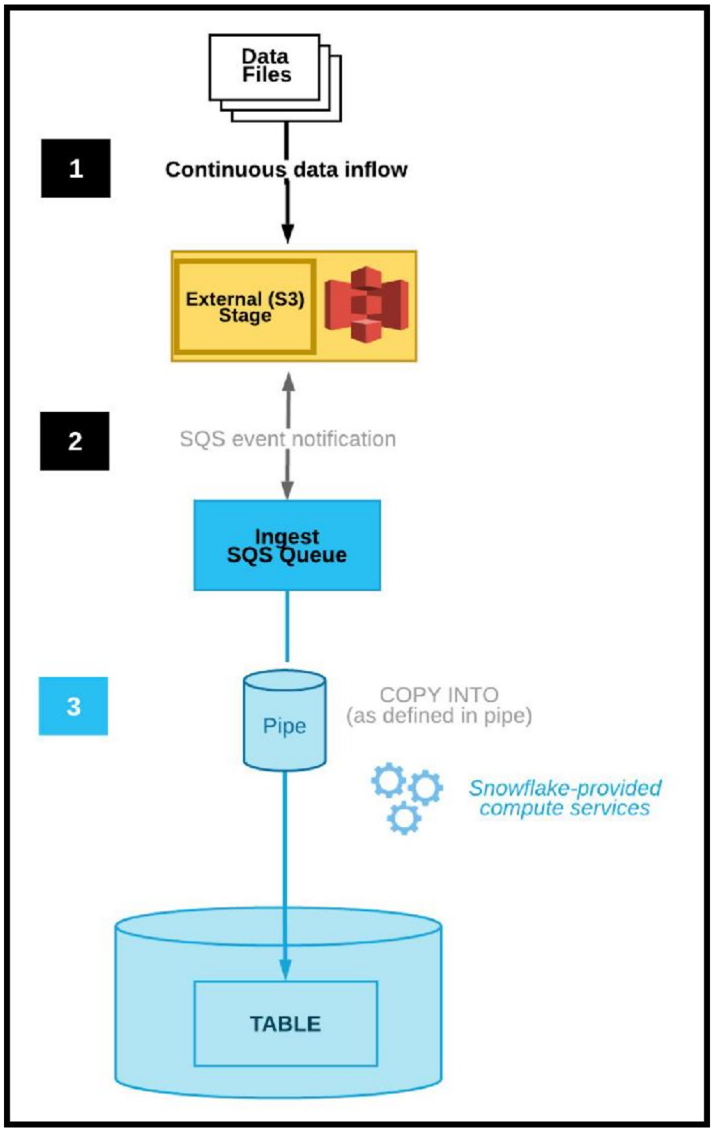
## Overview

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 22: Snowpipe

### 22.1 Leverage Snowpipe to Populate Table using S3 Event Notifications

This section describes the most common option for triggering Snowpipe data loads automatically using Amazon SQS (Simple Queue Service) notifications for an S3 bucket. The following diagram shows the Snowpipe auto-ingest process flow:



1. Data files are loaded in a stage.
2. An S3 event notification informs Snowpipe via an SQS queue that files are ready to load. Snowpipe copies the files into a queue.
3. A Snowflake-provided virtual warehouse loads data from the queued files into the target table based on parameters defined in the specified pipe.

22.1.1 Create a table that Snowpipe will use to write the incoming data. The data being loaded is JSON formatted so create a table using the VARIANT data type.

```
create or replace table trips_raw (v variant);
```

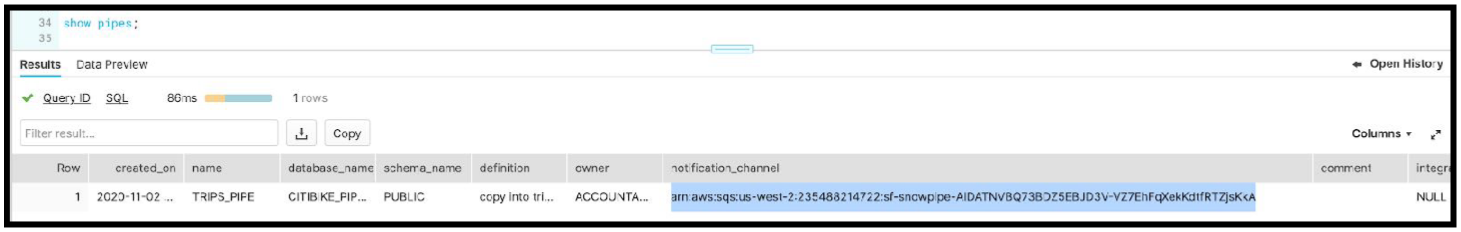
22.1.2 Create a PIPE definition to tell Snowpipe to load the JSON records in the STAGE into the table you just created.

```
create or replace pipe trips_pipe auto_ingest=true as copy into
trips_raw from@streaming_data/;
```

22.1.3 Confirm PIPE definition to tell Snowpipe to load the JSON records in the STAGE into the table you just created.

```
show pipes;
```

Make a note of the ARN of the SQS queue for the stage in the *notification_channel* column. Copy the ARN to a convenient location.



```
show pipes;
```

Results | Data Preview | Open History

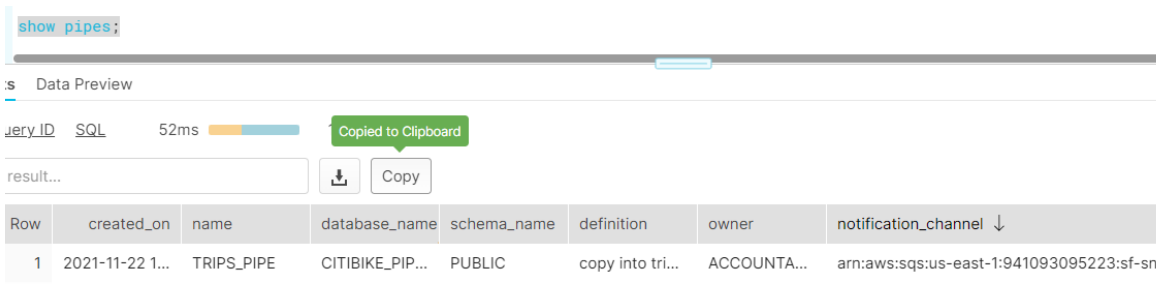
Query ID | SQL | 86ms | 1 rows

| Row | created_on | name | database_name | schema_name | definition | owner | notification_channel | comment | integr |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2020-11-02 ... | TRIPS_PIPE | CITIBIKE_PIP... | PUBLIC | copy into tri... | ACCOUNTA... | arn:aws:sqs:us-west-2:235488214722:sf-snowpipe-AIDATNVBQ73BDZ5EBJD3V-VZ7EhFqXekKdtfRTZjsKkA | | NULL |

22.1.4 Configuring Event Notifications. Complete the below configuration in your S3 bucket to automatically invoke Snowpipe when files are written there. This is the mechanism that is used to automatically load the streaming data files. Snowpipe relies on notifications from your S3 bucket to achieve this. Instructions for completing this configuration are found in the Snowflake documentation link below. ***Perform only Step 4- Configure Event Notifications and come back to this document***.

https://docs.snowflake.com/en/user-guide/data-load-snowpipe-auto-s3.html#step-4-configure-event-notifications

From the Snowflake console when we execute "show pipes" it will provide the SQS queue that is the notification channel.



```
show pipes;
```

Data Preview

Query ID | SQL | 52ms | Copied to Clipboard

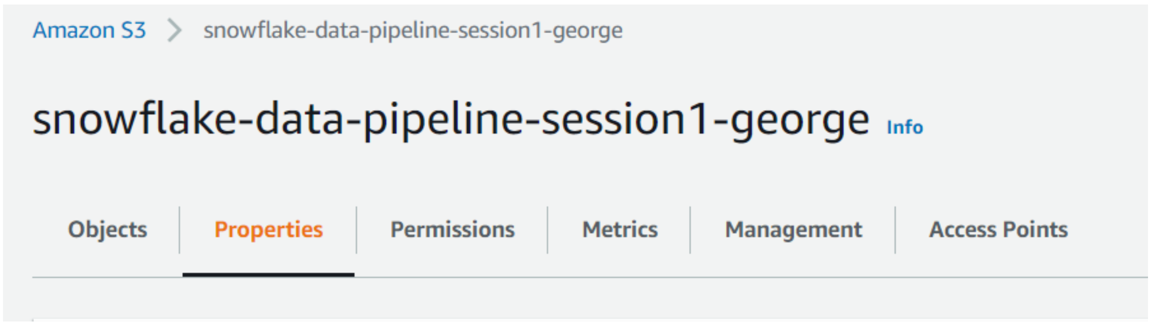
| Row | created_on | name | database_name | schema_name | definition | owner | notification_channel |
|---|---|---|---|---|---|---|---|
| 1 | 2021-11-22 1... | TRIPS_PIPE | CITIBIKE_PIP... | PUBLIC | copy into tri... | ACCOUNTA... | arn:aws:sqs:us-east-1:941093095223:sf-sn |

That SQS ARN is available by clicking the Copy button on the results pane and pasting that into a text file or other editor.

**arn:aws:sqs:us-east-1:941093095223:sf-snowpipe-AIDA5WHMFA43YOTSXNQ7J--ZFTKNPCWky2PyrpnYtzsg**

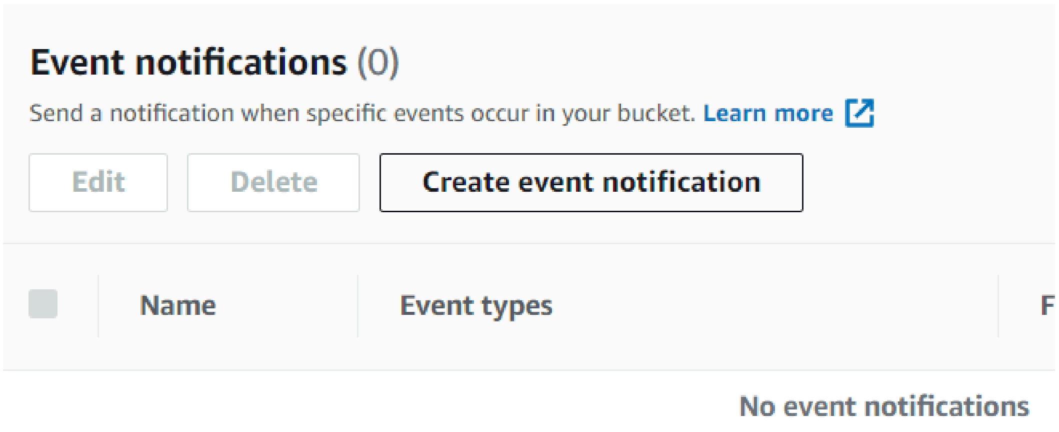
Sample screenshots of the S3 bucket enabling & configuring of event notifications below

View the S3 Console and choose the bucket that we created in our previous work



Amazon S3 > snowflake-data-pipeline-session1-george
snowflake-data-pipeline-session1-george Info
Objects
Properties
Permissions
Metrics
Management
Access Points

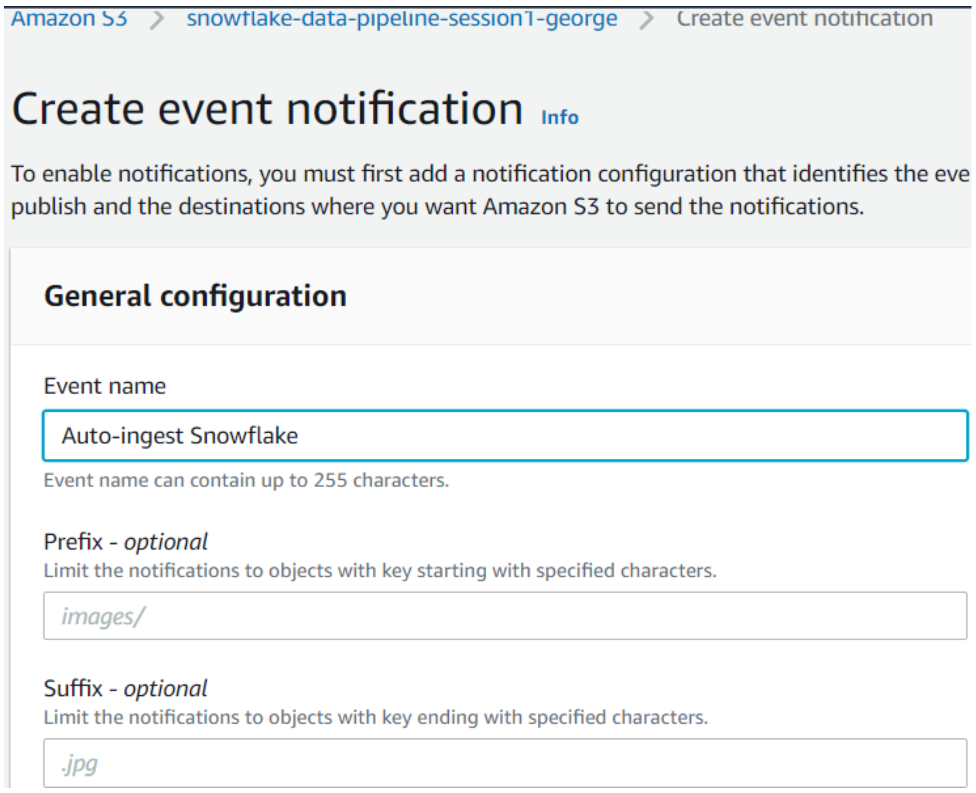
Choose the **Properties** tab and Scroll down to the **Event Notifications** section


Event notifications (0)

Send a notification when specific events occur in your bucket. Learn more

No event notifications

Choose **Create event notification**


Amazon S3 > snowflake-data-pipeline-session1-george > Create event notification

Create event notification Info

To enable notifications, you must first add a notification configuration that identifies the eve
publish and the destinations where you want Amazon S3 to send the notifications.

General configuration

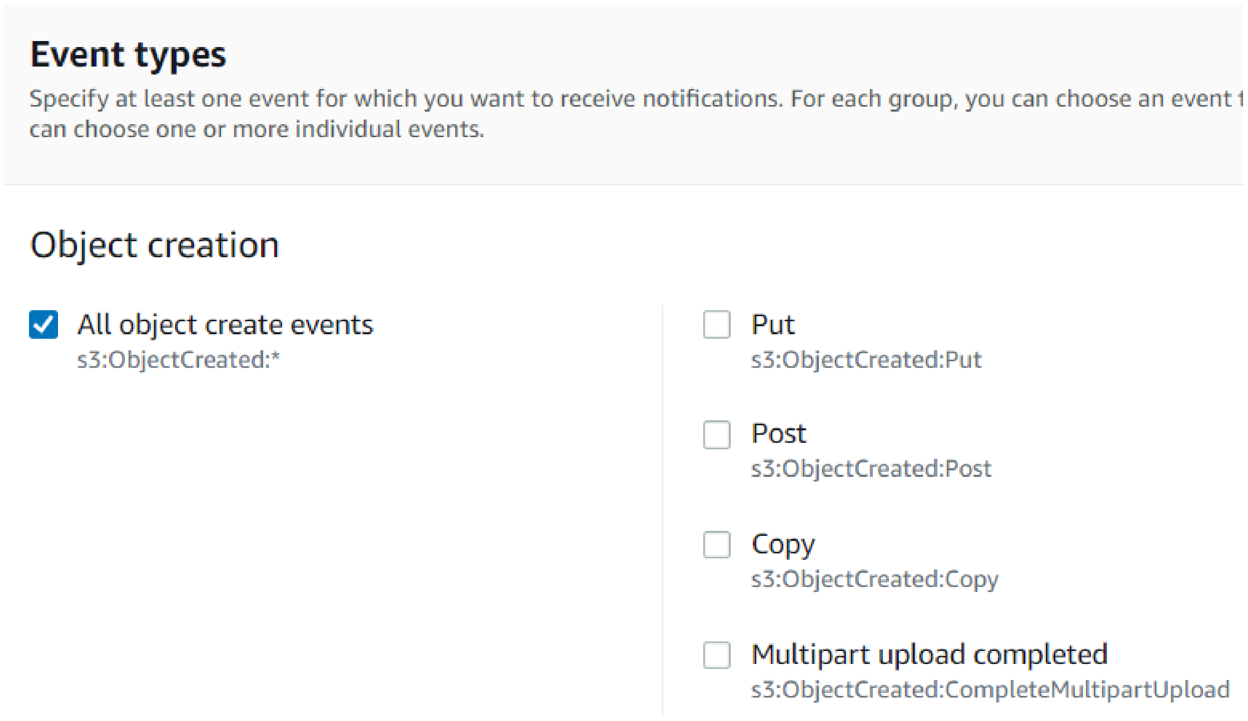
Event name: Auto-ingest Snowflake

Event name can contain up to 255 characters.

Prefix - *optional*
Limit the notifications to objects with key starting with specified characters.
*images/*

Suffix - *optional*
Limit the notifications to objects with key ending with specified characters.
*.jpg*

Enter **Auto-ingest Snowflake** for the Event name


Event types

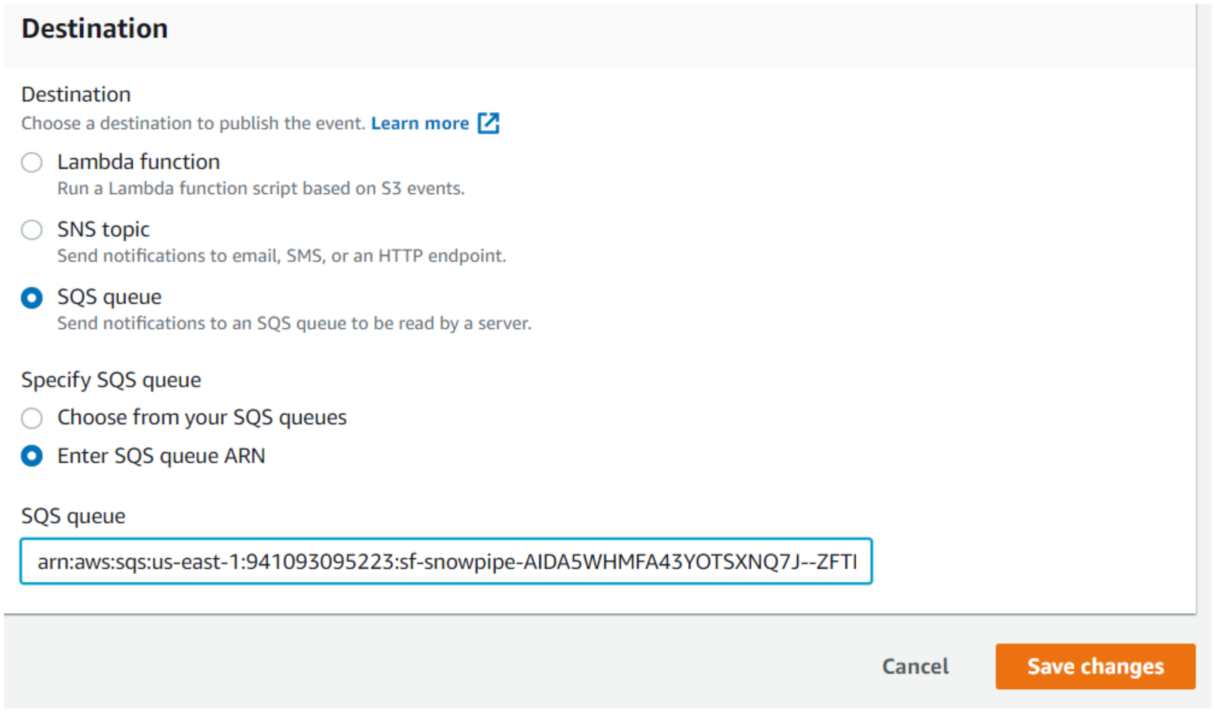
Specify at least one event for which you want to receive notifications. For each group, you can choose an event t
can choose one or more individual events.

Object creation

- [✓] All object create events s3:ObjectCreated:*
- [ ] Put s3:ObjectCreated:Put
- [ ] Post s3:ObjectCreated:Post
- [ ] Copy s3:ObjectCreated:Copy
- [ ] Multipart upload completed s3:ObjectCreated:CompleteMultipartUpload

22.2.1 Under Event Types and Object creation check the box for **All object create events**

22.2.2 Under Destination choose radio for SQS Queue and SQS Queue ARN and past the arn for the SQS queue that we retrieved with the “show pipes” result.



**Destination**

Destination

Choose a destination to publish the event. **Learn more**

( ) Lambda function
Run a Lambda function script based on S3 events.

( ) SNS topic
Send notifications to email, SMS, or an HTTP endpoint.

[✓ SQS queue]
Send notifications to an SQS queue to be read by a server.

Specify SQS queue

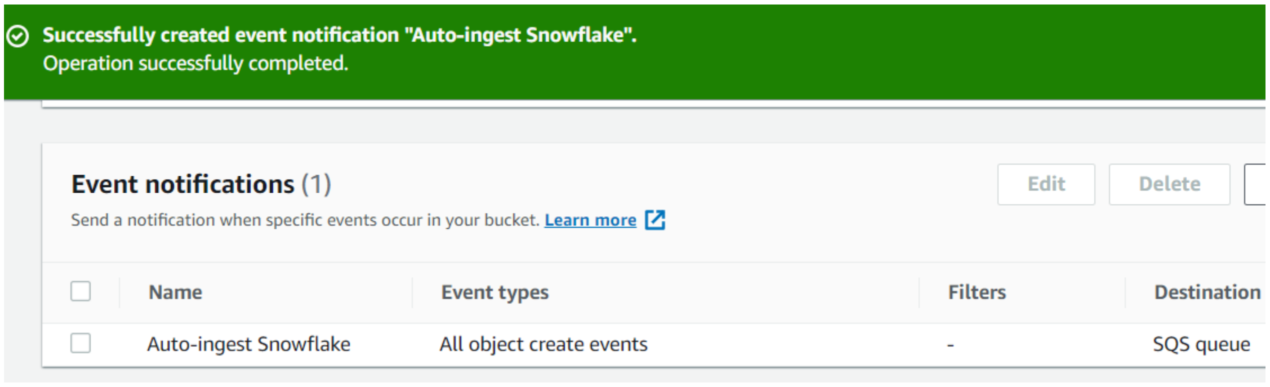
( ) Choose from your SQS queues

[✓ Enter SQS queue ARN]

SQS queue

arn:aws:sqs:us-east-1:941093095223:sf-snowpipe-AIDA5WHMFA43YOTSXNQ7J--ZFTI

22.2.3 Choose **Save changes**



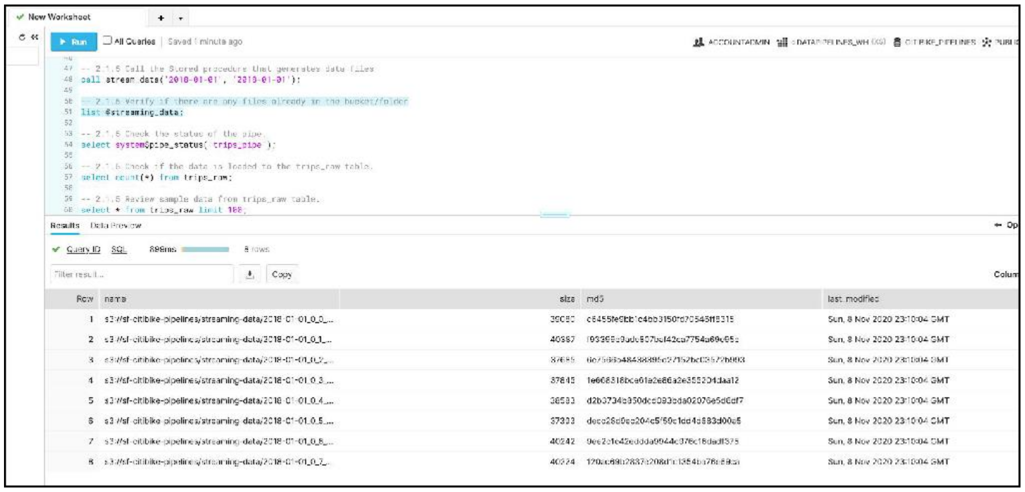
**Successfully created event notification "Auto-ingest Snowflake".**
Operation successfully completed.

**Event notifications** (1)

Send a notification when specific events occur in your bucket. Learn more

| Name | Event types | Filters | Destination |
|---|---|---|---|
| Auto-ingest Snowflake | All object create events | - | SQS queue |

22.2.4 Under Event Types and Object creation check the box for **All object create events**



```
-- 2.1.5 Call the Stored procedure that generates data files
call stream_data('2018-01-01', '2018-01-01');

-- 2.1.5 Verify if there are any files already in the bucket/folder
list @streaming_data;

-- 2.1.5 Check the status of the pipe.
select system$pipe_status( trips_pipe );

-- 2.1.5 Check if the data is loaded to the trips_raw table.
select count(*) from trips_raw;

-- 2.1.5 Review sample data from trips_raw table.
select * from trips_raw limit 100;
```

| Row | name | size | md5 | last_modified |
|---|---|---|---|---|
| 1 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_0_... | 39080 | c6455fe9bb1c4bb3150fd70545ff8315 | Sun, 8 Nov 2020 23:10:04 GMT |
| 2 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_1_... | 40387 | f93998c9adc607baf42ca7754a69c95c | Sun, 8 Nov 2020 23:10:04 GMT |
| 3 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_2_... | 37685 | 6c75665484888956d27152bc03572b903 | Sun, 8 Nov 2020 23:10:04 GMT |
| 4 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_3_... | 37845 | 1e066318bce61a2e86a2e355204daa12 | Sun, 8 Nov 2020 23:10:04 GMT |
| 5 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_4_... | 38583 | d2b3734b950dcc093bda02076e5d6df7 | Sun, 8 Nov 2020 23:10:04 GMT |
| 6 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_5_... | 37333 | decc28d0ec204c5f50c1dd4d883d00a5 | Sun, 8 Nov 2020 23:10:04 GMT |
| 7 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_6_... | 40242 | 9ee2c1c42eddda9944c376c18dadf375 | Sun, 8 Nov 2020 23:10:04 GMT |
| 8 | s3://sf-citibike-pipelines/streaming-data/2018-01-01_0_7_... | 40224 | 120ac69b2837c208d1c1354ba78c69ca | Sun, 8 Nov 2020 23:10:04 GMT |

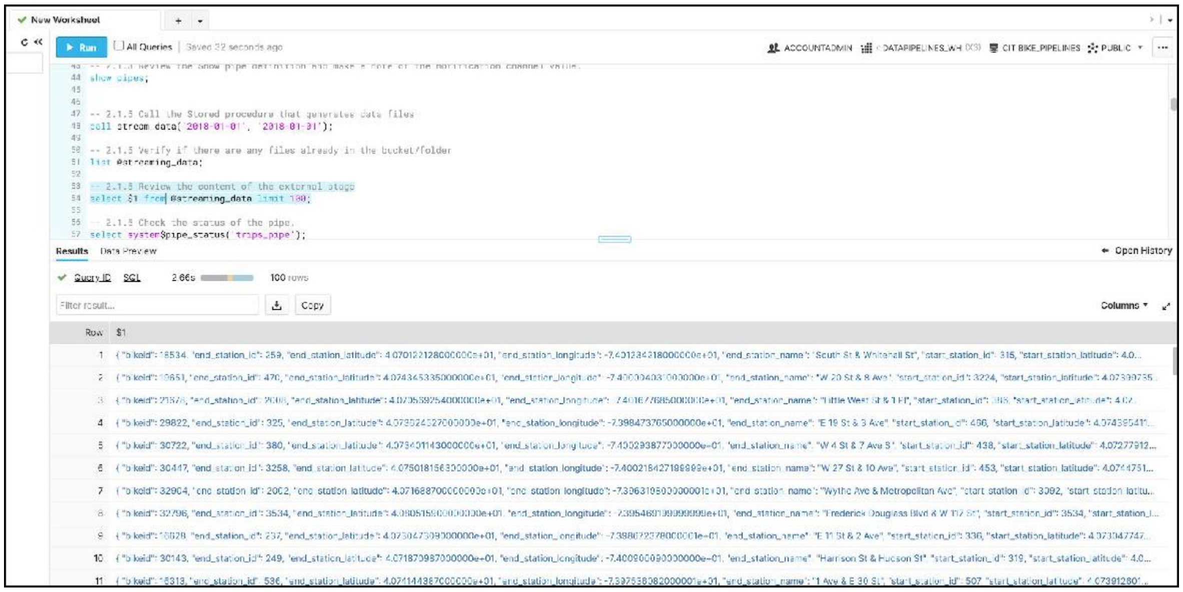
22.1.5 Confirm the setup is working by running the below steps.

Load 1 day of data to test the ingestion by calling the STREAM_DATA stored procedure.

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

22.1.6 Verify if there are any files already in the bucket/folder.

```
list @streaming_data;
```



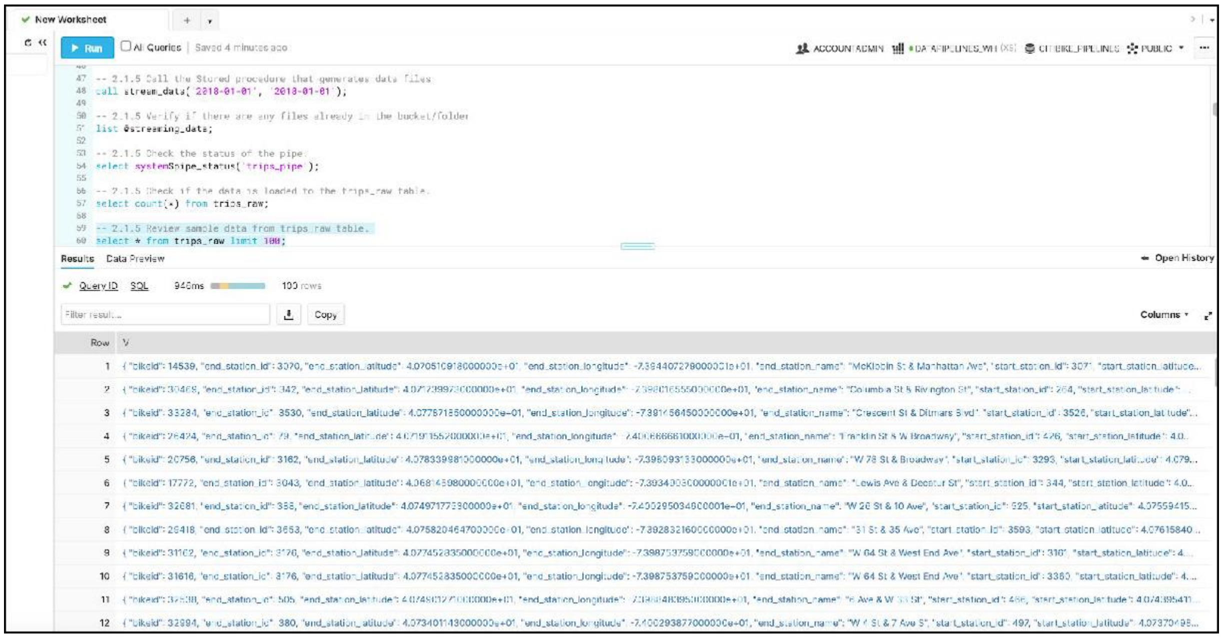
```
-- 2.1.3 Review the show pipe definition and make a note of the notification channel value.
show pipes;


-- 2.1.5 Call the Stored procedure that generates data files
call stream_data('2018-01-01', '2018-01-31');

-- 2.1.5 Verify if there are any files already in the bucket/folder
list @streaming_data;

-- 2.1.5 Review the content of the external stage
select $1 from @streaming_data limit 100;

-- 2.1.5 Check the status of the pipe.
select system$pipe_status('trips_pipe');
```

2.66s 100 rows

| Row | $1 |
|---|---|
| 1 | { "bikeid": 16534, "end_station_id": 259, "end_station_latitude": 4.070122128000000e+01, "end_station_longitude": -7.401284218000000e+01, "end_station_name": "South St & Whitehall St", "start_station_id": 316, "start_station_latitude": 4.0... |
| 2 | { "bikeid": 19651, "end_station_id": 470, "end_station_latitude": 4.074345335000000e+01, "end_station_longitude": -7.400004031000000e+01, "end_station_name": "W 20 St & 8 Ave", "start_station_id": 3224, "start_station_latitude": 4.07399735... |
| 3 | { "bikeid": 21576, "end_station_id": 2008, "end_station_latitude": 4.070559254000000e+01, "end_station_longitude": -7.401677685000000e+01, "end_station_name": "Little West St & 1 Pl", "start_station_id": 385, "start_station_latitude": 4.07... |
| 4 | { "bikeid": 29822, "end_station_id": 325, "end_station_latitude": 4.073624327000000e+01, "end_station_longitude": -7.398473765000000e+01, "end_station_name": "E 19 St & 3 Ave", "start_station_id": 466, "start_station_latitude": 4.074395411... |
| 5 | { "bikeid": 30722, "end_station_id": 380, "end_station_latitude": 4.073401143000000e+01, "end_station_longitude": -7.400293877000000e+01, "end_station_name": "W 4 St & 7 Ave S", "start_station_id": 438, "start_station_latitude": 4.07277912... |
| 6 | { "bikeid": 30447, "end_station_id": 3258, "end_station_latitude": 4.075018156300000e+01, "end_station_longitude": -7.400218427189999e+01, "end_station_name": "W 27 St & 10 Ave", "start_station_id": 453, "start_station_latitude": 4.0744751... |
| 7 | { "bikeid": 32904, "end_station_id": 2002, "end_station_latitude": 4.071688700000000e+01, "end_station_longitude": -7.396310600000001e+01, "end_station_name": "Wythe Ave & Metropolitan Ave", "start_station_id": 3092, "start_station_latitu... |
| 8 | { "bikeid": 32796, "end_station_id": 3534, "end_station_latitude": 4.080515900000000e+01, "end_station_longitude": -7.395460139999999e+01, "end_station_name": "Frederick Douglass Blvd & W 117 St", "start_station_id": 3534, "start_station_l... |
| 9 | { "bikeid": 16828, "end_station_id": 237, "end_station_latitude": 4.073047309000000e+01, "end_station_longitude": -7.398672378000001e+01, "end_station_name": "E 11 St & 2 Ave", "start_station_id": 336, "start_station_latitude": 4.073047747... |
| 10 | { "bikeid": 30143, "end_station_id": 249, "end_station_latitude": 4.071870987000000e+01, "end_station_longitude": -7.400900090000000e+01, "end_station_name": "Harrison St & Hudson St", "start_station_id": 319, "start_station_latitude": 4.0... |
| 11 | { "bikeid": 16313, "end_station_id": 536, "end_station_latitude": 4.071144387000000e+01, "end_station_longitude": -7.397538082000001e+01, "end_station_name": "1 Ave & E 30 St", "start_station_id": 507, "start_station_latitude": 4.07391280... |

22.1.7 Review the contents of the external stage

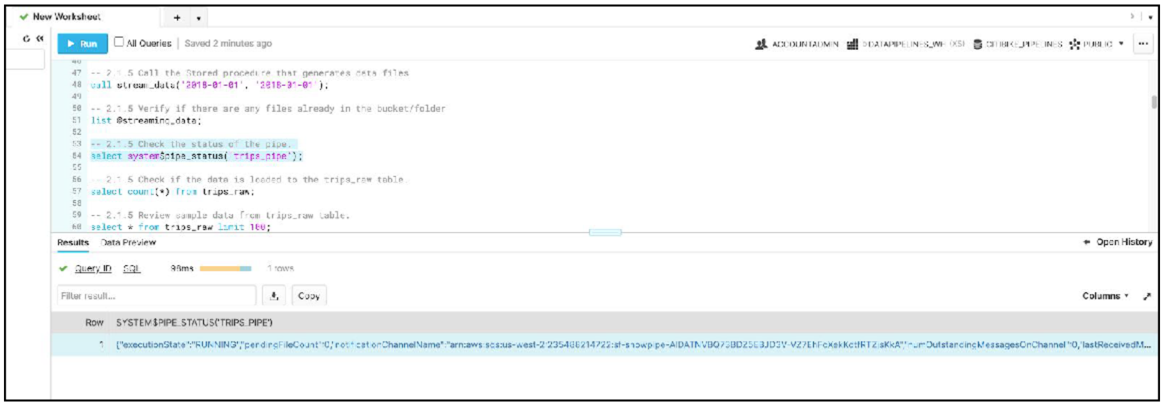
```
select $1 from @streaming_data limit 100;
```



```
-- 2.1.5 Call the Stored procedure that generates data files
call stream_data('2018-01-01', '2018-01-01');

-- 2.1.5 Verify if there are any files already in the bucket/folder
list @streaming_data;

-- 2.1.5 Check the status of the pipe.
select system$pipe_status('trips_pipe');

-- 2.1.5 Check if the data is loaded to the trips_raw table.
select count(*) from trips_raw;

-- 2.1.5 Review sample data from trips_raw table.
select * from trips_raw limit 100;
```

```
Query ID  SQL  946ms  100 rows

Row  V
1   { "bikeid": 14539, "end_station_id": 3070, "end_station_latitude": 4.070510918000000e+01, "end_station_longitude": -7.394407278000001e+01, "end_station_name": "McKibbin St & Manhattan Ave", "start_station_id": 3071, "start_station_latitude...
2   { "bikeid": 30469, "end_station_id": 342, "end_station_latitude": 4.071739973000000e+01, "end_station_longitude": -7.398016555000000e+01, "end_station_name": "Columbia St & Rivington St", "start_station_id": 264, "start_station_latitude"...
3   { "bikeid": 33284, "end_station_id": 3530, "end_station_latitude": 4.077871850000000e-01, "end_station_longitude": -7.391456450000000e+01, "end_station_name": "Crescent St & Ditmars Blvd", "start_station_id": 3526, "start_station_latitude"...
4   { "bikeid": 26424, "end_station_id": 79, "end_station_latitude": 4.071911552000000e+01, "end_station_longitude": -7.400666661000000e-01, "end_station_name": "Franklin St & W Broadway", "start_station_id": 426, "start_station_latitude": 4.0...
5   { "bikeid": 20756, "end_station_id": 3162, "end_station_latitude": 4.078339981000000e+01, "end_station_longitude": -7.398093133000000e+01, "end_station_name": "W 78 St & Broadway", "start_station_id": 3293, "start_station_latitude": 4.079...
6   { "bikeid": 17772, "end_station_id": 3043, "end_station_latitude": 4.068145980000000e+01, "end_station_longitude": -7.393400300000001e+01, "end_station_name": "Lewis Ave & Decatur St", "start_station_id": 344, "start_station_latitude": 4.0...
7   { "bikeid": 32681, "end_station_id": 358, "end_station_latitude": 4.074971773900000e+01, "end_station_longitude": -7.400295034900001e-01, "end_station_name": "W 26 St & 10 Ave", "start_station_id": 525, "start_station_latitude": 4.07559415...
8   { "bikeid": 29418, "end_station_id": 3653, "end_station_latitude": 4.075820464700000e-01, "end_station_longitude": -7.392832160000000e+01, "end_station_name": "31 St & 35 Ave", "start_station_id": 3593, "start_station_latitude": 4.07615840...
9   { "bikeid": 31162, "end_station_id": 3176, "end_station_latitude": 4.077452835000000e+01, "end_station_longitude": -7.398753759000000e+01, "end_station_name": "W 64 St & West End Ave", "start_station_id": 3161, "start_station_latitude": 4...
10  { "bikeid": 31616, "end_station_id": 3176, "end_station_latitude": 4.077452835000000e+01, "end_station_longitude": -7.398753759000000e+01, "end_station_name": "W 64 St & West End Ave", "start_station_id": 3360, "start_station_latitude": 4....
11  { "bikeid": 32538, "end_station_id": 505, "end_station_latitude": 4.074801271000000e+01, "end_station_longitude": -7.398848395000000e+01, "end_station_name": "6 Ave & W 33 St", "start_station_id": 466, "start_station_latitude": 4.07439541...
12  { "bikeid": 32994, "end_station_id": 380, "end_station_latitude": 4.073401143000000e+01, "end_station_longitude": -7.400293877000000e+01, "end_station_name": "W 4 St & 7 Ave S", "start_station_id": 497, "start_station_latitude": 4.07370495...
```

## 22.1.8 Check the status of the Snow Pipe



```
-- 2.1.5 Call the Stored procedure that generates data files
call stream_data('2018-01-01', '2018-01-01');

-- 2.1.5 Verify if there are any files already in the bucket/folder
list @streaming_data;

-- 2.1.5 Check the status of the pipe.
select system$pipe_status('trips_pipe');

-- 2.1.5 Check if the data is loaded to the trips_raw table
select count(*) from trips_raw;

-- 2.1.5 Review sample data from trips_raw table.
select * from trips_raw limit 100;
```

```
Query ID  SQL  98ms  1 rows

Row  SYSTEM$PIPE_STATUS('TRIPS_PIPE')
1    {"executionState":"RUNNING","pendingFileCount":0,"notificationChannelName":"arn:aws:sqs:us-west-2:235408214722:sf-snowpipe-AIDATNVBQ73BD25E3JD3V-VZ7EhFcXekKctIRTZisKkA","numOutstandingMessagesOnChannel":0,"lastReceivedM...
```

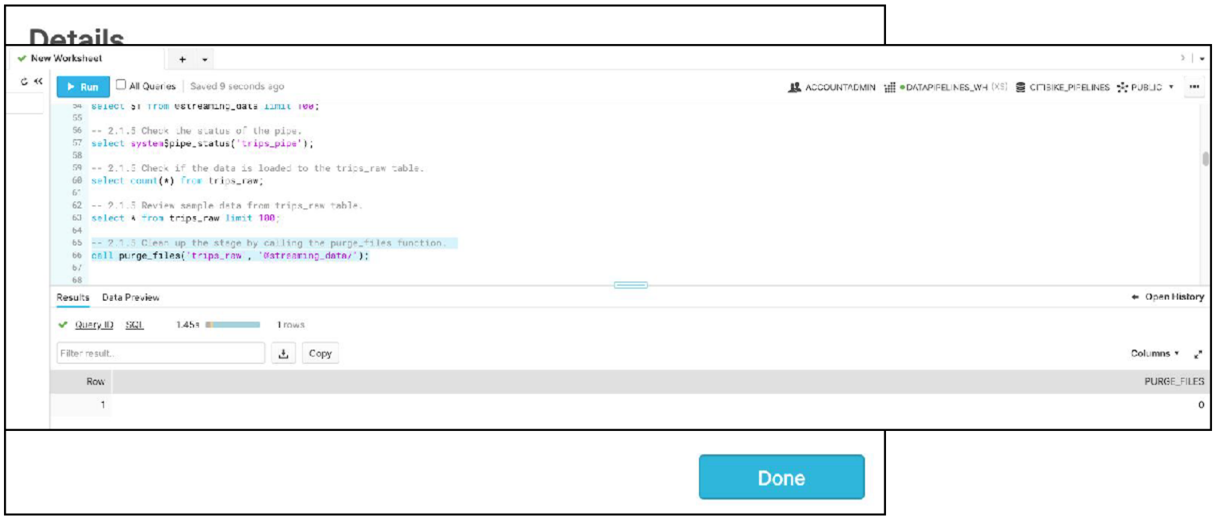
## 22.1.9 Further check the status

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

Snowpipe copies the data into our raw table. Review the table data.

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

Clean up the stage by calling the purge_files function.



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

-- 2.1.5 Check the status of the pipe.
select system$pipe_status('trips_pipe');

-- 2.1.5 Check if the data is loaded to the trips_raw table.
select count(*) from trips_raw;

-- 2.1.5 Review sample data from trips_raw table.
select * from trips_raw limit 100;

-- 2.1.5 Clean up the stage by calling the purge_files function.
call purge_files('trips_raw', '@streaming_data/');
```

| Row | PURGE_FILES |
|---|---|
| 1 | 0 |

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

Truncate table.

```
truncate table trips_raw;
```

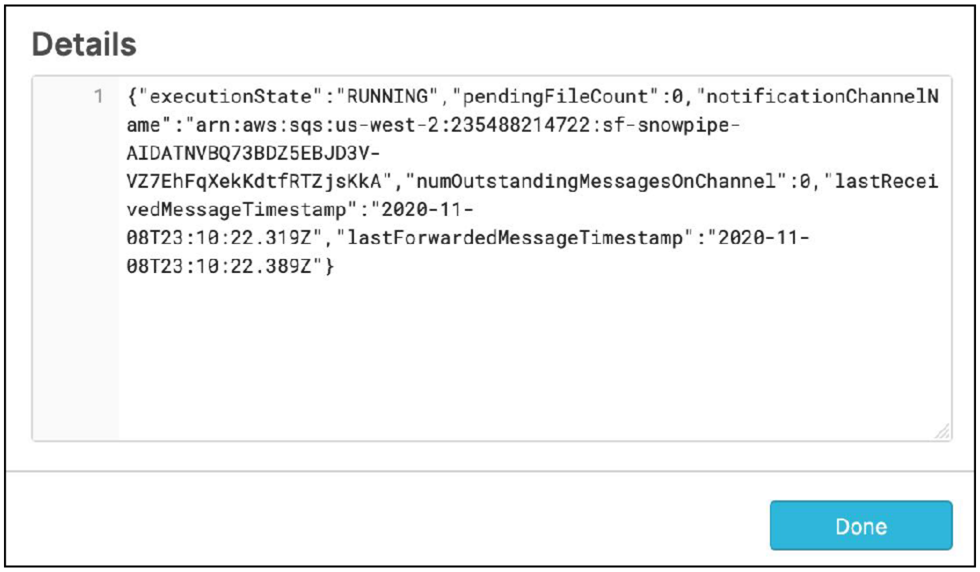
## 22.2 Troubleshooting Snowpipe

If data isn’t written to the table you can use the following commands to see where the processis broken.

Verify data was in fact written to your STAGE by the stored procedure.

```
list @streaming_data;
```

Next check the status of the PIPE using the system$pipe_status call. Make sure theresult shows status ‘RUNNING’ and there are no errors present.



```
{"executionState":"RUNNING","pendingFileCount":0,"notificationChannelName":"arn:aws:sqs:us-west-2:235488214722:sf-snowpipe-AIDATNVBQ73BDZ5EBJD3V-VZ7EhFqXekKdtfRTZjsKkA","numOutstandingMessagesOnChannel":0,"lastReceivedMessageTimestamp":"2020-11-08T23:10:22.319Z","lastForwardedMessageTimestamp":"2020-11-08T23:10:22.389Z"}
```

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

Lastly, check the information_schema.copy_history to see if there were any problems encountered while writing records to the trips_raw table. Parsing errors or permissions errors could accidentally affect your load jobs. You can find the details for each of the files you are loading. Play with the dateadd function to adjust the size of the window when looking at processed files.

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

***Note:*** *the above commands aren’t strictly used for troubleshooting purposes. You can use these anytime to gain insight into the status or jobs and track the velocity of arriving data.*