

```
/*
Tasty Bytes is a fictitious, global food truck network, that is on a mission to serve unique food
options with high
quality items in a safe, convenient and cost effective way. In order to drive forward on their
mission, Tasty Bytes
is beginning to leverage the Snowflake Data Cloud.
```

Within this Worksheet, we will walk through the end to end process required to load a CSV file containing Menu specific data that is currently hosted in Blob Storage.

```
--*/
```

```
-- Step 1: To start, let's set the Role and Warehouse context
-- USE ROLE: https://docs.snowflake.com/en/sql-reference/sql/use-role
-- USE WAREHOUSE: https://docs.snowflake.com/en/sql-reference/sql/use-warehouse
```

```
/*
- To run a single query, place your cursor in the query editor and select the Run button
(⌘-Return).
- To run the entire worksheet, select 'Run All' from the dropdown next to the Run button
(⌘-Shift-Return).
--*/
```

```
--> set the Role
USE ROLE accountadmin;
```

```
--> set the Warehouse
USE WAREHOUSE compute_wh;
```

```
-- Step 2: With context in place, let's now create a Database, Schema, and Table
-- CREATE DATABASE:
https://docs.snowflake.com/en/sql-reference/sql/create-database
-- CREATE SCHEMA: https://docs.snowflake.com/en/sql-reference/sql/create-schema
-- CREATE TABLE: https://docs.snowflake.com/en/sql-reference/sql/create-table
```

```
--> create the Tasty Bytes Database
CREATE OR REPLACE DATABASE tasty_bytes_sample_data;
```

```
--> create the Raw POS (Point-of-Sale) Schema
CREATE OR REPLACE SCHEMA tasty_bytes_sample_data.raw_pos;
```

```
--> create the Raw Menu Table
CREATE OR REPLACE TABLE tasty_bytes_sample_data.raw_pos.menu
(
    menu_id NUMBER(19,0),
```

```
menu_type_id NUMBER(38,0),
menu_type VARCHAR(16777216),
truck_brand_name VARCHAR(16777216),
menu_item_id NUMBER(38,0),
menu_item_name VARCHAR(16777216),
item_category VARCHAR(16777216),
item_subcategory VARCHAR(16777216),
cost_of_goods_usd NUMBER(38,4),
sale_price_usd NUMBER(38,4),
menu_item_health_metrics_obj VARIANT
);
```

--> confirm the empty Menu table exists
SELECT * FROM tasty_bytes_sample_data.raw_pos.menu;

-- Step 3: To connect to the Blob Storage, let's create a Stage

-- Creating an S3 Stage:

<https://docs.snowflake.com/en/user-guide/data-load-s3-create-stage>

--> create the Stage referencing the Blob location and CSV File Format
CREATE OR REPLACE STAGE tasty_bytes_sample_data.public.blob_stage
url = 's3://sfquickstarts/tastybytes/'
file_format = (type = csv);

--> query the Stage to find the Menu CSV file
LIST @tasty_bytes_sample_data.public.blob_stage/raw_pos/menu/;

-- Step 4: Now let's Load the Menu CSV file from the Stage

-- COPY INTO <table>: <https://docs.snowflake.com/en/sql-reference/sql/copy-into-table>

--> copy the Menu file into the Menu table
COPY INTO tasty_bytes_sample_data.raw_pos.menu
FROM @tasty_bytes_sample_data.public.blob_stage/raw_pos/menu/;

-- Step 5: Query the Menu table

-- SELECT: <https://docs.snowflake.com/en/sql-reference/sql/select>

-- TOP <n>: https://docs.snowflake.com/en/sql-reference/constructs/top_n

-- FLATTEN: <https://docs.snowflake.com/en/sql-reference/functions/flatten>

--> how many rows are in the table?

SELECT COUNT(*) AS row_count FROM tasty_bytes_sample_data.raw_pos.menu;

--> what do the top 10 rows look like?

```
SELECT TOP 10 * FROM tasty_bytes_sample_data.raw_pos.menu;
```

--> what menu items does the Freezing Point brand sell?

```
SELECT
    menu_item_name
FROM tasty_bytes_sample_data.raw_pos.menu
WHERE truck_brand_name = 'Freezing Point';
```

--> what is the profit on Mango Sticky Rice?

```
SELECT
    menu_item_name,
    (sale_price_usd - cost_of_goods_usd) AS profit_usd
FROM tasty_bytes_sample_data.raw_pos.menu
WHERE 1=1
AND truck_brand_name = 'Freezing Point'
AND menu_item_name = 'Mango Sticky Rice';
```

--> to finish, let's extract the Mango Sticky Rice ingredients from the semi-structured column

```
SELECT
    m.menu_item_name,
    obj.value:"ingredients":ARRAY AS ingredients
FROM tasty_bytes_sample_data.raw_pos.menu m,
    LATERAL FLATTEN (input =>
m.menu_item_health_metrics_obj:menu_item_health_metrics) obj
WHERE 1=1
AND truck_brand_name = 'Freezing Point'
AND menu_item_name = 'Mango Sticky Rice';
```

The screenshot shows the Snowflake SQL interface. On the left, the sidebar displays 'Databases' and 'Worksheets'. The main area shows a code editor with the following SQL script:

```
20 ---> set the Role
21 USE ROLE accountadmin;
22
23 ---> set the Warehouse
24 USE WAREHOUSE compute_wh;
25
26 -----
27 -- Step 2: With context in place, let's now create a Database, Schema, and Table
28 -- CREATE DATABASE: https://docs.snowflake.com/en/sql-reference/sql/create-database
29 -- CREATE SCHEMA: https://docs.snowflake.com/en/sql-reference/sql/create-schema
30
```

The code editor has syntax highlighting and line numbers. Below the code editor, there are tabs for 'Results' and 'Chart'. The 'Results' tab shows the message: 'Load sample data with SQL from S3 bucket' and 'Statement executed successfully.' A progress bar indicates the status is at 1. To the right of the results, there are options for 'Chart type' (Bar), 'Data' (status count), and 'Ask Copilot'.

Load sample data with SQL from S3 bucket

```
33
34 ---> create the Tasty Bytes Database
35 CREATE OR REPLACE DATABASE tasty_bytes_sample_data;
36
37 ---> Create the Raw POS (Point-of-Sale) Schema
38 CREATE OR REPLACE SCHEMA tasty_bytes_sample_data.raw_pos;
39
40 ---> Create the Raw Menu Table
41 CREATE OR REPLACE TABLE tasty_bytes_sample_data.raw_pos.menu
42 (
43     menu_id NUMBER(19,0),
```

Results Chart

Load sample data with SQL from S3 bucket

Database TASTY_BYTES_SAMPLE... 1

Chart type Bar

Data status count Ask Copilot

Load sample data with SQL from S3 bucket

```
33
34 ---> create the Tasty Bytes Database
35 CREATE OR REPLACE DATABASE tasty_bytes_sample_data;
36
37 ---> Create the Raw POS (Point-of-Sale) Schema
38 CREATE OR REPLACE SCHEMA tasty_bytes_sample_data.raw_pos;
39
40 ---> Create the Raw Menu Table
41 CREATE OR REPLACE TABLE tasty_bytes_sample_data.raw_pos.menu
42 (
43     menu_id NUMBER(19,0),
```

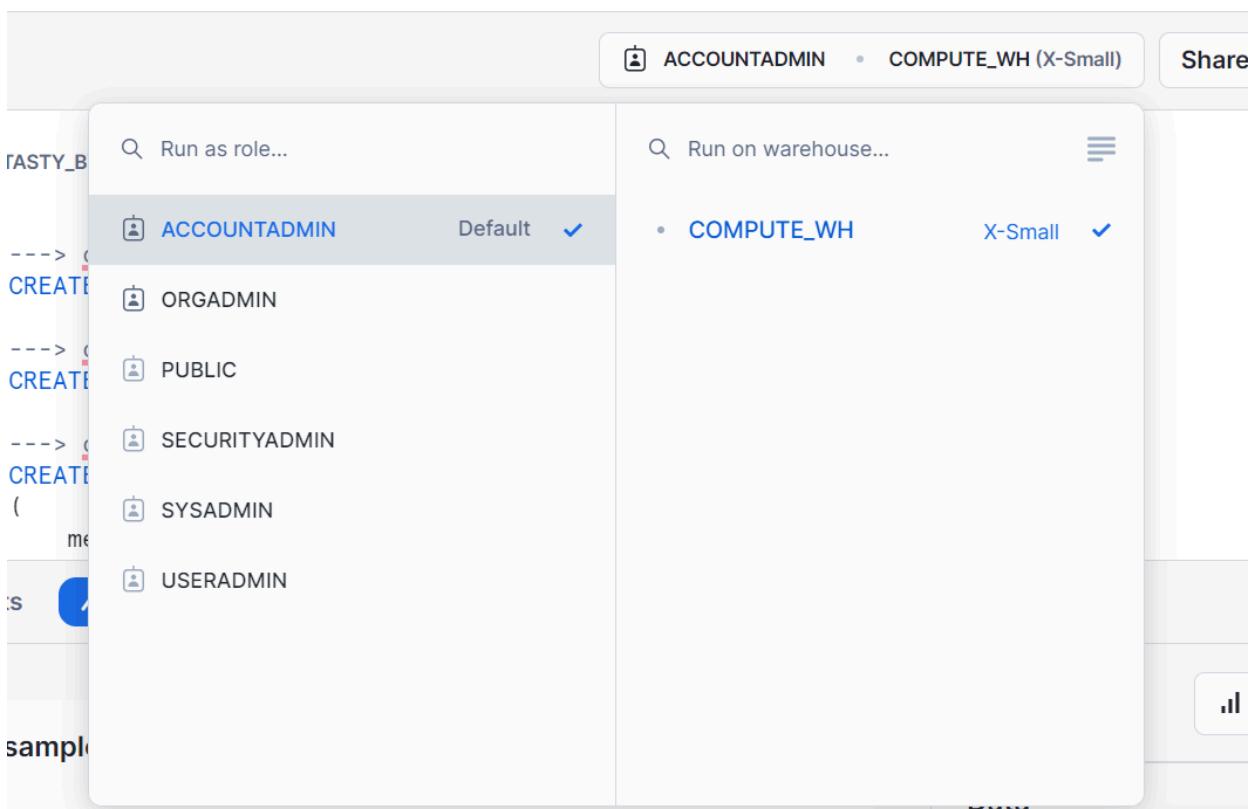
Results Chart

Load sample data with SQL from S3 bucket

Schema RAW_POS successfully cre... 1

Chart type Bar

Data status count Ask Copilot



When there is an error:

The screenshot shows the Snowflake UI with a query editor. The top navigation bar includes 'Load sample data with SQ...', '+', and a dropdown. Below it, the 'Databases' tab is selected. The main area shows a query editor with the following code:

```
CREATE OR REPLACE SCHEMA tasty_bytes_sample_data.raw_pos;
--> create the Raw Menu Table
CREATE OR REPLACE TABLE tasty_bytes_sample_data.raw_pos.menu
    menu_id NUMBER(19,0),
    menu_type_id NUMBER(38,0),
    menu_type VARCHAR(16777216),
    truck_brand_name VARCHAR(16777216),
    menu_item_id NUMBER(38,0),
    menu_item_name VARCHAR(16777216)
```

The line 'menu_type VARCHAR(16777216),
truck_brand_name VARCHAR(16777216),
menu_item_id NUMBER(38,0),
menu_item_name VARCHAR(16777216)' is highlighted in blue, indicating a syntax error. A yellow warning icon is displayed above the results table. The results table shows the error message: 'Syntax error: unexpected '<EOF>'.' (line 40). The bottom right corner shows 'Ask Copilot' and 'COMPUTE_WH'.

Query Results:

50

↳ Results ↵ Chart

	status
1	Table MENU successfully created.

Rows 1
Query ID 01b76754-0003-42ae-0...
End time Oct 1, 11:26 PM
Scan 0.0
Role ACCOUNTADMIN
Warehouse COMPUTE_WH

Show less ^ Ask Copilot

55
56 ---> confirm the empty Menu table exists
57 SELECT * FROM tasty_bytes_sample_data.raw_pos.menu;
58

↳ Results ↵ Chart

	MENU_ID	MENU_TYPE_ID	MENU_TYPE	TRUCK_BRAND_NAME
Query produced no results				

Query Details ...
Query duration 71ms
Rows 0
Query ID 01b76755-0003-42a9-0...
Show more ▾ Ask Copilot

55
56 ---> create the Stage referencing the Blob location and CSV File Format
57 CREATE OR REPLACE STAGE tasty_bytes_sample_data.public.blob_stage
58 url = 's3://sfquickstarts/tastybytes/'
59 file_format = (type = csv);
60
61 ---> try the Stage to find the Menu CSV file
62 LIST @tasty_bytes_sample_data.public.blob_stage/raw_pos/menu/;

↳ Results ↵ Chart

	status
1	Stage area BLOB_STAGE successfully created.

Query Details ...
Query duration 289ms
Rows 1
Query ID 01b76756-0003-42b3-...
Show more ▾ Ask Copilot

```

69      ---> query the Stage to find the Menu CSV file
70  LIST @tasty_bytes_sample_data.public.blob_stage/raw_pos/menu/;
71
72   
73
74  -----
75      -- Step 4: Now let's Load the Menu CSV file from the Stage
76      -- COPY INTO <table>: https://docs.snowflake.com/en/sql-reference/sql/copy-into-table
77

```

↳ Results **~ Chart**

	name	size	md5
1	s3://sfquickstarts/tastybytes/raw_pos/menu/menu.csv.gz	3478	9dd9a858e141c2e...

Query Details ...
 Query duration 144ms
 Rows 1
 Query ID 01b76756-0003-44d3-...

Show more

77 -----
78
79 ---> copy the Menu file into the Menu table
80 COPY INTO tasty_bytes_sample_data.raw_pos.menu
81 |FROM @tasty_bytes_sample_data.public.blob_stage/raw_pos/menu/;
82
83
84 -----

↳ Results **~ Chart**

	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_err
1	s3://sfquickstarts/tastybytes/raw_pos/menu/menu.csv.gz	LOADED	100	100	1	0	null

Query Details ...
 Query duration 1.5s
 Rows 1
 Query ID 01b76757-0003-44d3-...

Show more

91 ---> how many rows are in the table?
92 SELECT COUNT(*) AS row_count FROM tasty_bytes_sample_data.raw_pos.menu;
93
94
95 ---> what do the top 10 rows look like?
96 SELECT TOP 10 * FROM tasty_bytes_sample_data.raw_pos.menu;
97
98 ---> what menu items does the Freezing Point brand sell?
99 SELECT
 menu_item_name

↳ Results **~ Chart**

	ROW_COUNT
1	100

```

93 | ---> what do the top 10 rows look like?
94 | SELECT TOP 10 * FROM tasty_bytes_sample_data.raw_pos.menu;
95 |
96 |
97 | ---> what menu items does the Freezing Point brand sell?
98 | SELECT
99 |   menu_item_name
100 | FROM tasty_bytes_sample_data.raw_pos.menu
101 | WHERE truck_brand_name = 'Freezing Point';
102 |
103 | ---> what is the profit on Mango Sticky Rice?
104 | SELECT
105 |   menu_item_name,
106 |   (sale_price_usd - cost_of_goods_usd) AS profit_usd
107 | FROM tasty_bytes_sample_data.raw_pos.menu
108 | WHERE 1=1
109 | AND truck_brand_name = 'Freezing Point'
110 | AND menu_item_name = 'Mango Sticky Rice';
111

```

Results

MENU_ID	MENU_TYPE_ID	MENU_TYPE	TRUCK_BRAND_NAME	MENU_ITEM_ID	MENU_ITEM_NAME	ITEM_CATEGORY	ITEM_TYPE
1	10001	1	Ice Cream	Freezing Point	10	Lemonade	Beverage
2	10002	1	Ice Cream	Freezing Point	11	Sugar Cone	Dessert
3	10003	1	Ice Cream	Freezing Point	12	Waffle Cone	Dessert
4	10004	1	Ice Cream	Freezing Point	13	Two Scoop Bowl	Dessert
5	10005	1	Ice Cream	Freezing Point	14	Shake	Dessert
6	10006	1	Ice Cream	Freezing Point	15	Fried Ice Cream	Dessert
7	10007	1	Ice Cream	Freezing Point	16	Ice Cream Cone	Dessert
8	10008	1	Ice Cream	Freezing Point	17	Ice Cream Sandwich	Dessert
9	10009	1	Ice Cream	Freezing Point	18	Ice Cream Parfait	Dessert
10	10010	1	Ice Cream	Freezing Point	19	Ice Cream Toppings	Dessert
11	10011	1	Ice Cream	Freezing Point	20	Ice Cream Sundae	Dessert
12	10012	1	Ice Cream	Freezing Point	21	Ice Cream Float	Dessert
13	10013	1	Ice Cream	Freezing Point	22	Ice Cream Cones	Dessert
14	10014	1	Ice Cream	Freezing Point	23	Ice Cream Toppings	Dessert
15	10015	1	Ice Cream	Freezing Point	24	Ice Cream Sundae	Dessert
16	10016	1	Ice Cream	Freezing Point	25	Ice Cream Float	Dessert
17	10017	1	Ice Cream	Freezing Point	26	Ice Cream Cones	Dessert
18	10018	1	Ice Cream	Freezing Point	27	Ice Cream Toppings	Dessert
19	10019	1	Ice Cream	Freezing Point	28	Ice Cream Sundae	Dessert
20	10020	1	Ice Cream	Freezing Point	29	Ice Cream Float	Dessert
21	10021	1	Ice Cream	Freezing Point	30	Ice Cream Cones	Dessert
22	10022	1	Ice Cream	Freezing Point	31	Ice Cream Toppings	Dessert
23	10023	1	Ice Cream	Freezing Point	32	Ice Cream Sundae	Dessert
24	10024	1	Ice Cream	Freezing Point	33	Ice Cream Float	Dessert
25	10025	1	Ice Cream	Freezing Point	34	Ice Cream Cones	Dessert
26	10026	1	Ice Cream	Freezing Point	35	Ice Cream Toppings	Dessert
27	10027	1	Ice Cream	Freezing Point	36	Ice Cream Sundae	Dessert
28	10028	1	Ice Cream	Freezing Point	37	Ice Cream Float	Dessert
29	10029	1	Ice Cream	Freezing Point	38	Ice Cream Cones	Dessert
30	10030	1	Ice Cream	Freezing Point	39	Ice Cream Toppings	Dessert
31	10031	1	Ice Cream	Freezing Point	40	Ice Cream Sundae	Dessert
32	10032	1	Ice Cream	Freezing Point	41	Ice Cream Float	Dessert
33	10033	1	Ice Cream	Freezing Point	42	Ice Cream Cones	Dessert
34	10034	1	Ice Cream	Freezing Point	43	Ice Cream Toppings	Dessert
35	10035	1	Ice Cream	Freezing Point	44	Ice Cream Sundae	Dessert
36	10036	1	Ice Cream	Freezing Point	45	Ice Cream Float	Dessert
37	10037	1	Ice Cream	Freezing Point	46	Ice Cream Cones	Dessert
38	10038	1	Ice Cream	Freezing Point	47	Ice Cream Toppings	Dessert
39	10039	1	Ice Cream	Freezing Point	48	Ice Cream Sundae	Dessert
40	10040	1	Ice Cream	Freezing Point	49	Ice Cream Float	Dessert
41	10041	1	Ice Cream	Freezing Point	50	Ice Cream Cones	Dessert
42	10042	1	Ice Cream	Freezing Point	51	Ice Cream Toppings	Dessert
43	10043	1	Ice Cream	Freezing Point	52	Ice Cream Sundae	Dessert
44	10044	1	Ice Cream	Freezing Point	53	Ice Cream Float	Dessert
45	10045	1	Ice Cream	Freezing Point	54	Ice Cream Cones	Dessert
46	10046	1	Ice Cream	Freezing Point	55	Ice Cream Toppings	Dessert
47	10047	1	Ice Cream	Freezing Point	56	Ice Cream Sundae	Dessert
48	10048	1	Ice Cream	Freezing Point	57	Ice Cream Float	Dessert
49	10049	1	Ice Cream	Freezing Point	58	Ice Cream Cones	Dessert
50	10050	1	Ice Cream	Freezing Point	59	Ice Cream Toppings	Dessert
51	10051	1	Ice Cream	Freezing Point	60	Ice Cream Sundae	Dessert
52	10052	1	Ice Cream	Freezing Point	61	Ice Cream Float	Dessert
53	10053	1	Ice Cream	Freezing Point	62	Ice Cream Cones	Dessert
54	10054	1	Ice Cream	Freezing Point	63	Ice Cream Toppings	Dessert
55	10055	1	Ice Cream	Freezing Point	64	Ice Cream Sundae	Dessert
56	10056	1	Ice Cream	Freezing Point	65	Ice Cream Float	Dessert
57	10057	1	Ice Cream	Freezing Point	66	Ice Cream Cones	Dessert
58	10058	1	Ice Cream	Freezing Point	67	Ice Cream Toppings	Dessert
59	10059	1	Ice Cream	Freezing Point	68	Ice Cream Sundae	Dessert
60	10060	1	Ice Cream	Freezing Point	69	Ice Cream Float	Dessert
61	10061	1	Ice Cream	Freezing Point	70	Ice Cream Cones	Dessert
62	10062	1	Ice Cream	Freezing Point	71	Ice Cream Toppings	Dessert
63	10063	1	Ice Cream	Freezing Point	72	Ice Cream Sundae	Dessert
64	10064	1	Ice Cream	Freezing Point	73	Ice Cream Float	Dessert
65	10065	1	Ice Cream	Freezing Point	74	Ice Cream Cones	Dessert
66	10066	1	Ice Cream	Freezing Point	75	Ice Cream Toppings	Dessert
67	10067	1	Ice Cream	Freezing Point	76	Ice Cream Sundae	Dessert
68	10068	1	Ice Cream	Freezing Point	77	Ice Cream Float	Dessert
69	10069	1	Ice Cream	Freezing Point	78	Ice Cream Cones	Dessert
70	10070	1	Ice Cream	Freezing Point	79	Ice Cream Toppings	Dessert
71	10071	1	Ice Cream	Freezing Point	80	Ice Cream Sundae	Dessert
72	10072	1	Ice Cream	Freezing Point	81	Ice Cream Float	Dessert
73	10073	1	Ice Cream	Freezing Point	82	Ice Cream Cones	Dessert
74	10074	1	Ice Cream	Freezing Point	83	Ice Cream Toppings	Dessert
75	10075	1	Ice Cream	Freezing Point	84	Ice Cream Sundae	Dessert
76	10076	1	Ice Cream	Freezing Point	85	Ice Cream Float	Dessert
77	10077	1	Ice Cream	Freezing Point	86	Ice Cream Cones	Dessert
78	10078	1	Ice Cream	Freezing Point	87	Ice Cream Toppings	Dessert
79	10079	1	Ice Cream	Freezing Point	88	Ice Cream Sundae	Dessert
80	10080	1	Ice Cream	Freezing Point	89	Ice Cream Float	Dessert
81	10081	1	Ice Cream	Freezing Point	90	Ice Cream Cones	Dessert
82	10082	1	Ice Cream	Freezing Point	91	Ice Cream Toppings	Dessert
83	10083	1	Ice Cream	Freezing Point	92	Ice Cream Sundae	Dessert
84	10084	1	Ice Cream	Freezing Point	93	Ice Cream Float	Dessert
85	10085	1	Ice Cream	Freezing Point	94	Ice Cream Cones	Dessert
86	10086	1	Ice Cream	Freezing Point	95	Ice Cream Toppings	Dessert
87	10087	1	Ice Cream	Freezing Point	96	Ice Cream Sundae	Dessert
88	10088	1	Ice Cream	Freezing Point	97	Ice Cream Float	Dessert
89	10089	1	Ice Cream	Freezing Point	98	Ice Cream Cones	Dessert
90	10090	1	Ice Cream	Freezing Point	99	Ice Cream Toppings	Dessert
91	10091	1	Ice Cream	Freezing Point	100	Ice Cream Sundae	Dessert

Query Details

- Query duration: 324ms
- Rows: 10
- Query ID: 01b76758-0003-42a9-...

Show more **Ask Copilot**

Results

MENU_ITEM_NAME
Lemonade
Sugar Cone
Waffle Cone
Two Scoop Bowl
Bottled Water

Query Details

- Query duration: 395ms
- Rows: 10
- Query ID: 01b76759-0003-42b3-...

Show more **Ask Copilot**

Results

MENU_ITEM_NAME	PROFIT_USD
Mango Sticky Rice	3.7500

Query Details

- Query duration: 83ms
- Rows: 1
- Query ID: 01b7675a-0003-42b3-...

Show more **Ask Copilot**

Variant field

Results Chart

	CATEGORY	COST_OF_GOODS_USD	SALE_PRICE_USD	MENU_ITEM_HEALTH_METRICS_OBJ
1		0.6500	3.5000	{ "menu_item_health_metrics": [{ "ingredients": ["Lemons", "Sugar", "Lemonade"] }] }
2		2.5000	6.0000	{ "menu_item_health_metrics": [{ "ingredients": ["One Scoop", "Sugar Cone", "Waffle Cone"] }] }
3		2.5000	6.0000	{ "menu_item_health_metrics": [{ "ingredients": ["One Scoop", "Waffle Cone"] }] }
4		3.0000	7.0000	{ "menu_item_health_metrics": [{ "ingredients": ["Two Scoops", "Ice Cream Sandwich"] }] }
5		0.5000	2.0000	{ "menu_item_health_metrics": [{ "ingredients": ["12 Oz Bottle Water"] }] }
6		0.5000	3.0000	{ "menu_item_health_metrics": [{ "ingredients": ["12 Oz Bottle Soda"] }] }
7		0.7500	3.0000	{ "menu_item_health_metrics": [{ "ingredients": ["12 Oz Bottled Ice Tea"] }] }
8		1.0000	4.0000	{ "menu_item_health_metrics": [{ "ingredients": ["Ice Cream Sandwich"] }] }
9		1.2500	5.0000	{ "menu_item_health_metrics": [{ "ingredients": ["Sweet Mango", "Sticky Salted Rice"] }] }
10		0.5000	3.0000	{ "menu_item_health_metrics": [{ "ingredients": ["Popsicle", "Ice Cream Sandwich"], "is_dairy_free": true, "is_gluten_free": true, "is_healthy_flag": false, "is_nut_free_flag": false}] }

Ask Copilot

```

111
112 v   SELECT menu_item_health_metrics_obj
113   from tasty_bytes_sample_data.raw_pos.menu
114   WHERE truck_brand_name = 'Freezing Point'
115   AND menu_item_name = 'Mango Sticky Rice';
116
117   finish, let's extract the Mango Sticky Rice ingredients from the semi-structured column
118 v   SELECT
119     m.menu_item_name,

```

Results Chart

	MENU_ITEM_HEALTH_METRICS_OBJ
1	{ "menu_item_health_metrics": [{ "ingredients": ["Sweet Mango", "Sticky Salted Rice", "Coconut Milk", "Sesame Seeds"] }] }

Ask Copilot

```

112
113 v   SELECT
114   m.menu_item_name,
115   obj.value:"ingredients":::ARRAY AS ingredients
116   FROM tasty_bytes_sample_data.raw_pos.menu m,
117   LATERAL FLATTEN (input => m.menu_item_health_metrics_obj:menu_item_health_metrics) obj
118   WHERE 1=1
119   AND truck_brand_name = 'Freezing Point'
120   AND menu_item_name = 'Mango Sticky Rice';

```

Results Chart

	MENU_ITEM_NAME	INGREDIENTS
1	Mango Sticky Rice	["Sweet Mango", "Sticky Salted Rice", "Coconut Milk", "Sesame Seeds"]

Query Details

Query duration 91ms

Rows 1

Query ID 01b2

Ask Copilot

The screenshot shows the Snowflake Query History interface. On the left, there's a sidebar with navigation links like Home, Search, Projects, Data, Data Products, AI & ML, Monitoring, and Query History (which is selected). The main area is titled "Query History" and displays a table of 19 queries. The columns are SQL TEXT, QUERY ID, STATUS, USER, and WAREHOUSE. All queries are listed as "Success".

SQL TEXT	QUERY ID	STATUS	USER	WAREHOUSE
1 SELECT menu_item_health_metrics_obj from ...	01b7675f-0003-42a9-0c...	Success	MOHAMEDZUHAIRKA	COMPUTE_WH
2 SELECT m.menu_item_name, obj.value:"ingred...	01b7675e-0003-42b3-0...	Success	MOHAMEDZUHAIRKA	COMPUTE_WH
3 SELECT TOP 10 * FROM tasty_bytes_sample_da...	01b7675a-0003-42a9-0...	Success	MOHAMEDZUHAIRKA	—
4 SELECT menu_item_name, (sale_price_usd - ...)	01b7675a-0003-42b3-0...	Success	MOHAMEDZUHAIRKA	COMPUTE_WH
5 SELECT TOP 10 * FROM tasty_bytes_sample_da...	01b76759-0003-42b2-0...	Success	MOHAMEDZUHAIRKA	—
6 SELECT menu_item_name FROM tasty_bytes_sampl...	01b76759-0003-42b3-0...	Success	MOHAMEDZUHAIRKA	COMPUTE_WH
7 SELECT TOP 10 * FROM tasty_bytes_sample_da...	01b76758-0003-42a9-0...	Success	MOHAMEDZUHAIRKA	COMPUTE_WH
8 SELECT COUNT(*) AS row_count FROM tasty_b...	01b76758-0003-44d3-0...	Success	MOHAMEDZUHAIRKA	—
9 COPY INTO tasty_bytes_sample_data.raw_pos ...	01b76757-0003-44d3-0...	Success	MOHAMEDZUHAIRKA	COMPUTE_WH
10 LIST @tasty_bytes_sample_data.public.blob...	01b76756-0003-44d3-0...	Success	MOHAMEDZUHAIRKA	—

USE ROLE accountadmin;

create database if not exists ecommerce_db;

create schema if not exists ecommerce_dev;

use schema ecommerce_db.ecommerce_dev;

USE WAREHOUSE compute_wh;

create or replace table LINEITEM cluster by (L_SHIPDATE) as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" limit 2000000;

create or replace table ORDERS as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."ORDERS";

```

4
5   create schema if not exists ecommerce_dev;
6
7   use schema ecommerce_db.ecommerce_dev;
8
9   USE WAREHOUSE compute_wh;
10
11  create or replace table LINEITEM cluster by (L_SHIPDATE) as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" limit
12  2000000;
13  create or replace table ORDERS as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."ORDERS";

```

Results Chart

Snowflake executing... 3.6s

Start Time: Oct 2, 11:57 AM
ID: 01b76a43-0003-45e6-0000-000163f7e1c1
Warehouse: COMPUTE_WH
Produced rows: —
Bytes scanned: —

Ask Copilot

```

10
11  create or replace table LINEITEM cluster by (L_SHIPDATE) as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" limit
12  2000000;
13  create or replace table ORDERS as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."ORDERS";

```

Results **Chart**

status	
1	Table LINEITEM successfully created.

Query Details ...
 Query duration 4.7s
 Rows 1
 Query ID 01b2 Ask Copilot
[Show more](#)


```

12
13  create or replace table ORDERS as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."ORDERS";

```

Results **Chart**

Snowflake executing... 9.7s

Start Time	Oct 2, 11:58 AM
ID	01b76a44-0003-45ec-0001-63f700011aa
Warehouse	COMPUTE_WH (X-Small)
Produced rows	—
Bytes scanned	—

Ask Copilot


```

12
13  create or replace table ORDERS as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."ORDERS";

```

Results **Chart**

Snowflake executing... 3m 21s

Start Time	Oct 2, 11:58 AM
ID	01b76a44-0003-45ec-0001-63f700011aa
Warehouse	COMPUTE_WH (X-Small)
Produced rows	432.2M (432,242,688)
Bytes scanned	14.8GB (30%)

Ask Copilot


```

12
13  create or replace table ORDERS as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."ORDERS";

```

Results **Chart**

Snowflake executing... 10m 0s

Start Time	Oct 2, 11:58 AM
ID	01b76a44-0003-45ec-0001-63f700011aa
Warehouse	COMPUTE_WH (X-Small)
Produced rows	1.4B (1,382,285,312)
Bytes scanned	45.3GB (93%)

Ask Copilot


```

12
13  create or replace table ORDERS as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."ORDERS";

```

Results **Chart**

status	
1	Table ORDERS successfully created.

Query Details ...
 Query duration 10m 27s
 Rows 1
 Query ID 01b2 Ask Copilot
[Show more](#)

Screenshot of the Snowflake UI showing the Database Browser and a tooltip for the ORDERS table.

The Database Browser sidebar shows:

- TPCH_SF1
- TPCH_SF10
- TPCH_SF100
- TPCH_SF1000 (selected)
- Tables: CUSTOMER, LINEITEM, NATION, ORDERS (highlighted), PART, PARTSUPP, REGION, SUPPLIER
- TASTY_BYTES_SAMPLE_DATA

The main area shows a query editor with the following code:

```
5 create schema if not exists ecommerce_d
6
7 use schema ecommerce_db.ecommerce_dev;
8
9 USE WAREHOUSE compute_wh;
```

A tooltip for the ORDERS table displays the following details:

ORDERS	
	Definition
Type	Table
Number of rows	1.5B
Size	48.6GB
Cluster Key	LINEAR(O_ORDERDATE)
Owner	—
Created	2 years ago
Comment	Orders data as defined by TPC-H

Search objects

ECommerce_DB

- ECommerce_DEV
 - Tables
 - LINEITEM
 - ORDERS
 - INFORMATION_SCHEMA
 - PUBLIC
- SNOWFLAKE
- SNOWFLAKE_SAMPLE_DATA
- TASTY_BYTES_SAMPLE_DATA

5
6
7
8
9
10
11
12
13

1

```
use role accountadmin;  
use warehouse compute_wh;  
use database ecommerce_db;  
use schema ecommerce_db.ecommerce_dev;  
  
-- Temporary TABLE ---  
create or replace temporary table orders_tmp as select * from  
"ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" limit 50;
```

```
-- TRANSIENT TABLE ---
create or replace transient table orders as select * from
"ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" limit 50;

---- Transient Schema ----
create transient schema transient_schema;
use schema transient_schema;
create or replace table orders as select * from
"ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" limit 50;

---- Transient Database ----
create transient database transient_db;
create schema test_schema;
use database transient_db;
create or replace table orders as select * from
"ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" limit 50;

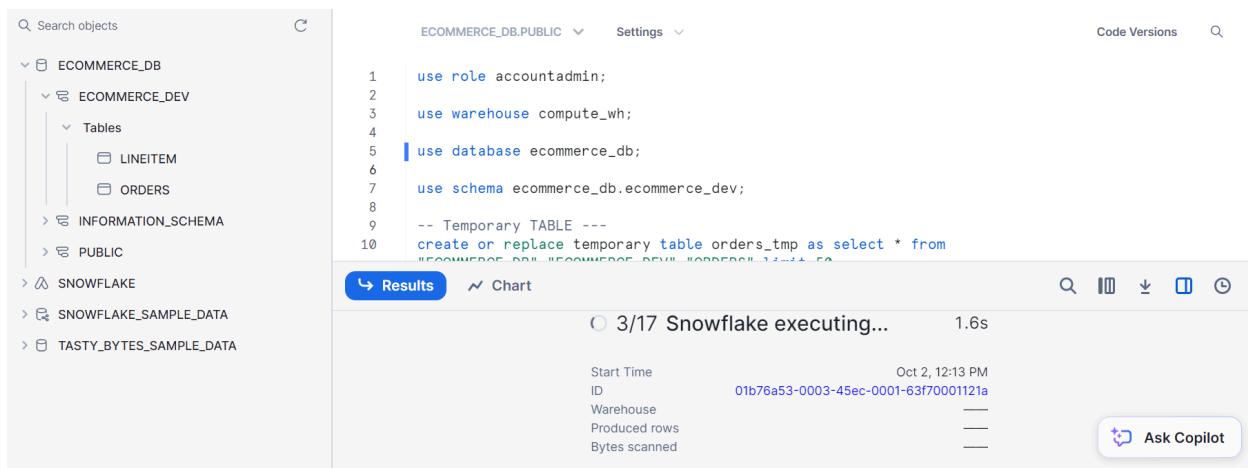
--- Convert Permanent table to transient table ----
create or replace table permanent_orders as select * from
"ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" limit 50;

create or replace transient table transient_orders as select * from permanent_orders limit 50;

drop table permanent_orders;

ALTER TABLE transient_orders RENAME TO permanent_orders;
```

Run all



The screenshot shows the Snowflake UI interface. On the left, there is a sidebar with a search bar and a tree view of databases and schemas. The main area is a code editor with the following SQL code:

```
1 use role accountadmin;
2 use warehouse compute_wh;
3
4 use database ecommerce_db;
5
6 use schema ecommerce_db.ecommerce_dev;
7
8 -- Temporary TABLE ---
9 create or replace temporary table orders_tmp as select * from
"ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" limit 50;
```

The code editor has a 'Results' tab selected at the bottom. The results pane shows the status: "3/17 Snowflake executing..." and a duration of "1.6s". Below the status, there is a table with execution details:

Start Time	Oct 2, 12:13 PM
ID	01b76a53-0003-45ec-0001-63f70001121a
Warehouse	—
Produced rows	—
Bytes scanned	—

At the bottom right of the results pane is a button labeled "Ask Copilot".

Search objects

E COMMERCE_DB

ECOMMERCE_DEV

- Tables
 - LINEITEM
 - ORDERS

INFORMATION_SCHEMA

PUBLIC

SNOWFLAKE

SNOWFLAKE_SAMPLE_DATA

TASTY_BYTES_SAMPLE_DATA

TRANSIENT_DB.PUBLIC

Settings

Code Versions

Results

use role accountadmin;

use warehouse compute_wh;

use database ecommerce_db;

use schema ecommerce_db.ecommerce_dev;

-- Temporary TABLE --

create or replace temporary table orders_tmp as select * from "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS";

11/17 Snowflake executing... 1.6s

Start Time: Oct 2, 12:14 PM

ID: 01b76a54-0003-45ed-0000-000163f7f249

Warehouse:

Produced rows:

Bytes scanned:

Ask Copilot

This screenshot shows the Snowflake UI interface. On the left, there's a sidebar with a tree view of databases and schemas. The current context is 'TRANSIENT_DB.PUBLIC'. The main area shows a code editor with a SQL script. The script creates a temporary table named 'orders_tmp' by selecting all columns from the 'ORDERS' table in the 'ECOMMERCE_DEV' schema. Below the code editor, a progress bar indicates the execution is 1.6s complete. A detailed log table shows the start time as Oct 2, 12:14 PM, and the ID of the execution as 01b76a54-0003-45ed-0000-000163f7f249. The results tab shows a single row with the status 'Statement executed successfully.'

E COMMERCE_DB

ECOMMERCE_DEV

- Tables
 - LINEITEM
 - ORDERS
 - ORDERS_TMP

INFORMATION_SCHEMA

PUBLIC

TRANSIENT_SCHEMA

- Tables
 - ORDERS

use role accountadmin;

use warehouse compute_wh;

use database ecommerce_db;

use schema ecommerce_db.econ

-- Temporary TABLE --

create or replace temporary table "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS_TMP" as select * from "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS";

Results

status

1 Statement executed successfully.

This screenshot shows the Snowflake UI interface. On the left, there's a sidebar with a tree view of databases and schemas. The current context is 'TRANSIENT_SCHEMA'. The main area shows a code editor with a SQL script. The script creates a temporary table named 'ORDERS_TMP' by selecting all columns from the 'ORDERS' table in the 'ECOMMERCE_DEV' schema. Below the code editor, a progress bar indicates the execution is 1.6s complete. A detailed log table shows the start time as Oct 2, 12:14 PM, and the ID of the execution as 01b76a54-0003-45ed-0000-000163f7f249. The results tab shows a single row with the status 'Statement executed successfully.'

```

use role accountadmin;

--- Change the warehouse name if need be ---
use warehouse compute_wh;
use database ecommerce_db;
use schema ecommerce_db.ecommerce_dev;

-- View for Orders with "Urgent" Priority ---
create or replace view urgent_priority_orders as
select * from "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" where
o_orderpriority='1-URGENT';

-- Create a materialized view ---
create or replace materialized view vw_aggregated_orders as
select
    count(1) as total_orders,
    O_ORDERSTATUS as order_status ,
    O_ORDERDATE as order_date
from
    "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS"
where o_orderpriority='1-URGENT'
group by 2,3;

-- Create a secure materialized view ---
create or replace secure materialized view secure_vw_aggregated_orders as
select
    count(1) as total_orders,
    O_ORDERSTATUS as order_status ,
    O_ORDERDATE as order_date
from
    "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS"
where o_orderpriority='1-URGENT'
group by 2,3;

```

```

O_ORDERSTATUS as order_status ,
O_ORDERDATE as order_date
from
    "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS"
where o_orderpriority='1-URGENT'
group by 2,3;

```

---- Add a clustering key to the view

```

ALTER MATERIALIZED VIEW vw_aggregated_orders CLUSTER BY (order_date);
ALTER MATERIALIZED VIEW secure_vw_aggregated_orders CLUSTER BY (order_date);

```

TRANSIENT_DB.PUBLIC ▾ Settings ▾

Code Versions 🔍

```

1 use role accountadmin;
2
3 -- Change the warehouse name if need be ----
4 use warehouse compute_wh;
5 use database ecommerce_db;
6 use schema ecommerce_db.ecommerce_dev;
7
8 -- View for Orders with "Urgent" Priority ---
9 create or replace view urgent_priority_orders as
10 select * from "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" where o_orderpriority='1-URGENT'
11

```

↳ Results **↗ Chart** 🔍 ⏪ ⏴ ⏵ ⏷ ⏸ ⏹

3/8 Snowflake executing... 1.8s

Start Time	Oct 2, 12:45 PM
ID	01b76a73-0003-45ed-0000-000163f7f289
Warehouse	—
Produced rows	—
Bytes scanned	—

Ask Copilot

Materialized view not supported in free version.

```

-- Create a materialized view ---
14 create or replace materialized view vw_aggregated_orders as
15 select
16     count(1) as total_orders,
17     O_ORDERSTATUS as order_status ,
18     O_ORDERDATE as order_date
19
20 from
    "ECOMMERCE_DB"."ECOMMERCE_DEV". "ORDERS"

```

↳ Results **↗ Chart** 🔍 ⏪ ⏴ ⏵ ⏷ ⏸ ⏹

⚠️

000002 (0A000): Unsupported feature 'MATERIALIZED VIEWS'.

Ask Copilot

```
use role accountadmin;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

create or replace role view_role;

grant usage on warehouse compute_wh to role view_role;

grant usage on database ECOMMERCE_DB to role view_role;

grant usage on schema ECOMMERCE_DEV to role view_role;

grant select on
"ECOMMERCE_DB"."ECOMMERCE_DEV"."SECURE_VW_AGGRAGATED_ORDERS" to
role view_role;

grant select on
"ECOMMERCE_DB"."ECOMMERCE_DEV"."URGENT_PRIORITY_ORDERS" to role
view_role;

grant select on "ECOMMERCE_DB"."ECOMMERCE_DEV"."VW_AGGRAGATED_ORDERS"
to role view_role;

grant role view_role to user mohamedzuhairka;

-- Test the GET_DDL using the new role created above ----

use role view_role;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

select * from SECURE_VW_AGGRAGATED_ORDERS limit 20;

select get_ddl('view','URGENT_PRIORITY_ORDERS');
select get_ddl('view','SECURE_VW_AGGRAGATED_ORDERS');
```

SIDDHARTH ACCOUNTADMIN

```

nd database objects
tarting with...
| ECOMMERCE_DB
| SNOWFLAKE_SAMPLE_DATA

```

```

    / grant usage on warehouse COMPUTE_WH to role view_role;
    8 grant usage on database ECOMMERCE_DB to role view_role;
    10 grant usage on schema ECOMMERCE_LIV to role view_role;
    12 grant select on "ECOMMERCE_DB"."ECOMMERCE_LIV"."SECURE_VW_AGGRAGATED_ORDERS" to role view_
    14
    15 grant select on "ECOMMERCE_DB"."ECOMMERCE_LIV"."URGENT_PRIORITY_ORDERS" to role view_
    16
    17 grant select on "ECOMMERCE_DB"."ECOMMERCE_LIV"."VW_AGGRAGATED_ORDERS" to role view_
    18
    19 grant role view_role to user siddharth;
    20
    21 -- Test the GET_DDL using the new role created above -----
    22
    23 use role view_role;
    24
    25 use schema "ECOMMERCE_DB"."ECOMMERCE_LIV";
    26
    27 select * from SECURE_VW_AGGRAGATED_ORDERS limit 20;
    28
    29
    30
    31
    32

```

Role: VIEW_ROLE Change
Warehouse: COMPUTE_WH (XS) On
Database: ECOMMERCE_DB
Schema: ECOMMERCE_LIV

Results Data Preview

Query_ID SQL 43ms 1 rows

Get DDL of normal view (it allows)

```

DIRTY_ORDERS
ATTED_ORDERS
CHEMA
.E_DATA

```

```

    19 grant role view_role to user siddharth;
    20
    21 -- Test the GET_DDL using the new role created above -----
    22
    23 use role view_role;
    24
    25 use schema "ECOMMERCE_DB"."ECOMMERCE_LIV";
    26
    27 select * from URGENT_PRIORITY_ORDERS limit 20;
    28
    29 select get_ddl('view','URGENT_PRIORITY_ORDERS');
    30
    31 select get_ddl('view','SECURE_VW_AGGRAGATED_ORDERS');
    32

```

Results Data Preview

Query_ID SQL 1 rows

Row	GET_DDL('VIEW','URGENT_PRIORITY_ORDERS')
1	create or replace view URGENT_PRIORITY_ORDERS(O_ORDERKEY, O_CUSTKEY, O_ORDERSTATUS, O_TOTALPRICE, O_ORDERDATE, O_ORDERPRIORITY, O_CLERK, O_SHIPPRIORITY, O_C...

Get DDL of materialized view (it is restricted as it doesn't have access)

```

30
31 select get_ddl('view','SECURE_VW_AGGRAGATED_ORDERS');
32

```

Results Data Preview

Query_ID SQL 26ms

SQL compilation error: Object does not exist, or operation cannot be performed.

Snowflake - Partitions

Table Name : PARTSUPP

Clustering Key

Micro-partition-1

PS_PARTKEY	PS_SUPPKEY	PS_AVAILQTY	PS_SUPPLYCOST	PS_COMMENT
51578321	4078327	2356	23.56	Some text
116736079	4078327	5543	23.56	Some text
179236090	4078327	1121	17.00	Some text

Micro-partition-2

PS_PARTKEY	PS_SUPPKEY	PS_AVAILQTY	PS_SUPPLYCOST	PS_COMMENT
196736070	2201164	3223	22.72	Some text
19236090	2201164	3211	24.2	Some text
139877911	2201164	2133	16.89	Some text

Micro-partition-3

PS_PARTKEY	PS_SUPPKEY	PS_AVAILQTY	PS_SUPPLYCOST	PS_COMMENT
24236087	7993204	1433	13.88	Some text
49236091	7993204	1222	20.45	Some text

Snowflake - Partitions

Table Name : PARTSUPP

Micro-partition-1

PS_PARTKEY	51578321	116736079	179236090
PS_SUPPKEY	4078327	4078327	4078327
PS_AVAILQTY	2356	5543	1121
PS_SUPPLYCOST	23.56	21.56	17.00
PS_COMMENT	Some text	Some text	Some text

Micro-partition-2

PS_PARTKEY	51578321	116736079	179236090
PS_SUPPKEY	2201164	2201164	2201164
PS_AVAILQTY	3223	3211	1560
PS_SUPPLYCOST	22.72	24.2	17.00
PS_COMMENT	Some text	Some text	Some text

Micro-partition-3

PS_PARTKEY	51578321	116736079
PS_SUPPKEY	7993204	7993204
PS_AVAILQTY	2356	5543
PS_SUPPLYCOST	23.56	21.56
PS_COMMENT	Some text	Some text

SQL : select ps_partkey,ps_supplycost from partsupp where ps_suppkey='4078327'

Scanned micro-partition = micro-partition-1

Pruned micro-partitions = micro-partition-2 & 3

```

use role accountadmin;

use schema "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000";

use warehouse compute_wh;

--- Check Clustering info in a table ---
select system$clustering_information('LINEITEM');

show tables like '%LINE%'
```

```
select system$clustering_information('PARTSUPP','ps_suppkey');

--- Check Distinct values before selecting the clustering key ---
select count(1),count(distinct ps_suppkey) from
SNOWFLAKE_SAMPLE_DATA.TPCH_SF1000.PARTSUPP;

-- Clear Cache ---
ALTER SESSION SET USE_CACHED_RESULT = FALSE;

---- Run Sql commands----

select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" limit 20000;

select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" where
l_shipdate='1998-12-01';

select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" where
l_shipdate in ('1998-12-01','1998-09-20');

select
L_ORDERKEY,L_PARTKEY,L_SUPPKEY
from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" where l_shipdate in
('1998-12-01','1998-09-20');

-- Change Schema ----

use schema "ECOMMERCE_DB"."ECOMMERCE_LIV";

select system$clustering_information('LINEITEM');

--- Check total number of partitions in the table ----

select * from ORDERS limit 2000;

-- Create a 3XL Warehouse ---
ALTER WAREHOUSE "ETL_XL"
SET WAREHOUSE_SIZE = 'XXXLARGE'
AUTO_SUSPEND = 300
AUTO_RESUME = TRUE
MIN_CLUSTER_COUNT = 1
MAX_CLUSTER_COUNT = 1
SCALING_POLICY = 'STANDARD' COMMENT = "";
```

```

--- Re-create the table with a clustering key ----
create or replace table LINEITEM cluster by (L_SHIPDATE)
as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM";

-- Alter Warehouse after the above execution----
ALTER WAREHOUSE "ETL_XL"
SET WAREHOUSE_SIZE = 'MEDIUM'
AUTO_SUSPEND = 300
AUTO_RESUME = TRUE
MIN_CLUSTER_COUNT = 1
MAX_CLUSTER_COUNT = 1
SCALING_POLICY = 'STANDARD'
COMMENT = ";

-- Get credit usage from automatic reclustering ---
select *
from table(
    snowflake.information_schema.automatic_clustering_history
(
    date_range_start=>dateadd(h, -12, current_timestamp)
)
);

-- Disable automatic re-clustering ---
alter table LINEITEM suspend recluster;

```



```

1 use schema "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000";
2
3 use warehouse compute_wh;
4
5 Click Clustering info in a table ---
6 select system$clustering_information('LINEITEM');
7
8 show tables like '%LINE%'
9
10 select system$clustering_information('PARTSUPP', 'ps_suppkey');
11

```

Results		Chart	Query Details
	status		Query duration 44ms
1	Statement executed successfully.		Rows 1
		Query ID 01b7	Ask Copilot

```

5   --- Check Clustering info in a table ---
6   select system$clustering_information('LINEITEM');
7     tables like '%LINE%'
8
9
10  select system$clustering_information('DARTSUPPLIERSUPPLY');

```

↳ Results ↳ Chart

SYSTEM\$CLUSTERING_INFORMATION

1	{ "cluster_by_keys": "LINEAR(L_SHIPDATE)" }
---	---

SYSTEM\$CLUSTERING_INFORMATION('LINEITEM')

```
{
  "cluster_by_keys" : "LINEAR(L_SHIPDATE)",
  "total_partition_count" : 10336,
  "total_constant_partition_count" : 8349,
  "average_overlaps" : 0.6908,
  "average_depth" : 1.4082,
  "partition_depth_histogram" : {
    "00000" : 0,
    "00001" : 8310,
    "00002" : 599,
    "00003" : 844,
  }
}
```

```

7   show tables like '%LINE%'
8
9
10  select system$clustering_information('DARTSUPPLIERSUPPLY');

```

↳ Results ↳ Chart

bytes	owner	retention_time	automatic_clustering	change_tracking	is_external	enable_sch
1 04640		1	ON	OFF	N	N

```

21  ---- Run Sql commands-----
22
23  select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" limit 20000;
24
25

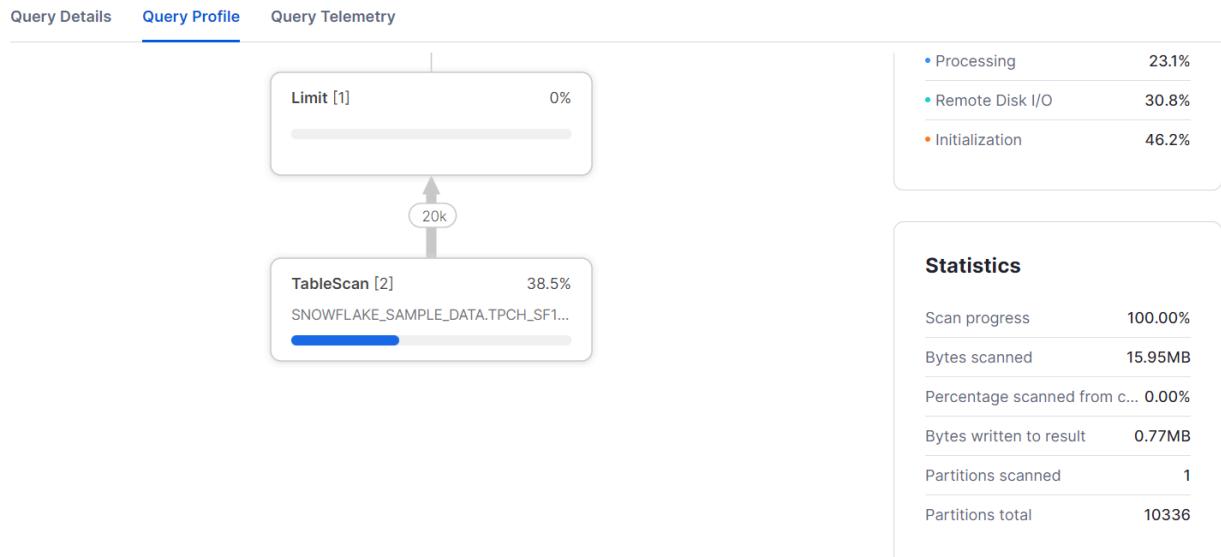
```

↳ Results ↳ Chart

	L_ORDERKEY	L_PARTKEY	L_SUPPKEY	L_LINENUMBER
19995	1964777602	38802111	6302121	3
19996	1965041024	181420052	6420089	3
19997	1964644802	66995971	6995972	6
19998	1964859428	114664136	9664159	3
19999	1964551111	113147025	3147026	4
20000	1964590273	91503383	1503384	5

Query Details ...
 Query duration 1.8s

 Rows 20K
 Query ID 01b77808-0003-a2f5-0...
 Show more
 L_ORDERKEY



```

28
29   select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" where l_shipdate in ('1998-12-01','1998-
30   09-20');
31
32   L_ORDERKEY,L_PARTKEY,L_SUPPKEY

```

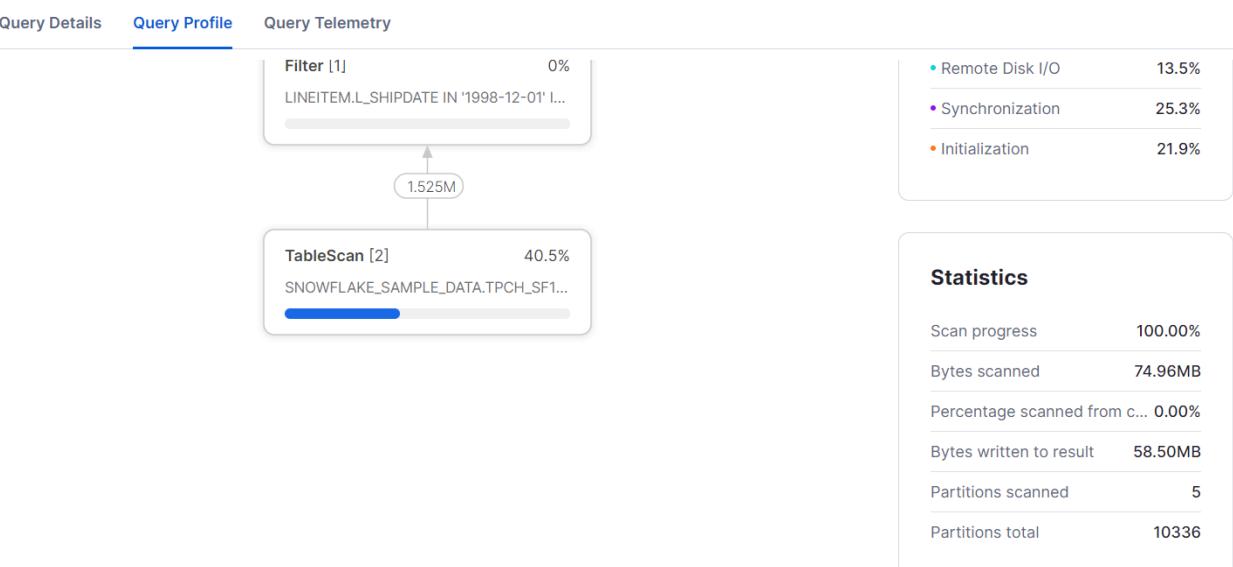
↳ Results ↳ Chart

	L_ORDERKEY	L_PARTKEY	L_SUPPKEY	L_LINENUMBER
624356	5457876579	120155212	2655225	4
624357	5457960417	93096879	8096898	1
624358	5457874403	143464852	3464853	2
624359	5457807440	91702704	1702705	7

Rows 1.5M
Query ID 01b7780e-0003-a21d-0...
Show more

Ask Copilot

COMPUTE_WH MOHAMEDZUHAIRKA



```

10  show tables like '%LINE%'
11
12  select system$clustering_information('PARTSUPP', 'ps_suppkey');
13
14  --- Check Distinct values before selecting the clustering key ---

```

Results **Chart**

SYSTEM\$CLUSTERING_INFORMATION	
1	{ "cluster_by_keys": "LINEAR(PS_SUPPKEY)" }

A SYSTEM\$CLUSTERING_INFORMATION('PARTSUPP','PS_SUPPKEY')

```

{
  "cluster_by_keys" : "LINEAR(PS_SUPPKEY)",
  "notes" : "Clustering key columns contain high cardinality key PS_SUPPKEY which might result in expensive re-clustering. Consider reducing the cardinality of clustering keys. Please refer to https://docs.snowflake.net/manuals/user-guide/tables-clustering-keys.html for more information."
  "total_partition_count" : 2315,
  "total_constant_partition_count" : 0,
  "average_overlaps" : 1.8721,
  "average_depth" : 2.0043,
}

```

Ask Copilot

Here, the cardinality is very high which is not a good thing when talking about clustering.

```

14  --- Check Distinct values before selecting the clustering key,
15  select count(1),count(distinct ps_suppkey) from SNOWFLAKE_SAMPLE_DATA.TPCH_SF1000.PARTSUPP;
16
17  Cache ---

```

Results **Chart**

	COUNT(1)	COUNT(DISTINCT PS_SUPPKEY)
1	800000000	10000000

Query Details
Query duration 12s

From the above query, we can see that it may create 10 million micro-partitions which is not good.

```

74  select *
75  from table(
76    snowflake.information_schema.automatic_clustering_history
77    (
78      date_range_start=>dateadd(h, -12, current_timestamp)
79    )
80  );
81
82  -- Table automatic re-clustering ---
83  alter table LINEITEM suspend recluster;

```

Results **Chart**

START_TIME	END_TIME	CREDITS_USED	NUM_BYTES_RECLUSTERED
Query produced no results			

Query Details
Query duration 537ms
Rows 0
Query ID 01b2

Ask Copilot

```
use role accountadmin;
```

```

use warehouse compute_wh;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

create table lineitem_clone clone lineitem;

alter table lineitem_clone cluster by (L_SUPPKEY,L_SHIPDATE);

```

6 | create table lineitem_clone clone lineitem;

7 | |
8 | alter table lineitem_clone cluster by (L_SUPPKEY,L_SHIPDATE);

9 | alter table lineitem_clone cluster by (L_SUPPKEY,L_SHIPDATE);

status		Query Details	...
1	Table LINEITEM_CLONE successfully created.	Query duration	803ms
		Rows	1
		Query ID	01b77bf0-0003-a2ab-0...

status		Query Details	...
1	Statement executed successfully.	Query duration	180ms
		Rows	1
		Query ID	01b77bf2-0003-45e4-0...

```

-- Retreive history of all the queries ----
select * from table(information_schema.query_history()) order by start_time;

select * from table(information_schema.query_history_by_user()) order by start_time;

select * from table(information_schema.query_history_by_warehouse()) order by start_time;

WITH access_history AS (
  SELECT distinct query_id "QUERY_ID"
    ,split(base.value:objectName, '.')[0]::string "DATABASE_NAME"
    ,split(base.value:objectName, '.')[1]::string "SCHEMA_NAME"
    ,split(base.value:objectName, '.')[2]::string "TABLE_NAME"
  FROM snowflake.account_usage.access_history
  ,lateral flatten (base_objects_accessed) base

```

```

)
select regexp_replace(replace(replace(query_text,char(10),' '),char(13),' ',' *',' ') query_text
 ,count(*) query_count
 ,(avg(execution_time) / 1000) / 60 avg_exec_time_mins
 ,avg((nullifzero(PARTITIONS_SCANNED) / nullifzero(PARTITIONS_TOTAL))) * 100
 "avg_%_partitions_scanned"
from snowflake.account_usage.query_history qh
join access_history ah
on qh.query_id = ah.query_id
where to_date(start_time) >= dateadd('months', -1, current_date())
//and ah.database_name = 'snowflake_sample_data'
//and ah.table_name = <TABLE_NAME>
and error_code is null
//and (query_text ilike '%where%' or query_text ilike '%join%')
and warehouse_size is not null
group by 1
//having avg_exec_time_mins > 0.1
order by query_count desc;

```

2 -- Retreive history of all the queries ----

3 select * from table(information_schema.query_history()) order by start_time;

4

5 select * from table(information_schema.query_history_by_user()) order by start_time;

↳ Results ↵ Chart

	WAREHOUSE_NAME	WAREHOUSE_SIZE	WAREHOUSE_TYPE	CLUSTER_NUMBER	QUERY_ID
1	IN	COMPUTE_WH	null	STANDARD	null
2	IN	COMPUTE_WH	X-Small	STANDARD	1
3	IN	COMPUTE_WH	null	STANDARD	null
4	IN	COMPUTE_WH	X-Small	STANDARD	1
5	IN	COMPUTE_WH	X-Small	STANDARD	1
6		null	STANDARD		01b76759-0003-42b2-0000-000163f73025

Ask Copilot

```

5  select * from table(information_schema.query_history_by_user()) order by start_time;
6
7  select * from table(information_schema.query_history_by_warehouse()) order by start_time;
8
9

```

↳ Results ↗ Chart

		SCHEMA_NAME	QUERY_TYPE	SESSION_ID	USER_NAME
1	IPLE_DATA	RAW_POS	CREATE	5972131845	MOHAMEDZUHAIRKA
2	IPLE_DATA	RAW_POS	LIST_FILES	5972131845	MOHAMEDZUHAIRKA
3	IPLE_DATA	RAW_POS	COPY	5972131845	MOHAMEDZUHAIRKA
4	IPLE_DATA	RAW_POS	SELECT	5972131845	MOHAMEDZUHAIRKA
5	IPLE_DATA	RAW_POS	SELECT	5972131845	MOHAMEDZUHAIRKA
6	IPLE_DATA	RAW_POS	SELECT	5972131845	MOHAMEDZUHAIRKA

Query Details

- Query duration 649ms
- Rows 100
- Query ID 01b77bf8-0003-a2ab-0...

Show more ▾

QUERY_ID Ask Copilot 100% filled

```

6
7  select * from table(information_schema.query_history_by_warehouse()) order by start_time;
8
9
10 WITH access_history AS (
    SELECT ...
)

```

↳ Results ↗ Chart

QUERY_ID	QUERY_TEXT	
1	01b76740-0003-42b5-0000-000163f7a009	USE ROLE accountadmin;
2	01b76741-0003-42b4-0000-000163f72011	USE WAREHOUSE compute_wh;
3	01b76751-0003-42b0-0000-000163f75019	CREATE OR REPLACE DATABASE tasty_bytes_sample;
4	01b76751-0003-42ae-0000-000163f78019	CREATE OR REPLACE SCHEMA tasty_bytes_sample;
5	01b76752-0003-42b0-0000-000163f75021	CREATE OR REPLACE TABLE tasty_bytes_sample_d...
6	01b76754-0003-42ae-0000-000163f78021	CREATE OR REPLACE TABLE tasty_bytes_sample_d...

Query Details

- Query duration 328ms
- Rows 83
- Query ID 01b77bfa-0003-45e7-0...

Show more ▾

QUERY_ID Ask Copilot 100% filled

```

25   JOIN access_history ah
26   ON qh.query_id = ah.query_id
27   WHERE TO_DATE(start_time) >= DATEADD('months', -1, CURRENT_DATE())
28   //AND ah.database_name = 'snowflake_sample_data'
29   //AND ah.table_name = <TABLE_NAME>
30   AND error_code IS NULL
31   //AND (query_text ILIKE '%WHERE%' OR query_text ILIKE '%JOIN%')
32   AND warehouse_size IS NOT NULL
33   GROUP BY 1
34   //HAVING avg_exec_time_mins > 0.1
35   ORDER BY query_count DESC;

```

↳ Results ↗ Chart

QUERY_TEXT	QUERY_COUNT	AVG_EXEC_TIME_MINS	avg_%partitions_sca...
Query produced no results			

Query Details

- Query duration 2.9s
- Rows 0
- Query ID 01b77bfa-0003-45e7-0...

Ask Copilot

Caching example:

```
use role accountadmin;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

-- Metadata caching ----
select count(1) from lineitem ;

select min(L_ORDERKEY),max(L_ORDERKEY) from lineitem;

-- Results Data caching : Example Query ----

set current_dt='1994-01-01';
select top 10 * from lineitem;
select
    sum(l_quantity) as sum_qty,
    count(*) as order_count,
    date(l_shipdate) as shipped_date,
    l_shipmode as shipped_mode
from
    LINEITEM
where
    shipped_date >= date($current_dt) - 7
group by
    shipped_date,
    shipped_mode
order by
    shipped_date;

-- Clear Result Cache ---
ALTER SESSION SET USE_CACHED_RESULT = FALSE;

-- Clear Data Cache ---

alter warehouse compute_wh suspend;

alter warehouse compute_wh resume;
```

First time execution:

```
6  -- Metadata caching ----
7  | select count(1) from lineitem ;
8
9  select min(L_ORDERKEY),max(L_ORDERKEY) from lineitem;
10
11
```

↳ Results ~ Chart

	COUNT(1)	Rows
1	2000000	1

Query ID: 01b77c07-0003-a21e-0001-63f70001d0e6

Query - 01b77c07-0003-a21e-0001-63f70001d0e6

MOHAMEDZUHAIRKA

Details Query Profile Query Telemetry



METADATA-BASED RESULT [0] 100%

```
6  -- Metadata caching ----
7  | select count(1) from lineitem ;
8
9  select min(L_ORDERKEY),max(L_ORDERKEY) from lineitem;
10
11
```

↳ Results ~ Chart

	COUNT(1)	Rows
1	2000000	1

Query ID: 01b77c09-0003-a363-0...

Query - 01b77c09-0003-a363-0001-63f70001f026

MOHAMEDZUHAIRKA

Query Details **Query Profile** Query Telemetry

[] + -

QUERY RESULT REUSE [0] 100%

01b77c09-0003-a363-0001-63f70001f...

9	select min(L_ORDERKEY),max(L_ORDERKEY) from lineitem;
10	
11	

↳ Results ↵ Chart

Query Details ...
Query duration 172ms

	MIN(L_ORDERKEY)	MAX(L_ORDERKEY)
1	7973	5999999523

Query - 01b77c0b-0003-45e3-0001-63f700010e22

MOHAMEDZUHAIRKA

Query Details **Query Profile** Query Telemetry

[] + -

METADATA-BASED RESULT [0] 100%

This results also coming from metadata cache layer.

Lets see the actual query result from disk

First clear query results cache and warehouse cache:

```
36 -- Clear Data Cache ---  
37  
38 alter warehouse compute_wh suspend;  
39  
40 alter warehouse compute_wh resume;
```

The screenshot shows the Snowflake UI with three tabs: 'Results' (selected), 'Chart', and 'Query Details'. The 'Results' tab displays a single row with the status 'Statement executed successfully.' The 'Query Details' panel on the right shows a green progress bar for 'Query duration' at 71ms and 1 row processed.

```
59  
40 alter warehouse compute_wh resume;
```

The screenshot shows the Snowflake UI with three tabs: 'Results' (selected), 'Chart', and 'Query Details'. The 'Results' tab displays a single row with the status 'Statement executed successfully.' The 'Query Details' panel on the right shows a green progress bar for 'Query duration' at 55ms and 1 row processed.

```
51 -- Clear Result Cache ---  
52  
53 ALTER SESSION SET USE_CACHED_RESULT = FALSE;  
54  
55
```

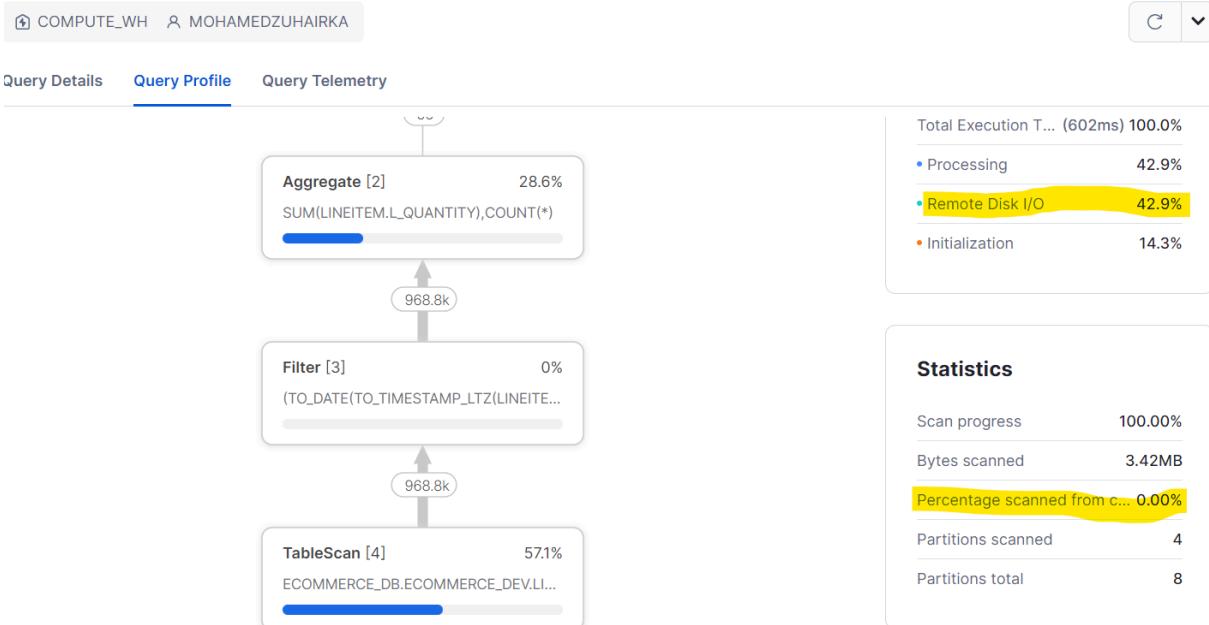
The screenshot shows the Snowflake UI with three tabs: 'Results' (selected), 'Chart', and 'Query Details'. The 'Results' tab displays a single row with the status 'Statement executed successfully.' The 'Query Details' panel on the right shows a green progress bar for 'Query duration' at 45ms and 1 row processed.

```
16 select  
17     sum(l_quantity) as sum_qty,  
18     count(*) as order_count,  
19     date(l_shipdate) as shipped_date,  
20     l_shipmode as shipped_mode  
21 from  
22     LINEITEM  
23 where  
24     shipped_date >= date($current_dt) - 7  
25 group by  
26     shipped_date,
```

The screenshot shows the Snowflake UI with three tabs: 'Results' (selected), 'Chart', and 'Query Details'. The 'Results' tab displays a table with four columns: SUM_QTY, ORDER_COUNT, SHIPPED_DATE, and SHIPPED_MODE. The 'Query Details' panel on the right shows a green progress bar for 'Query duration' at 836ms and 56 rows processed. A button for 'Ask Copilot' is visible in the bottom right corner.

	SUM_QTY	ORDER_COUNT	SHIPPED_DATE	SHIPPED_MODE
1	273626.00	10783	1994-03-31	TRUCK
2	279217.00	11022	1994-03-31	REG AIR
3	279541.00	11057	1994-03-31	FOB
4	277205.00	10845	1994-03-31	SHIP

Query - 01b77c18-0003-a1d1-0001-63f70001b13e



Reexecute same query:

```

16   select
17       sum(l_quantity) as sum_qty,
18       count(*) as order_count,
19       date(l_shipdate) as shipped_date,
20       l_shipmode as shipped_mode
21   from
22       LINEITEM
23   where
24       shipped_date >= date($current_dt) - 7
25   group by
26       shipped_date,

```

Results Chart

	SUM_QTY	ORDER_COUNT	SHIPPED_DATE	SHIPPED_MODE
1	273626.00	10783	1994-03-31	TRUCK
2	279217.00	11022	1994-03-31	REG AIR
3	279541.00	11057	1994-03-31	FOB
A	277705.00	10815	1994-03-31	CWID

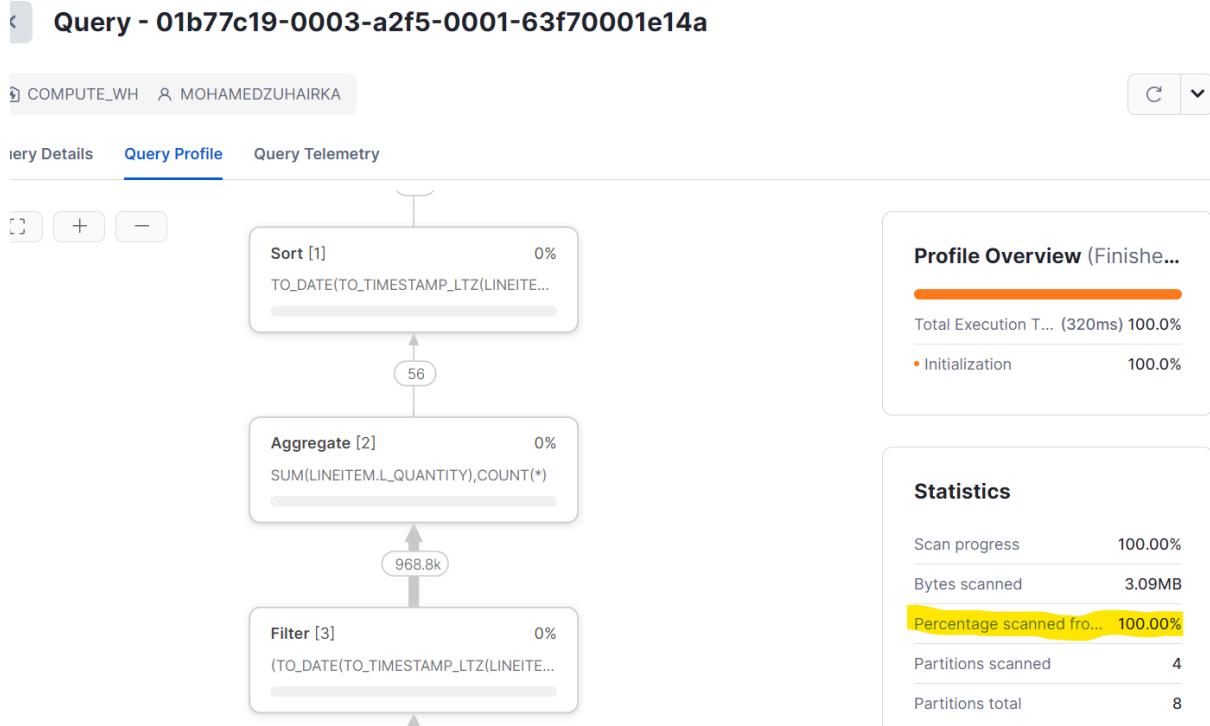
Query Details

Query duration: 362ms

Rows: 56

Query ID: 01b77c18-0003-a1d1-0001-63f70001b13e

Ask Copilot



Search Optimization:

```

use role accountadmin;

create or replace database test_db_so;
create or replace schema test_schema;
use schema test_db_so.test_schema;

-- Create a table with no Search optimization feature enabled ----
create or replace table lineitem_no_so clone
"ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM";
-- OR --
-- create or replace table lineitem_no_so cluster by(l_shipdate) as select * from
"SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM";

-- Create a table and enable the Search optimization feature ----
create or replace table lineitem_so clone
"ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM";

select * from lineitem_no_so where l_orderkey = 2412266214;
select system$ESTIMATE_SEARCH_OPTIMIZATION_COSTS('lineitem_no_so');

```

```

alter table lineitem_no_so add search optimization;
show tables like "%lineitem_no_so%";

-- Point lookup query ---
select * from lineitem_no_so where l_orderkey='2412266214' limit 10;

--- Drop search optimization ---
alter table lineitem_no_so drop search optimization;

-- Clear Result and WH Cache ---
ALTER SESSION SET USE_CACHED_RESULT = FALSE;
alter warehouse prod_xl suspend;
alter warehouse prod_xl resume;

```

The screenshot shows a Snowflake session interface. The top navigation bar includes 'ECOMMERCE_DB.PUBLIC' and 'Settings'. The main area displays a multi-line SQL script:

```

5  create or replace schema test_schema;
6
7  use schema test_db_so.test_schema;
8
9  -- Create a table with no Search optimization feature enabled ----
10 create or replace table lineitem_no_so clone "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM";
11 -- OR ---
12 -- create or replace table lineitem_no_so cluster by(l_shipdate) as select * from
13   "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM";
14

```

Below the code, the results tab is selected, showing the status: '3/6 Snowflake executing...' and a duration of '1.4s'. The execution details panel shows:

- Start Time: Oct 5, 4:15 PM
- ID: 01b77c25-0003-a2ab-0001-63f70001a18a
- Warehouse: (unspecified)
- Produced rows: (unspecified)
- Bytes scanned: (unspecified)

A 'Ask Copilot' button is located in the bottom right corner.

Point lookup query on non-cluster key:

The screenshot shows a Snowflake session interface. The top navigation bar includes 'ECOMMERCE_DB.PUBLIC' and 'Settings'. The main area displays a single-line SQL query:

```

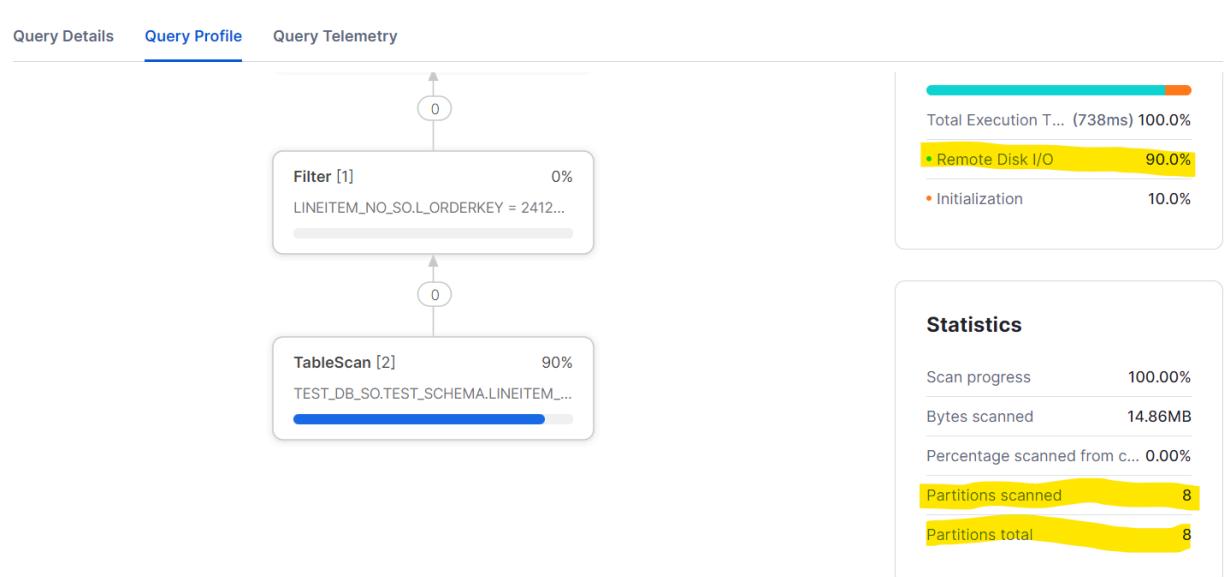
1/ 18  select * from lineitem_no_so where l_orderkey = 2412266214;

```

Below the code, the results tab is selected, showing the status: '3/6 Snowflake executing...' and a duration of '1.4s'. The execution details panel shows:

- Start Time: Oct 5, 4:15 PM
- ID: 01b77c25-0003-a2ab-0001-63f70001a18a
- Warehouse: (unspecified)
- Produced rows: 0
- Bytes scanned: 0

The results table has columns: L_ORDERKEY, L_PARTKEY, L_SUPPKEY, L_LINENUMBER, L_QUANTITY. A message in the table body says 'Query produced no results'. A 'Show more' button is available in the bottom right corner. A 'Ask Copilot' button is located in the bottom right corner.



```
19 | select system$ESTIMATE_SEARCH_OPTIMIZATION_COSTS('lineitem_no_so');
20 |
```

↳ Results ↵ Chart

SYSTEM\$ESTIMATE_SEARCH_OPTIMIZATION_COSTS('lineitem_no_so')

```
1 { "tableName": "LINEITEM_NO_SO",
```

Ask Copilot

After search optimization enabled:

```

19 alter table lineitem_so add search optimization;
20
21 show tables like '%lineitem_so%';
22
23 //create or replace table lineitem_no_so cluster by(l_shipdate) as select * from "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM";
24

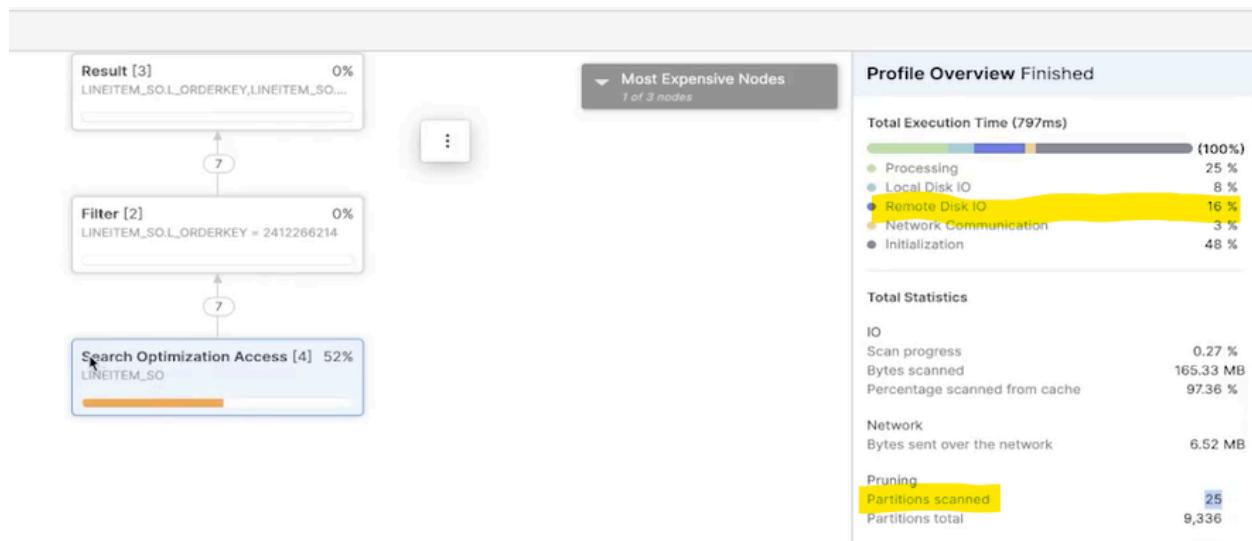
```

Results Data Preview Open History

Query_ID	SQL	77ms	1 rows

Filter result... Copy Columns ▾

nt	cluster_by	rows	bytes	owner	retention_time	automatic_clust	change_tracking	search_optimiza	search_optimization_pr	search_optimiza	is_extern
LINEAR(L_S...	5999989709	1570955863...	SYSADMIN	1	OFF	OFF	ON	100	49954499584	N	



Data ingestion:

Connection setup and load csv file into snowflake:

```
-- Step-1 ---- : Create an IAM Role in AWS
```

```
use role accountadmin;
```

```
-- STEP-2 : Create an integration object -----
```

```
CREATE STORAGE INTEGRATION aws_sf_data
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN =
'arn:aws:iam::682470999581:role/aws-saml-bsys-ems-dev-developer'
STORAGE_ALLOWED_LOCATIONS = ('s3://ems-pt-result/ecommerce_dev/');
```

```
desc INTEGRATION aws_sf_data;
```

```

-- Grant usage access to Sysadmin Role ---
grant usage on integration aws_sf_data to role sysadmin;

-- Grant access to Sysadmin Role ---
grant select on all tables in schema "ECOMMERCE_DB"."ECOMMERCE_DEV" to role
sysadmin;

use role sysadmin;

use database ecommerce_db;

create schema ecommerce_dev;

create or replace table lineitem cluster by (L_SHIPDATE) as select * from
"ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM" limit 1;
truncate table lineitem;

CREATE FILE FORMAT csv_load_format
  TYPE = 'CSV'
  COMPRESSION = 'AUTO'
  FIELD_DELIMITER = ','
  RECORD_DELIMITER = '\n'
  SKIP_HEADER =1
  FIELD_OPTIONALLY_ENCLOSED_BY = '\042'
  TRIM_SPACE = FALSE
  ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE
  ESCAPE = 'NONE'
  ESCAPE_UNENCLOSED_FIELD = '\134'
  DATE_FORMAT = 'AUTO'
  TIMESTAMP_FORMAT = 'AUTO';

-- Create a stage for lineitem table ---
create stage stg_lineitem_csv_dev
storage_integration = aws_sf_data
url = '{your_bucket_name}/ecommerce_dev/lineitem/lineitem_csv/'
file_format = csv_load_format;

list @stg_lineitem_csv_dev;

copy into lineitem from @stg_lineitem_csv_dev ON_ERROR = ABORT_STATEMENT;

-- Validate the data---
select * from lineitem limit 10;

```

Amazon S3 > Buckets > ems-pt-result > ecommerce_dev/

ecommerce_dev/

Objects | Properties

Objects (1) Info
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

lineitem/

-- STEP-2 : Create an integration object -----
CREATE STORAGE INTEGRATION aws_sf_data
TYPE = EXTERNAL_STAGE
STORAGE_PROVIDER = S3
ENABLED = TRUE
STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::██████████:role/aws-saml-bsys-ems-dev-developer'
STORAGE_ALLOWED_LOCATIONS = ('s3://ems-pt-result/ecommerce_dev/');
desc INTEGRATION aws_sf_data;

Results | **Chart**

status		Query Details
1	Integration AWS_SF_DATA successfully created.	Query duration 363ms

13 desc INTEGRATION aws_sf_data;
14

Results | **Chart**

	property_type	property_value	Query Details
3	CATIONS	List s3://ems-pt-result/ecommerce_dev/	Query duration 295ms
4	CATIONS	List	Rows 8
5	R_ARN	arn:aws:iam::180294178125:user/w57r0000-s	Query ID 01b77c5d-0003-a363-0...
6	N	arn:aws:iam::██████████:role/aws-saml-bsys-ems-dev-developer	Show more
7	AL_ID	JXB38456_SFCRole=5_Wq9S3GUacW7M1WtildaC1lzAdGM=	property Ask Copilot
8		String	

14 -- Grant usage access to Sysadmin Role ---
15
16 grant usage on integration aws_sf_data to role sysadmin;
17

Results | **Chart**

status		Query Details
1	Statement executed successfully.	Query duration 82ms

```
18 use database ecommerce_db;
19 create schema ecommerce_liv;
20 -- Grant create stage access to Sysadmin Role ---
21 grant create stage on schema "ECOMMERCE_DB"."ECOMMERCE_LIV" to role sysadmin;
```

↳ Results ↵ Chart

🔍 ⏪ ⏴ ⏵ ⏷ ⏸

	status
1	Statement executed successfully.

Query Details ...
Query duration 57ms
Rows 1
Query ID 01b77c66-0003-a21d-0...

```
32 -- Create a file format ---
33 CREATE FILE FORMAT csv_load_format
34   TYPE = 'CSV'
35   COMPRESSION = 'AUTO'
36   FIELD_DELIMITER = ','
37   RECORD_DELIMITER = '\n'
38   SKIP_HEADER =1
39   FIELD_OPTIONALLY_ENCLOSED_BY = '\042'
40   TRIM_SPACE = FALSE
41   ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE
```

↳ Results ↵ Chart

🔍 ⏪ ⏴ ⏵ ⏷ ⏸

	status
1	File format CSV_LOAD_FORMAT successfully created.

Query Details ...
Query duration 123ms
Rows 1

```
24
25 -- Create a stage for lineitem table ---
26 create stage stg_lineitem_csv_dev
27 storage_integration = aws_sf_data
28 url = 's3://sid-snowflake-data/commerce_dev/lineitem/lineitem_csv/'
29 file_format = csv_load_format;]
```

30

31

Results Data Preview

✓ Query ID SQL 776ms 1 rows

Filter result...



Copy

Row status

1 Stage area STG_LINEITEM_CSV_DEV successfully created.

30
31 list @stg_lineitem_csv_dev;
32
33

Results Data Preview ← Open Hist

✓ Query ID SQL 775ms 73 rows

Filter result... Copy Columns ▾

Row	name	size	md5	last_modified
64	s3://sid-snowflake-data/e-commerce_dev/lineitem/lineitem_csv/data_8_0_0.csv.gz	1625898	e5f65be2d25267c3051734...	Fri, 11 Feb 2022 12:46:03 G...
65	s3://sid-snowflake-data/e-commerce_dev/lineitem/lineitem_csv/data_8_2_0.csv.gz	695699	1f2db6e75711b8f6dde44e7...	Fri, 11 Feb 2022 12:46:03 G...
66	s3://sid-snowflake-data/e-commerce_dev/lineitem/lineitem_csv/data_8_3_0.csv.gz	1563517	8f2e7bf32dc5aa98db6efe7...	Fri, 11 Feb 2022 12:46:03 G...
67	s3://sid-snowflake-data/e-commerce_dev/lineitem/lineitem_csv/data_8_4_0.csv.gz	31358	a1d378c76b47fc09038e65...	Fri, 11 Feb 2022 12:46:03 G...
68	s3://sid-snowflake-data/e-commerce_dev/lineitem/lineitem_csv/data_8_5_0.csv.gz	1335599	75998a3cd50e3070eb9e16...	Fri, 11 Feb 2022 12:46:03 G...
69	s3://sid-snowflake-data/e-commerce_dev/lineitem/lineitem_csv/data_9_0_0.csv.gz	152431	7d8fa7109944b8fce144400...	Fri, 11 Feb 2022 12:46:03 G...
70	s3://sid-snowflake-data/e-commerce_dev/lineitem/lineitem_csv/data_9_2_0.csv.gz	475826	18aee6efaaa545cc5702ae4...	Fri, 11 Feb 2022 12:46:03 G...

```
32
33 copy into lineitem
34 from @stg_lineitem_csv_dev
35 ON_ERROR = ABORT_STATEMENT;
36
37
```

Results Data Preview ← Open History

✓ Query ID SQL 4.36s 73 rows

Filter result... Copy Columns ▾

Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_charac	first_error_column
1	s3://sid-snow...	LOADED	16127	16127	1	0	NULL	NULL	NULL	NULL
2	s3://sid-snow...	LOADED	12031	12031	1	0	NULL	NULL	NULL	NULL
3	s3://sid-snow...	LOADED	255	255	1	0	NULL	NULL	NULL	NULL
4	s3://sid-snow...	LOADED	767	767	1	0	NULL	NULL	NULL	NULL
5	s3://sid-snow...	LOADED	3839	3839	1	0	NULL	NULL	NULL	NULL
6	s3://sid-snow...	LOADED	12031	12031	1	0	NULL	NULL	NULL	NULL
7	s3://sid-snow...	LOADED	7935	7935	1	0	NULL	NULL	NULL	NULL

Load json file into snowflake:

```
use role sysadmin;

use schema ecommerce_db.ecommerce_liv;

CREATE OR REPLACE FILE FORMAT json_load_format TYPE = 'JSON' ;

-- Create a stage for lineitem table ---
create stage stg_lineitem_json_dev
storage_integration = aws_sf_data
url = '{your_bucket_name}/ecommerce_dev/lineitem/lineitem_json/'
file_format = json_load_format;

-- List all the files to check before loading ---
list @stg_lineitem_json_dev;
```

```

-- Select the data directly from staged location to validate the data before loading ---
select $1 from @stg_lineitem_json_dev limit 10;

--- #####First approach#####
-- insert into lineitem directly from staged location ---
insert into lineitem
select
    $1:L_ORDERKEY,
    $1:L_PARTKEY,
    $1:L_SUPPKEY,
    $1:L_LINENUMBER,
    $1:L_QUANTITY,
    $1:L_EXTENDEDPRICE,
    $1:L_DISCOUNT,
    $1:L_TAX,
    $1:L_RETURNFLAG::varchar,
    $1:L_LINESSTATUS::varchar,
    $1:L_SHIPDATE::varchar,
    $1:L_COMMITDATE::varchar,
    $1:L_RECEIPTDATE::varchar,
    $1:L_SHIPINSTRUCT::varchar,
    $1:L_SHIPMODE::varchar,
    $1:L_COMMENT
from
    @stg_lineitem_json_dev ;
-- limit 10;

truncate table lineitem;

--- #####Second approach#####
-- Create a raw table with variant datatype column ---
create table lineitem_raw_json (src variant );

-- Ingest data into the raw table from staged location ---
copy into lineitem_raw_json from @stg_lineitem_json_dev ON_ERROR =
ABORT_STATEMENT;

-- insert into the structured table from raw table ---
insert into lineitem
select
    SRC:L_ORDERKEY,
    SRC:L_PARTKEY,
    SRC:L_SUPPKEY,
    SRC:L_LINENUMBER,
    SRC:L_QUANTITY,
    SRC:L_EXTENDEDPRICE,
    SRC:L_DISCOUNT,
    SRC:L_TAX,

```

```

SRC:L_RETURNFLAG,
SRC:L_LINESTATUS,
SRC:L_SHIPDATE,
SRC:L_COMMITDATE,
SRC:L_RECEIPTDATE,
SRC:L_SHIPINSTRUCT,
SRC:L_SHIPMODE,
SRC:L_COMMENT
from
lineitem_raw_json ;
-- limit 10;

```

The screenshot shows the Snowflake SQL interface. A code block at the top creates a stage area named `stg_lineitem_json_dev` from an S3 bucket. Below it, a results table shows a single row indicating the stage area was successfully created.

Row	status
1	Stage area STG_LINEITEM_JSON_DEV successfully created.

The screenshot shows two queries. The first query attempts to copy data from the stage area into a table named `lineitem`, but it fails due to a JSON file format error. The second query, which uses a placeholder `$1` for the stage area name and includes a `limit 10` clause, executes successfully and returns 10 rows of JSON data.

Row	\$1
1	{ "L_COMMENT": " express d", "L_COMMITDATE": "1997-02-12", "L_DISCOUNT": 0.06, "L_EXTENDEDPRICE": 52707.55, "L_LINENUMBER": 4, "L_LINESTATUS": "O", "L_ORDERKEY": 59...
2	{ "L_COMMENT": "ages behind the fi", "L_COMMITDATE": "1996-11-27", "L_DISCOUNT": 0.08, "L_EXTENDEDPRICE": 78476.64, "L_LINENUMBER": 1, "L_LINESTATUS": "O", "L_ORDERK...
3	{ "L_COMMENT": "nto beans n", "L_COMMITDATE": "1997-01-25", "L_DISCOUNT": 0.05, "L_EXTENDEDPRICE": 67736.64, "L_LINENUMBER": 3, "L_LINESTATUS": "O", "L_ORDERKEY": 5...
4	{ "L_COMMENT": "equests packages are slyly", "L_COMMITDATE": "1996-12-05", "L_DISCOUNT": 0.07, "L_EXTENDEDPRICE": 34069.2, "L_LINENUMBER": 4, "L_LINESTATUS": "O", "L...

Since Json is semi structured, load into structure format may be throw the compilation error.

To load json, first approach, select raw data with key fields from stage and insert into structured table. But the values will be in double quotes.

```

25   $1:L_SUPPKEY,
26   $1:L_LINENUMBER,
27   $1:L_QUANTITY,
28   $1:L_EXTENDEDPRICE,
29   $1:L_DISCOUNT,
30   $1:L_TAX,
31   $1:L_RETURNFLAG,
32   $1:L_LINESSTATUS,
33   $1:L_SHIPDATE,
34   $1:L_COMMITDATE,
35   $1:L_RECEIPTDATE,
36   $1:L_SHIPINSTRUCT,
37   $1:L_SHIPMODE,
38   $1:L_COMMENT
39   from
40   @stg_lineitem_json_dev
41 limit 10;

```

Results Data Preview Open History

✓ Query ID SQL 623ms 10 rows

Filter result...	Copy	Columns ▾									
NTITY	\$1:L_EXTENDED	\$1:L_DISCOUNT	\$1:L_TAX	\$1:L_RETURNFL	\$1:L_LINESSTATU	\$1:L_SHIPDATE	\$1:L_COMMITD	\$1:L_RECEIPTDA	\$1:L_SHIPINSTR	\$1:L_SHIPMODE	\$1:L_COMMENT
60616.92	0.04	0.04	"N"	"QD"	"1997-09-24"	"1997-10-11"	"1997-10-05"	"DELIVER IN ..."	"FOB"	"sly agains..."	
21057.4	0.1	0.02	"N"	"O"	"1997-09-24"	"1997-10-17"	"1997-10-08"	"TAKE BACK..."	"MAIL"	"y permanen..."	
23923.06	0.06	0.07	"N"	"O"	"1997-09-24"	"1997-10-20"	"1997-10-23"	"NONE"	"AIR"	"ockey playe..."	
11749.32	0	0.02	"N"	"O"	"1997-09-24"	"1997-08-24"	"1997-10-14"	"COLLECT C..."	"AIR"	"iously expre..."	

We can remove it by referring type as varchar:

```

Run All Queries | Saved 3 seconds ago
SYSADMIN COMPUTE_WH (XS) ECOMMERCE_DB ECOMMERCE_DEV ...
26
27   $1:L_QUANTITY,
28   $1:L_EXTENDEDPRICE,
29   $1:L_DISCOUNT,
30   $1:L_TAX,
31   $1:L_RETURNFLAG::varchar,
32   $1:L_LINESSTATUS::varchar,
33   $1:L_SHIPDATE::varchar,
34   $1:L_COMMITDATE::varchar,
35   $1:L_RECEIPTDATE::varchar,
36   $1:L_SHIPINSTRUCT::varchar,
37   $1:L_SHIPMODE::varchar

```

Results Data Preview Open History

✓ Query ID SQL 303ms 10 rows

Filter result...	Copy	Columns ▾									
NTITY	\$1:L_EXTENDED	\$1:L_DISCOUNT	\$1:L_TAX	\$1:L_RETURNFL	\$1:L_LINESSTATU	\$1:L_SHIPDATE	\$1:L_COMMITD	\$1:L_RECEIPTDA	\$1:L_SHIPINSTR	\$1:L_SHIPMODE	\$1:L_COMMENT
58021.92	0.05	0.05	N	O	1998-07-28	1998-10-02	NULL	NULL	FOB	lites haggle f...	
40654.3	0.08	0	N	O	1998-07-28	1998-07-20	NULL	COLLECT C...	FOB	foxes. re	
32937.58	0.08	0.06	N	O	1998-07-28	1998-08-24	NULL	NULL	FOB	r accounts. q...	
49671.84	0.05	0.05	N	O	1998-07-28	1998-07-28	NULL	NULL	TRUCK	unts use styl...	
55942.52	0.07	0	N	O	1998-07-28	1998-07-05	NULL	NULL	REG AIR	efully even	

To load json, second approach, first create new table and store the raw data into it. Then from new table, transform it and store it in structured table.

```

15
16 create or replace table lineitem_raw_json (src variant);
17
18

```

Results Data Preview

✓ Query_ID SQL 195ms 1 rows

Filter result...



Copy

Row	status
1	Table LINEITEM_RAW_JSON successfully created.

```

18 copy into lineitem_raw_json from @stg_lineitem_json_dev ON_ERROR = ABORT_STATEMENT;
19 |
20 |
21 |
22 |
23 |
24

```

Results Data Preview

[Open History](#)

✓ Query_ID SQL 6.29s 128 rows

Filter result...



Copy

Columns ▾

Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_charac	first_error_column
1	s3://sid-snow...	LOADED	7936	7936	1	0	NULL	NULL	NULL	NULL
2	s3://sid-snow...	LOADED	7936	7936	1	0	NULL	NULL	NULL	NULL
3	s3://sid-snow...	LOADED	7936	7936	1	0	NULL	NULL	NULL	NULL

```

21
22 select
23   SRC:L_ORDERKEY,
24   SRC:L_PARTKEY,
25   SRC:L_SUPPKEY,
26   SRC:L_LINENUMBER,
27   SRC:L_QUANTITY,
28   SRC:L_EXTENDEDPRICE,
29   SRC:L_DISCOUNT,
30   SRC:L_TAX,
31   SRC:L_RETURNFLAG,
32   SRC:L_LINESTATUS,
33   SRC:L_SHIPDATE,
34   SRC:L_COMMITDATE,
35   SRC:L_RECEIPTDATE,
36   SRC:L_SHIPINSTRUCT,
37   SRC:L_SHIPMODE,
38   SRC:L_COMMENT

```

Results Data Preview

[Open History](#)

✓ Query_ID SQL 333ms 10 rows

Filter result...



Copy

Columns ▾

Row	SRC:L_ORDERKEY	SRC:L_PARTKEY	SRC:L_SUPPKEY	SRC:L_LINENUM	SRC:L_QUANTITY	SRC:L_EXTENDEDPRICE	SRC:L_DISCOUNT	SRC:L_TAX	SRC:L_RETURNFLAG	SRC:L_LINESTATUS	SRC:L_SHIPDATE
1	2844948486	72018843	9518865	1	33	58021.92	0.05	0.05	"N"	"O"	"1998-07"
2	5358629669	104881094	7381105	2	38	40654.3	0.08	0	"N"	"O"	"1998-07"
3	1547249569	187766210	266229	1	26	32937.58	0.08	0.06	"N"	"O"	"1998-07"

```

21
22 insert into lineitem
23 select
24   SRC:L_ORDERKEY,
25   SRC:L_PARTKEY,
26   SRC:L_SUPKEY,
27   SRC:L_LINENUMBER,
28   SRC:L_QUANTITY,
29   SRC:L_EXTENDEDPRICE,
30   SRC:L_DISCOUNT,
31   SRC:L_TAX,
32   SRC:L_RETURNFLAG,
33   SRC:L_LINESTATUS,
34   SRC:L_SHIPDATE,
35   SRC:L_COMMITDATE,
36   SRC:L_RECEIPTDATE,
37   SRC:L_SHIPINSTRUCT,
38   SRC:L_SHIPMODE,
39   SRC:L_COMMENT

```

Results Data Preview Open History

✓ Query ID SQL 556ms 1 rows

Filter result... Copy

Columns x

Row	number of rows inserted
1	10

Continuous data ingestion:

Amazon S3 > sid-snowflake-data > ecommerce_dev/ > lineitem/ > lineitem_snowpipe/ > csv/

csv/ Copy S3 URI

Objects Properties

Objects (0)
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

< 1 > ⚙️

Name	Type	Last modified	Size	Storage class
No objects You don't have any objects in this folder.				

Upload

```

use role sysadmin;

use schema ecommerce_db.ecommerce_liv;

-- Create a stage for lineitem table ---
create stage stg_lineitem_csv_dev
storage_integration = aws_sf_data
url = '{your_bucket_name}/ecommerce_dev/lineitem/lineitem_csv/'
file_format = csv_load_format;

```

```

list @stg_lineitem_csv_dev;

create or replace pipe lineitem_pipe auto_ingest=true as
copy into lineitem from @stg_lineitem_csv_dev ON_ERROR = continue;

show pipes;

select * from lineitem limit 10;

select * from information_schema.load_history where table_name='LINEITEM' order by
last_load_time desc limit 10;

use role accountadmin;

-- Switch role to accountadmin before running this command --
select *
from table(information_schema.pipe_usage_history(
    date_range_start=>dateadd('hour',-3,current_timestamp()),
    pipe_name=>'lineitem_pipe'));

```

The screenshot shows the Snowflake UI interface. At the top, there is a code editor window containing the SQL commands for creating a stage area, a pipe, and performing a copy operation. Below the code editor is a results panel. The results panel has tabs for 'Results' and 'Data Preview', with 'Results' being active. It displays a single row of data indicating the successful creation of the stage area. The results table has two columns: 'Row' and 'status'. The first row shows the message 'Stage area STG_LINEITEM_CSV_DEV successfully created.'.

Row	status
1	Stage area STG_LINEITEM_CSV_DEV successfully created.

```

12 create or replace pipe lineitem_pipe auto_ingest=true as
13 copy into lineitem from @stg_lineitem_csv_dev ON_ERROR = continue;
14

```

Results Data Preview

✓ [Query ID](#) [SQL](#) 608ms 1 rows

Filter result... [Download](#) [Copy](#)

Row	status
1	Pipe LINEITEM_PIPE successfully created.

```

14
15 show pipes;
16

```

Results Data Preview [Open History](#)

✓ [Query ID](#) [SQL](#) 50ms 1 rows

Filter result... [Download](#) [Copy](#) Columns ▾

Row	created_on	name	database_name	schema_name	definition	owner	notification_channel	comment
1	2022-02-12 ...	LINEITEM_PIPE	ECOMMERCE	ECOMMERCE	copy into lin...	SYSADMIN	arn:aws:sqs:eu-central-1:206954943453:sf-snowpipe-AIDATA...	

Then enable event notification in s3 bucket(enable in bucket not in directory)

Event notifications (0)
Send a notification when specific events occur in your bucket. [Learn more](#)

Name	Event types	Filters	Destination type	Destination
No event notifications Choose Create event notification to be notified when a specific event occurs. Create event notification				

Amazon EventBridge
For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. [Learn more](#) or see [EventBridge pricing](#)

Send notifications to Amazon EventBridge for all events in this bucket	Edit
Off	

Create event notification Info

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

General configuration

Event name

Event name can contain up to 255 characters.

Prefix - optional

Limit the notifications to objects with key starting with specified characters.

Suffix - optional

Limit the notifications to objects with key ending with specified characters.

Event types

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

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

206954943453:sf-snowpipe-AIDATAL3Z2POYZR3RZM5K-A3ejjCy4g30L51DpmUDnJA

Now, upload some files:

Amazon S3 > sid-snowflake-data > ecommerce_dev/ > lineitem/ > lineitem_snowpipe/ > csv/

csv/

[Copy S3 URI](#)

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	data_0_4_0.csv.gz	gz	February 12, 2022, 21:30:17 (UTC+04:00)	30.6 KB	Standard
<input type="checkbox"/>	data_0_5_0.csv.gz	gz	February 12, 2022, 21:30:22 (UTC+04:00)	779.5 KB	Standard

```
20
21 select * from information_schema.load_history where table_name='LINEITEM' order by last_load_time desc limit 10;
22
```

Results Data Preview [Open History](#)

✓ [Query_ID](#) [SQL](#) 825ms 10 rows

Filter result... [Copy](#)

Row	SCHEMA_NAME	FILE_NAME	TABLE_NAME	LAST_LOAD_TIM	STATUS	R
1	ECOMMERC...	s3://sid-snowflake-data/ecommerce_dev/lineitem/lineitem_snowpipe/csv/data_0_4_0.csv.gz	LINEITEM	2022-02-12 ...	LOADED	
2	ECOMMERC...	s3://sid-snowflake-data/ecommerce_dev/lineitem/lineitem_csv/data_15_3_0.csv.gz	LINEITEM	2022-02-12 ...	LOADED	
3	ECOMMERC...	s3://sid-snowflake-data/ecommerce_dev/lineitem/lineitem_csv/data_2_0_0.csv.gz	LINEITEM	2022-02-12 ...	LOADED	

Unload/Extract data into S3:

```

use schema "ECOMMERCE_DB"."ECOMMERCE_LIV";

-- Extract/Unload data ---
copy into s3://sid-snowflake-data/unloaded_data/lineitem/
from
(
  select * from "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM" limit 100000
)
storage_integration=aws_sf_data
single=false
file_format = csv_load_format;

-- Extract/Unload data using partition by ---
copy into s3://sid-snowflake-data/unloaded_data/lineitem_partitioned/
from
(
  select * from "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM" limit 100000
)
partition by L_SHIPDATE
storage_integration=aws_sf_data
single=false
file_format = csv_load_format;

copy into s3://sid-snowflake-data/unloaded_data/lineitem_parquet/
from
(
  select * from "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM"
  limit 1000000
)
storage_integration=aws_sf_data
single=false
file_format = parquet_format;

create or replace file format json_unload_format
type='json'

copy into s3://sid-snowflake-data/ecommerce_dev/lineitem/lineitem_json/
from
(
  select
  object_construct(
  'L_ORDERKEY',L_ORDERKEY,
  'L_PARTKEY',L_PARTKEY,
  'L_SUPPKEY',L_SUPPKEY,
  'L_LINENUMBER',L_LINENUMBER,
  'L_QUANTITY',L_QUANTITY,
  'L_EXTENDEDPRICE',L_EXTENDEDPRICE,
  'L_DISCOUNT',L_DISCOUNT,

```

```

'L_TAX',L_TAX,
'L_RETURNFLAG',L_RETURNFLAG,
'L_LINESTATUS',L_LINESTATUS,
'L_SHIPDATE',L_SHIPDATE,
'L_COMMITDATE',L_COMMITDATE,
'L_RECEIPTDATE',L_RECEIPTDATE,
'L_SHIPINSTRUCT',L_SHIPINSTRUCT,
'L_SHIPMODE',L_SHIPMODE,
'L_COMMENT',L_COMMENT
)
from "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM"
limit 1000000
)
storage_integration=aws_sf_data
single=false
file_format = json_unload_format;

```

4
5 copy into s3://sid-snowflake-data/unloaded_data/lineitem/
6 from
7 (
8 select * from "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM" limit 100000
9)
10 storage_integration=aws_sf_data
11 single=false
12 file_format = csv_load_format;



Row	rows_unloaded	input_bytes	output_bytes
1	100000	15364609	3960605

Unload by partition key:

13
14 copy into s3://sid-snowflake-data/unloaded_data/lineitem_partitioned/
15 from
16 (
17 select * from "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM" limit 100000
18)
19 partition by L_SHIPDATE
20 storage_integration=aws_sf_data
21 single=false
22 file_format = csv_load_format;|



Row	rows_unloaded	input_bytes	output_bytes
1	100000	15363678	3980757

Amazon S3 > sid-snowflake-data > unloaded_data/ > lineitem_partitioned/

lineitem_partitioned/

Objects Properties

Objects (54)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your more

Copy S3 URI Copy URL Download Open Delete Actions Create

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	<input type="checkbox"/>	Size
<input type="checkbox"/>	<input type="checkbox"/> 1996-06-01/	Folder	-		
<input type="checkbox"/>	<input type="checkbox"/> 1996-06-02/	Folder	-		
<input type="checkbox"/>	<input type="checkbox"/> 1996-06-03/	Folder	-		
<input type="checkbox"/>	<input type="checkbox"/> 1996-06-04/	Folder	-		
<input type="checkbox"/>	<input type="checkbox"/> 1996-06-05/	Folder	-		

Unload into json format:

```
34 'L_PARTKEY',L_PARTKEY,  
35 'L_SUPPKEY',L_SUPPKEY,  
36 'L_LINENUMBER',L_LINENUMBER,  
37 'L_QUANTITY',L_QUANTITY,  
38 'L_EXTENDEDPRICE',L_EXTENDEDPRICE,  
39 'L_DISCOUNT',L_DISCOUNT,  
40 'L_TAX',L_TAX,  
41 'L_RETURNFLAG',L_RETURNFLAG,  
42 'L_LINESTATUS',L_LINESTATUS,  
43 'L_SHIPDATE',L_SHIPDATE,  
44 'L_COMMITDATE',L_COMMITDATE,  
45 'L_RECEIPTDATE',L_RECEIPTDATE,  
46 'L_SHIPINSTRUCT',L_SHIPINSTRUCT,  
47 'L_SHIPMODE',L_SHIPMODE,  
48 'L_COMMENT',L_COMMENT  
49 )  
50 from "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM"  
51 limit 1000000  
52 )  
53 storage_integration=aws_sf_data  
54 single=false  
55 file_format = json_unload_format;
```

Results Data Preview Open History

✓ Query ID SQL 1.1s 1 rows Copy Filter result... Columns

Row	rows_unloaded	input_bytes	output_bytes
1	1000000	371560417	48962954

Tasks:

```
use role accountadmin;
```

```

create role task_owner;

grant usage on warehouse compute_wh to role task_owner;

grant usage on database ECOMMERCE_DB to role task_owner;

grant usage on schema ECOMMERCE_DB.ECOMMERCE_DEV to role task_owner;

grant select on "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM" to role task_owner;
grant select on "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" to role task_owner;

grant create task on schema ECOMMERCE_DB.ECOMMERCE_DEV to role task_owner;
grant execute task on account to role task_owner;
GRANT OWNERSHIP ON all tasks in schema "ECOMMERCE_DB"."ECOMMERCE_DEV"
TO ROLE task_owner;
grant create table on schema ECOMMERCE_DB.ECOMMERCE_DEV to role task_owner;

grant role task_owner to user mohamedzuhairka;

use role task_owner;

```

```

3 create role task_owner;
4
5 grant usage on warehouse prod_xl to role task_owner;
6 grant usage on database ECOMMERCE_DB to role task_owner;
7 grant usage on schema ECOMMERCE_DB.ECOMMERCE_LIV to role task_owner;
8 grant select on "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM" to role task_owner;
9 grant select on "ECOMMERCE_DB"."ECOMMERCE_LIV"."ORDERS" to role task_owner;
10
11 grant create task on schema ECOMMERCE_DB.ECOMMERCE_LIV to role task_owner;
12 grant execute task on account to role task_owner;
13 GRANT OWNERSHIP ON all tasks in schema "ECOMMERCE_DB"."ECOMMERCE_LIV" TO ROLE task_owner;
14 grant create table on schema ECOMMERCE_DB.ECOMMERCE_LIV to role task_owner;
15
16 grant role task_owner to user siddharth;
17
18 use role task_owner;

```

Results Data Preview

✓ Query ID SQL 38ms 1 rows

Filter result...

Row	status
1	Statement executed successfully.

```

9 grant select on "ECOMMERCE_DB"."ECOMMERCE_LIV"."ORDERS" to role task_owner;
10
11 grant create task on schema ECOMMERCE_DB.ECOMMERCE_LIV to role task_owner;
12 grant execute task on account to role task_owner;
13 GRANT OWNERSHIP ON all tasks in schema "ECOMMERCE_DB"."ECOMMERCE_LIV" TO ROLE task_owner;
14 grant create table on schema ECOMMERCE_DB.ECOMMERCE_LIV to role task_owner;
15
16 grant role task_owner to user siddharth;
17
18 use role task_owner;

```

Results Data Preview

Query_ID SQL 39ms 1 rows

Filter result...

Row status

1 Statement executed successfully.

```

use ecommerce_db.ecommerce_dev;

--- Create a table to store the results ---
create or replace TABLE DAILY_AGGREGATED_SUMMARY (
    SUM_QTY NUMBER(24,2),
    TOTAL_BASE_PRICE NUMBER(24,2),
    TOTAL_DISCOUNT_PRICE NUMBER(37,4),
    TOTAL_CHARGE NUMBER(38,6),
    ORDER_COUNT NUMBER(18,0),
    SHIPPED_DATE DATE,
    SHIPPED_MODE VARCHAR(10)
);

create or replace TABLE ORDERS_BY_SHIPMODE (
    TOTAL_ORDERS NUMBER(30,0),
    TOTAL_DISCOUNT NUMBER(38,0),
    SHIPPED_DATE DATE,
    SHIPPED_MODE VARCHAR(10)
);

-- Standalone Task -----
create task TSK_DAILY_SALES_SUMMARY
warehouse = prod_xl
schedule = 'using cron 0 8 * * * UTC' as
insert into "ECOMMERCE_DB"."ECOMMERCE_DEV"."DAILY_AGGREGATED_SUMMARY"
select
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as total_base_price,
    sum(l_extendedprice * (1-l_discount)) as total_discount_price,
    sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as total_charge,
    count(*) as order_count,

```

```

date(l_shipdate) as shipped_date,
l_shipmode as shipped_mode
from
    "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM"
where
    shipped_date = '1996-07-17'
group by
    shipped_date,
    shipped_mode;

show tasks;

alter task DAILY_AGGREGATED_SUMMARY suspend;

-- Dependent Task ----

create task TSK_ORDERS_BY_SHIPMODE
warehouse = prod_xl
AFTER TSK_DAILY_SALES_SUMMARY
insert into "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS_BY_SHIPMODE"
select
    round(sum(order_count)) as total_orders ,
    round(sum(total_discount_price),0) as total_discount,
    shipped_date,
    shipped_mode
from
    daily_aggregated_summary
group by 3,4

alter task TSK_ORDERS_BY_SHIPMODE resume ;
alter task DAILY_AGGREGATED_SUMMARY resume ;

---- Check Task History -----
use role accountadmin;

select *
    from table(information_schema.task_history(
        scheduled_time_range_start=>dateadd('hour',-1,current_timestamp()),
        result_limit => 10,
        task_name=>'task_name'));

```

```
22 create or replace TABLE DAILY_AGGREGATED_SUMMARY (
23     SUM_QTY NUMBER(24,2),
24     TOTAL_BASE_PRICE NUMBER(24,2),
25     TOTAL_DISCOUNT_PRICE NUMBER(37,4),
26     TOTAL_CHARGE NUMBER(38,6),
27     ORDER_COUNT NUMBER(18,0),
28     SHIPPED_DATE DATE,
29     SHIPPED_MODE VARCHAR(10)
30 );
31
32 create or replace TABLE ORDERS_BY_SHIPMODE (
33     TOTAL_ORDERS NUMBER(30,0),
34     TOTAL_DISCOUNT NUMBER(38,0),
35     SHIPPED_DATE DATE,
36     SHIPPED_MODE VARCHAR(10)
37 );
38
```

Results Data Preview

✓ Query ID SQL 148ms 1 rows

Filter result...



Copy

Row	status
1	Table DAILY_AGGREGATED_SUMMARY successfully created.

```
31
32 create or replace TABLE ORDERS_BY_SHIPMODE (
33     TOTAL_ORDERS NUMBER(30,0),
34     TOTAL_DISCOUNT NUMBER(38,0),
35     SHIPPED_DATE DATE,
36     SHIPPED_MODE VARCHAR(10)
37 );
38
```

Results Data Preview

✓ Query ID SQL 172ms 1 rows

Filter result...



Copy

Row	status
1	Table ORDERS_BY_SHIPMODE successfully created.

```

39
40 create task TSK_DAILY_SALES_SUMMARY
41 warehouse = prod_xl
42 schedule = 'using cron 0 8 * * * UTC' as
43 insert into "ECOMMERCE_DB"."ECOMMERCE_LIV"."DAILY_AGGREGATED_SUMMARY"
44 select
45     sum(l_quantity) as sum_qty,
46     sum(l_extendedprice) as total_base_price,
47     sum(l_extendedprice * (1-l_discount)) as total_discount_price,
48     sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as total_charge,
49     count(*) as order_count,
50     date(l_shipdate) as shipped_date,
51     l_shipmode as shipped_mode
52 from
53     "ECOMMERCE_DB"."ECOMMERCE_LIV"."LINEITEM"
54 where
55     shipped_date = '1996-07-17'
56 group by
57     shipped_date,
58     shipped_mode ;
59

```

Results Data Preview

✓ Query ID SQL 90ms 1 rows

Filter result...



Copy

Row	status
1	Task TSK_DAILY_SALES_SUMMARY successfully created.

60 show tasks;|

Results Data Preview

Open History

✓ Query ID SQL 62ms 1 rows

Filter result...



Copy

Columns ▾

Row	created_on	name	id	database_name	schema_name	owner	comment	warehouse	schedule	predecessors	state
1	2022-03-19 ...	TSK_DAILY_...	01a30849-2...	ECOMMERCE...	ECOMMERCE...	TASK_OWNER		PROD_XL	using cron 0...	NULL	suspended

62 alter task TSK_DAILY_SALES_SUMMARY resume;

Results Data Preview

✓ Query ID SQL 119ms 1 rows

Filter result...



Copy

Row	status
1	Statement executed successfully.

To create child task, the parent or root task should be in suspend state.

```

64
65 create task TSK_ORDERS_BY_SHIPMODE
66 warehouse = prod_xl
67 AFTER TSK_DAILY_SALES_SUMMARY as
68 insert into "ECOMMERCE_DB"."ECOMMERCE_LIV"."ORDERS_BY_SHIPMODE"
69 select
70     round(sum(order_count)) as total_orders ,
71     round(sum(total_discount_price),0) as total_discount,
72     shipped_date,
73     shipped_mode
74 from
75     daily_aggregated_summary
76 group by 3,4;

```

Results Data Preview

✗ Query ID SQL 85ms

Unable to update graph with root task ECOMMERCE_DB.ECOMMERCE_LIV.TSK_DAILY_SALES_SUMMARY since that root task is not suspended.

After parent task is suspended

```

64
65 create task TSK_ORDERS_BY_SHIPMODE
66 warehouse = prod_xl
67 AFTER TSK_DAILY_SALES_SUMMARY as
68 insert into "ECOMMERCE_DB"."ECOMMERCE_LIV"."ORDERS_BY_SHIPMODE"
69 select
70     round(sum(order_count)) as total_orders ,
71     round(sum(total_discount_price),0) as total_discount,
72     shipped_date,
73     shipped_mode
74 from
75     daily_aggregated_summary
76 group by 3,4;
77

```

Results Data Preview

✓ Query ID SQL 78ms 1 rows

Filter result...



Copy

Row	status
1	Task TSK_ORDERS_BY_SHIPMODE successfully created.

name	id	database_name	schema_name	owner	comment	warehouse	schedule	predecessors	state
TSK_DAILY_SALES_SUMMARY	01a30849-2...	ECOMMERCE_DB	ECOMMERCE_LIV	TASK_OWNER		PROD_XL	using cron 0...	NULL	suspended
TSK_ORDERS_BY_SHIPMODE	01a3084b-2...	ECOMMERCE_DB	ECOMMERCE_LIV	TASK_OWNER		PROD_XL	NULL	ECOMMERCE_DB.ECOMMERCE_LIV.TSK_DAILY_SALES_SUMMARY	suspended

To resume the tasks, first resume child task and then parent task. That is the order.

Streams:

```
use role sysadmin;
use database ecommerce_db;

create or replace schema streams_test;

--- Create a raw table to test the streams ---
CREATE OR REPLACE TABLE members_raw (
    id number(8) NOT NULL,
    name varchar(255) default NULL,
    fee number(3) NULL
);

--- Create a production table which will consume the streams data ---
CREATE OR REPLACE TABLE members_prod (
    id number(8) NOT NULL,
    name varchar(255) default NULL,
    fee number(3) NULL
);

--- Create a standard (delta) stream on the raw table :members_raw---
CREATE OR REPLACE STREAM members_std_stream ON TABLE members_raw;

--- Check the streams ---
select * from members_std_stream

--- Check the stream offset ---
SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream') as
members_table_st_offset;

SELECT
to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as
members_table_st_offset;

--- Insert some data into the raw table : members_raw ---

INSERT INTO members_raw (id,name,fee)
VALUES
(1,'Joe',0),
(2,'Jane',0),
(3,'George',0),
(4,'Betty',0),
(5,'Sally',0)

--- Check the streams ---
```

```
select * from members_std_stream

--- Check the stream offset ----
SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream') as
members_table_st_offset;

--- Query the streams data by ingesting the CDC streams data into the production table ----
SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION =
'INSERT';

--- Consume the streams data by ingesting the CDC streams data into the production table :
DML Operation ----
INSERT INTO members_prod(id,name,fee)
SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION =
'INSERT';

--- Check the production table -----
select * from members_prod;

--- Check the offset ----
SELECT
to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as
members_table_st_offset;

--- Insert some more data into the raw table ---
INSERT INTO members_raw (id,name,fee) VALUES (6,'sid',0),(7,'Billy',0),(8,'Katie',0);

--- Check the streams -----
select * from members_std_stream;

--- Update the raw table -----
update members_raw set fee=10 where id=7;

--- Check the streams -----
select * from members_std_stream;

--- Consume the streams ---
INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM
members_std_stream WHERE METADATA$ACTION = 'INSERT';

--- Lets update the raw table again ----
update members_raw set fee=20 where id=7;
```

```

--- Consume both (inserts and updates) from the streams ---
merge into members_prod as mp
using (select id,name,fee from members_std_stream mstr where metadata$action='INSERT' )
as mstr
on mp.id = mstr.id
when matched then update set mp.fee = mstr.fee,mp.name = mstr.name
when not matched then insert (id,name,fee) values (mstr.id,mstr.name,cast(mstr.fee as
numeric));

```

```

7 --- Create a raw table to test the streams ---
8 CREATE OR REPLACE TABLE members_raw (
9   id number(8) NOT NULL,
10  name varchar(255) default NULL,
11  fee number(3) NULL
12 );
13
14 --- Create a production table which will consume the streams data ---
15 CREATE OR REPLACE TABLE members_prod (
16   id number(8) NOT NULL,
17  name varchar(255) default NULL,
18  fee number(3) NULL
19 );
20
21 --- Create a standard (delta) stream on the raw table :members_raw---

```

Results Data Preview

✓ [Query ID](#) [SQL](#) 269ms  1 rows

Filter result...



[Copy](#)

Row	status
1	Table MEMBERS_RAW successfully created.

```

13
14 --- Create a production table which will consume the streams data ---
15 CREATE OR REPLACE TABLE members_prod (
16   id number(8) NOT NULL,
17  name varchar(255) default NULL,
18  fee number(3) NULL
19 );
20
21 --- Create a standard (delta) stream on the raw table :members_raw---

```

Results Data Preview

✓ [Query ID](#) [SQL](#) 185ms  1 rows

Filter result...



[Copy](#)

Row	status
1	Table MEMBERS_PROD successfully created.

```

21 --- Create a standard (delta) stream on the raw table :members_raw---
22 CREATE OR REPLACE STREAM members_std_stream ON TABLE members_raw;
23
24 --- Check the streams ----

```

Results Data Preview

✓ Query ID SQL 186ms 1 rows

Filter result...

Row	status
1	Stream MEMBERS_STD_STREAM successfully created.

```

23
24 --- Check the streams ----
25 select * from members_std_stream;
26

```

Results Data Preview

✓ Query ID SQL 117ms 0 rows

Filter result...

Columns ▾

Row	ID	NAME	FEE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
-----	----	------	-----	------------------	--------------------	------------------

```

27 --- Check the stream offset ----
28 SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream') as members_table_st_offset;
29
30 SELECT to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as members_table_st_offset;
31

```

Results Data Preview

✓ Query ID SQL 238ms 1 rows

Filter result...

Row	MEMBERS_TABLE_ST_OFFSET
-----	-------------------------

1 0

```

33
34 INSERT INTO members_raw (id, name, fee)
35 VALUES
36 (1, 'Joe', 0),
37 (2, 'Jane', 0),
38 (3, 'George', 0),
39 (4, 'Betty', 0),
40 (5, 'Sally', 0);
41
42
43 select * from members_std_stream;

```

Results Data Preview

✓ Query ID SQL 1.43s 1 rows

Filter result...

Row

1

```

41
42 select * from members_std_stream;
43

```

Results Data Preview

✓ Query ID SQL 501ms 5 rows

Row	ID	NAME	FEE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	1	Joe	0	INSERT	FALSE	4f35e8ddd6b00de924513...
2	2	Jane	0	INSERT	FALSE	26940c421f3574c0c2950...
3	3	George	0	INSERT	FALSE	99670769bf83a71134bb9...
4	4	Betty	0	INSERT	FALSE	74db03241256e49e2fbdb...
5	5	Sally	0	INSERT	FALSE	a03ad26860a644a36828...

```

48 INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT';
49
50 select * from members_prod;

```

Results Data Preview

✓ Query ID SQL 541ms 1 rows

Row
1

```

49
50 select * from members_prod;
51
52 select * from members_std_stream;
53

```

Results Data Preview

✓ Query ID SQL 172ms 5 rows

Row	ID	NAME	FEE
1	1	Joe	0
2	2	Jane	0
3	3	George	0
4	4	Betty	0
5	5	Sally	0

Once the data has been consumed, it will be removed from streams:

```

51
52 select * from members_std_stream;
53

```

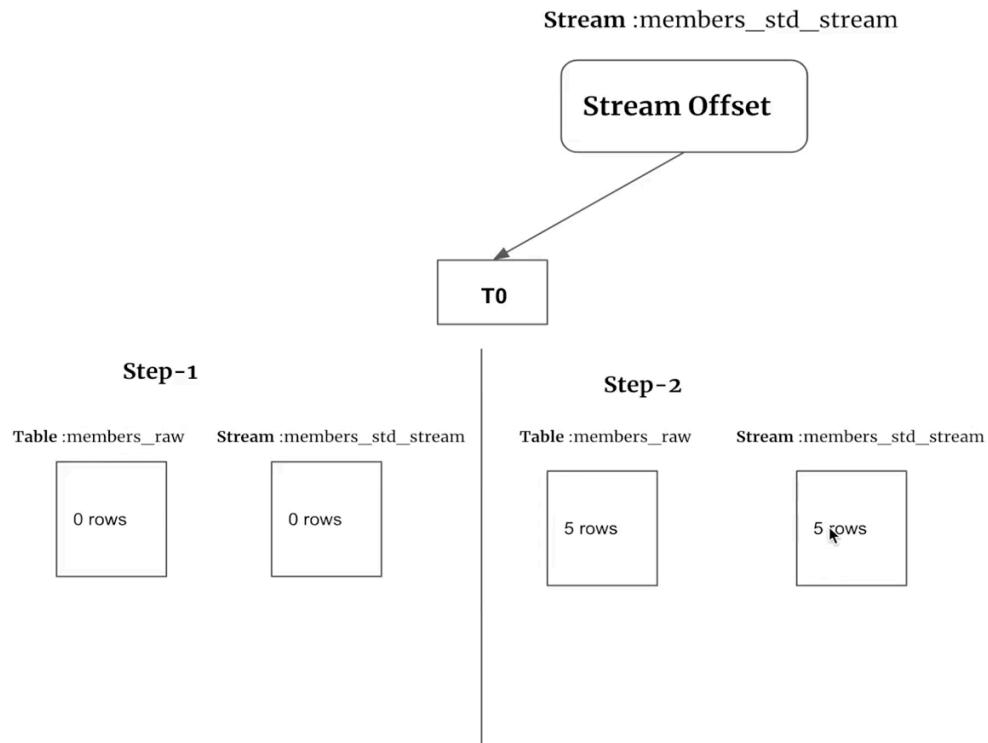
Results Data Preview

✓ Query ID SQL 451ms 0 rows

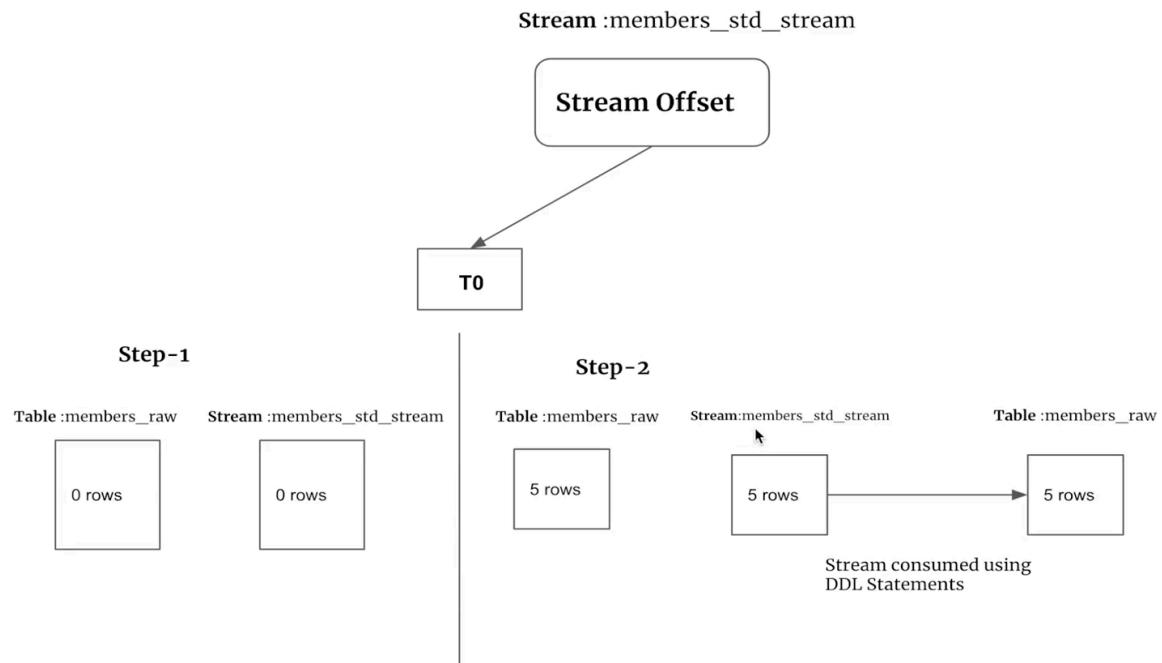
Row	ID	NAME	FEE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
-----	----	------	-----	------------------	--------------------	------------------

Behind the action:

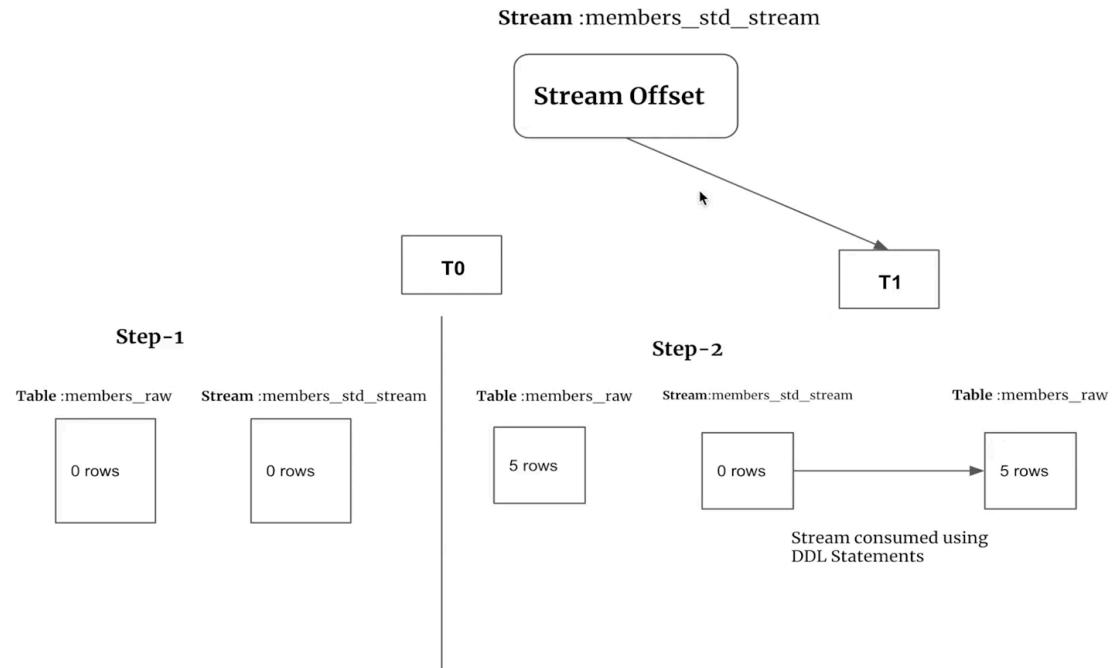
Snowflake - Stream Types



Snowflake - Stream Types



Snowflake - Stream Types



T0:

```
27 --- Check the stream offset ----
28 SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream') as members_table_st_offset;
29
30 SELECT to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as members_table_st_offset;
31
```

Results Data Preview

Row	MEMBERS_TABLE_ST_OFFSET
1	0

T1:

```
54 SELECT to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as members_table_st_offset;
55
56
```

Results Data Preview

Row	MEMBERS_TABLE_ST_OFFSET
1	2022-03-03 17:48:44.725

Consume model in stream:

Here, we have added three more rows in raw table and update one row. But, when we select the stream, for ID 7, it is showing action as **INSERT** instead of **UPDATE**. This is because we haven't consumed the **INSERT** action from the stream as of now.

```

56
57 INSERT INTO members_raw (id,name,fee)
58 VALUES
59 (6,'sid',0),
60 (7,'Billy',0),
61 (8,'Katie',0);
62
63 select * from members_std_stream;
64
65 update members_raw set fee=10 where id=7;
66
67 select * from members_std_stream;
68
69 INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT';
70

```

Results Data Preview Open History

Row	ID	NAME	Fee	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	6	sid	0	INSERT	FALSE	294663dcd204d2c10244...
2	7	Billy	10	INSERT	FALSE	69d5e4092b39df97fb24b...
3	8	Katie	0	INSERT	FALSE	ad9e58bf74e26883a8fff3...

After we consume the stream records and reupdate the same ID. Then, when we check the stream, we can see the old and new value of same record.

```

68
69 INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT';
70

```

Results Data Preview Open History

Row
1


```

73 update members_raw set fee=20 where id=7;
74
75 select * from members_std_stream;
76
77 //and metadata$isupdate='TRUE'

```

Results Data Preview Open History

Row	ID	NAME	Fee	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	7	Billy	20	INSERT	TRUE	69d5e4092b39df97fb24b...
2	7	Billy	10	DELETE	TRUE	69d5e4092b39df97fb24b...

```

78
79 merge into members_prod as mp
80 using
81 (
82   select id,name,fee from members_std_stream mstr where metadata$action='INSERT'
83 ) as mstr
84 on mp.id = mstr.id
85 when matched then update set mp.fee = mstr.fee,mp.name = mstr.name
86 when not matched then insert (id,name,fee) values (mstr.id,mstr.name,cast(mstr.fee as numeric));
87
88

```

Results Data Preview [Open History](#)

✓ Query_ID SQL 810ms 1 rows

Filter result... [Download](#) [Copy](#) Columns ▾

Row	number of rows inserted	number of rows updated
1	0	1

Append only stream:

```

use role sysadmin;
use database ecommerce_db;

create or replace schema streams_test;

--- Create a raw table to test the streams ---
CREATE OR REPLACE TABLE members_raw (
  id number(8) NOT NULL,
  name varchar(255) default NULL,
  fee number(3) NULL
);

--- Create a production table which will consume the streams data ---
CREATE OR REPLACE TABLE members_prod (
  id number(8) NOT NULL,
  name varchar(255) default NULL,
  fee number(3) NULL
);

--- Create a append-only stream on the raw table :members_raw---
CREATE OR REPLACE STREAM members_append_stream ON TABLE members_raw
append_only=true;

--- Check the streams ---
select * from members_append_stream

--- Check the stream offset ---
SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_append_stream') as
members_table_st_offset;

SELECT
to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_append_stream'))
as members_table_st_offset;

```

```
--- Insert some data into the raw table : members_raw ---
```

```
INSERT INTO members_raw (id,name,fee)
VALUES
(1,'Joe',0),
(2,'Jane',0),
(3,'George',0),
(4,'Betty',0),
(5,'Sally',0)
```

```
--- Check the streams ----
```

```
select * from members_append_stream
```

```
--- Check the stream offset ----
```

```
SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream') as
members_table_st_offset;
```

```
--- Query the streams data by ingesting the CDC streams data into the production table ----
```

```
SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION =
'INSERT';
```

```
--- Consume the streams data by ingesting the CDC streams data into the production table :
DML Operation ----
```

```
INSERT INTO members_prod(id,name,fee)
SELECT id, name, fee FROM members_std_stream;
```

```
--- Check the production table -----
```

```
select * from members_prod;
```

```
--- Check the offset ----
```

```
SELECT
to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as
members_table_st_offset;
```

```
--- Insert some more data into the raw table ---
```

```
INSERT INTO members_raw (id,name,fee) VALUES (6,'sid',0),(7,'Billy',0),(8,'Katie',0);
```

```
--- Check the streams -----
```

```
select * from members_std_stream;
```

```
--- Update the raw table -----
```

```
update members_raw set fee=10 where id=7;
```

```
--- Check the streams -----
```

```
select * from members_std_stream;
```

```

--- Consume the streams ---
INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM
members_std_stream WHERE METADATA$ACTION = 'INSERT';

--- Lets update the raw table again ----
update members_raw set fee=20 where id=7;

--- Consume both (inserts and updates) from the streams ---
merge into members_prod as mp
using (select id,name,fee from members_std_stream mstr where metadata$action='INSERT' )
as mstr
on mp.id = mstr.id
when matched then update set mp.fee = mstr.fee,mp.name = mstr.name
when not matched then insert (id,name,fee) values (mstr.id,mstr.name,cast(mstr.fee as
numeric));

```

20
21 CREATE OR REPLACE STREAM members_append_stream ON TABLE members_raw append_only=true;
22
23 --- Check the streams ----
24 select * from members_append_stream;|
25
26 --- Check the stream offset ----
27 SELECT SYSTEM\$STREAM_GET_TABLE_TIMESTAMP('members_append_stream') as members_table_st_offset;
28
29 SELECT to_timestamp(SYSTEM\$STREAM_GET_TABLE_TIMESTAMP('members_append_stream')) as members_table_st_offset;

Results Data Preview

✓ Query ID SQL 168ms 1 rows

Filter result...

Row	status
1	Stream MEMBERS_APPEND_STREAM successfully created.

26 --- Check the stream offset ----
27 SELECT SYSTEM\$STREAM_GET_TABLE_TIMESTAMP('members_append_stream') as members_table_st_offset;
28
29 SELECT to_timestamp(SYSTEM\$STREAM_GET_TABLE_TIMESTAMP('members_append_stream')) as members_table_st_offset;
30
31 --- Insert some data into the raw table : members_raw ---
32

Results Data Preview

✓ Query ID SQL 60ms 1 rows

Filter result...

Row	MEMBERS_TABLE_ST_OFFSET
1	0

```

32
33 INSERT INTO members_raw (id,name,fee)
34 VALUES
35 (1,'Joe',0),
36 (2,'Jane',0),
37 (3,'George',0),
38 (4,'Betty',0),
39 (5,'Sally',0);
40

```

Results Data Preview

✓ Query_ID SQL 478ms 1 rows

Filter result...



Copy

Row

1

```

40
41 select * from members_append_stream;
42
43 SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_append_stream') as members_table_st_offset;
44
45 SELECT id, name, fee FROM members_append_stream WHERE METADATA$ACTION = 'INSERT';
46

```

Results Data Preview

[Open History](#)

✓ Query_ID SQL 266ms 5 rows

Filter result...



Copy

Columns ▾

Row	ID	NAME	FEE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	1	Joe	0	INSERT	FALSE	42e5f5b691fce60baab7cc...
2	2	Jane	0	INSERT	FALSE	1bb7007d37712f8950328...
3	3	George	0	INSERT	FALSE	a903861a242796afb6f440...
4	4	Betty	0	INSERT	FALSE	e06ab53f01186dbe9b493...
5	5	Sally	0	INSERT	FALSE	6620a023cf63ebea29ad1...

```

46
47 INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM members_append_stream ;
48
49 select * from members_prod;
50

```

Results Data Preview

✓ Query_ID SQL 576ms 1 rows

Filter result...



Copy

Row

1

```

48
49 select * from members_prod;
50
51 select * from members_append_stream;
52
53 SELECT to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_append_stream')) as members_table_st_offset;
54

```

Results Data Preview

✓ Query ID SQL 202ms 5 rows

Filter result...

Copy

Row	ID	NAME
1	1	Joe
2	2	Jane
3	3	George
4	4	Betty
5	5	Sally

```

51 select * from members_append_stream;
52
53 SELECT to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_append_stream')) as members_table_st_offset;
54

```

Results Data Preview

✓ Query ID SQL 204ms 0 rows

Filter result...

Copy

Columns

Row	ID	NAME	FEE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1						

```

53 SELECT to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_append_stream')) as members_table_st_offset;
54
55
56 update members_raw set fee=10 where id=4;

```

Results Data Preview

✓ Query ID SQL 48ms 1 rows

Filter result...

Copy

Columns

Row	MEMBERS_TABLE_ST_OFFSET
1	2022-03-04 07:58:05.439

In append-only, streams only capture INSERT actions, so update will not come to streams.

```

55
56 update members_raw set fee=10 where id=4;
57
58 select * from members_append_stream;
59
60 INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM members_append_stream WHERE METADATA$ACTION = 'INSERT';

```

Results Data Preview

✓ Query ID SQL 380ms 1 rows

Filter result...

Copy

Columns

Row	number of rows updated	number of multi-joined rows updated
1	1	0

```

58 select * from members_append_stream;
59
60 INSERT INTO members_prod(id,name,fee) SELECT id, name, fee FROM members_append_stream WHERE METADATA$ACTION = 'INSERT';
61

```

Results Data Preview Open History

✓ Query_ID SQL 172ms 0 rows

Filter result... Columns ▾

Row	ID	NAME	FEE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID

Streams in transaction:

```

use role sysadmin;
use database ecommerce_db;

create or replace schema streams_test;

--- Create a raw table to test the streams ---
CREATE OR REPLACE TABLE members_raw (
    id number(8) NOT NULL,
    name varchar(255) default NULL,
    fee number(3) NULL,
    member_type varchar(10) not null
);

--- Create a production table which will consume the streams data ---
CREATE OR REPLACE TABLE members_music_prod (
    id number(8) NOT NULL,
    name varchar(255) default NULL,
    fee number(3) NULL
);

CREATE OR REPLACE TABLE members_games_prod (
    id number(8) NOT NULL,
    name varchar(255) default NULL,
    fee number(3) NULL
);

--- Create a standard (delta) stream on the raw table :members_raw---
CREATE OR REPLACE STREAM members_std_stream ON TABLE members_raw;

--- Check the streams ---
select * from members_std_stream

--- Check the stream offset ---
SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream') as
members_table_st_offset;

SELECT

```

```

to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as
members_table_st_offset;

--- Insert some data into the raw table : members_raw ---

INSERT INTO members_raw (id,name,fee,member_type)
VALUES
(1,'Joe',0,'games'),
(2,'Jane',0,'music'),
(3,'George',0,'games'),
(4,'Betty',0,'games'),
(5,'Sally',0,'music');

--- Check the streams ----
select * from members_std_stream

--- Check the stream offset ----
SELECT SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream') as
members_table_st_offset;

--- Query the streams data by ingesting the CDC streams data into the production table ----
SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION =
'INSERT';

--- Beging the transaction & consume the streams data ----

begin ;

insert into members_games_prod
SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION =
'INSERT' and member_type='games';

insert into members_music_prod
SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION =
'INSERT' and member_type='music';

commit;

--- Check the production tables ----
select * from members_games_prod;
select * from members_music_prod;

--- Check the offset ----
SELECT
to_timestamp(SYSTEM$STREAM_GET_TABLE_TIMESTAMP('members_std_stream')) as
members_table_st_offset;

```

```

26
27 CREATE OR REPLACE STREAM members_std_stream ON TABLE members_raw;
28
29
30 INSERT INTO members_raw (id, name, fee, member_type)

```

Results Data Preview

✓ Query_ID SQL 159ms 1 rows

Filter result...



Copy

Row	status
1	Stream MEMBERS_STD_STREAM successfully created.

```

30 INSERT INTO members_raw (id, name, fee, member_type)
31 VALUES
32 (1, 'Joe', 0, 'games'),
33 (2, 'Jane', 0, 'music'),
34 (3, 'George', 0, 'games'),
35 (4, 'Betty', 0, 'games'),
36 (5, 'Sally', 0, 'music');
37
38 select * from members_std_stream;
39
40 begin;

```

Results Data Preview

✓ Query_ID SQL 1.14s 1 rows

Filter result...



Copy

Row
1

```

37
38 select * from members_std_stream;
39
40 begin;
41
42 insert into members_games_prod
43 SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT' and member_type='games';

```

Results Data Preview

Open History

✓ Query_ID SQL 519ms 5 rows

Filter result...



Copy

Columns ▾

Row	ID	NAME	FEES	MEMBER_TYPE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	1	Joe	0	games	INSERT	FALSE	c486986981b378410...
2	2	Jane	0	music	INSERT	FALSE	b17f706177dfc3b63d...
3	3	George	0	games	INSERT	FALSE	0ed720b3122ca5c5b...
4	4	Betty	0	games	INSERT	FALSE	25982a442db0934f4...
5	5	Sally	0	music	INSERT	FALSE	420f4e59508191bba6..

```

41
42 insert into members_games$prod
43 SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT' and member_type='games';
44
45 insert into members_music_prod
46 SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT' and member_type='music';

```

Results Data Preview

✓ Query ID SQL 779ms 1 rows

Filter result...

Row
1

As we didn't commit the transaction yet, the stream will still be available in the streams.

```

37
38 select * from members_std_stream;
39
40 begin;
41
42 insert into members_games_prod
43 SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT' and member_type='games';
44
45 insert into members_music_prod
46 SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT' and member_type='music';
47
48 commit;
49
50

```

Results Data Preview [Open History](#)

✓ Query ID SQL 258ms 5 rows

Filter result... Columns ▾

Row	ID	NAME	FEE	MEMBER_TYPE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	1	Joe	0	games	INSERT	FALSE	c486986981b378410...
2	2	Jane	0	music	INSERT	FALSE	b17f706177dfc3b63d...
3	3	George	0	games	INSERT	FALSE	0ed720b3122ca5c5b...
4	4	Betty	0	games	INSERT	FALSE	25982a442db0934f4...
5	5	Sally	0	music	INSERT	FALSE	420f4e59508191bba6...

After commit and check, you may see the streams become empty.

```

47
48 commit;
49
50

```

Results Data Preview

✓ Query ID SQL 421ms 1 rows

Filter result...

Row	status
1	Statement executed successfully.

```

38 select * from members_std_stream;
39
40 begin;
41 |
42 insert into members_games_prod
43 SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT' and member_type='games';
44
45 insert into members_music_prod
46 SELECT id, name, fee FROM members_std_stream WHERE METADATA$ACTION = 'INSERT' and member_type='music';
47
48 commit;
49
50

```

results Data Preview Open History

✓ Query ID SQL 609ms 0 rows

Filter result... Copy Columns ▾

Row	ID	NAME	FEE	MEMBER_TYPE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID

Note: The normal SQL statements are auto-commit by default.

ChangeTracking:

```

use database ecommerce_db;

create or replace schema streams_test;

CREATE OR REPLACE TABLE members_raw (
    id number(8) NOT NULL,
    name varchar(255) default NULL,
    fee number(3) NULL,
    member_type varchar(10) not null
);

--- Enable change tracking ----
ALTER TABLE members_raw SET CHANGE_TRACKING = TRUE;

SET ts1 = (SELECT CURRENT_TIMESTAMP());

INSERT INTO members_raw (id,name,fee,member_type)
VALUES
(1,'Joe',0,'games'),
(2,'Jane',0,'music'),
(3,'George',0,'games');

update members_raw set fee=20 where id=3;

SELECT * FROM members_raw CHANGES(INFORMATION => default) at(TIMESTAMP =>$ts1);

SET ts2 = (SELECT CURRENT_TIMESTAMP());

INSERT INTO members_raw (id,name,fee,member_type)
VALUES

```

```
(4,'Betty',0,'games'),  
(5,'Sally',0,'music');
```

```
SELECT * FROM members_raw CHANGES(INFORMATION => default) at(TIMESTAMP =>$ts1);
```

```
SELECT * FROM members_raw CHANGES(INFORMATION => default) at(TIMESTAMP => $ts2);
```

```
11  
12 ALTER TABLE members_raw SET CHANGE_TRACKING = TRUE;  
13  
14 SET ty1 = (SELECT CURRENT_TIMESTAMP());  
15  
16 INSERT INTO members_raw (id, name, fee, member_type)
```

Results Data Preview

✓ Query ID SQL 185ms 1 rows

Filter result...



Copy

Row	status
1	Statement executed successfully.

```
13  
14 SET ty1 = (SELECT CURRENT_TIMESTAMP());  
15  
16 INSERT INTO members_raw (id, name, fee, member_type)
```

Results Data Preview

✓ Query ID SQL 38ms 1 rows

Filter result...



Copy

Row	status
1	Statement executed successfully.

```

15
16 INSERT INTO members_raw (id,name,fee,member_type)
17 VALUES
18 (1,'Joe',0,'games'),
19 (2,'Jane',0,'music'),
20 (3,'George',0,'games');
21
22 update members_raw set fee=20 where id=3;
23

```

Results Data Preview

[Query ID](#) [SQL](#) 596ms 1 rows

Filter result...



[Copy](#)

Row

1

```

21
22 update members_raw set fee=20 where id=3;
23

```

Results Data Preview

[Open History](#)

[Query ID](#) [SQL](#) 465ms 1 rows

Filter result...



[Copy](#)

[Columns](#)

Row

number of rows updated

number of multi-joined rows updated

1

1

0

In the below command, we are saying to select all changes (passing default as a param to information means actions includes both new inserted data and updated.) between the timestamps(ts1 and now).

```

23
24 SELECT * FROM members_raw CHANGES(INFORMATION => default) at(TIMESTAMP =>\$ts1);
25

```

Results Data Preview

[Open History](#)

[Query ID](#) [SQL](#) 3 rows

Filter result...



[Copy](#)

[Columns](#)

Row	ID	NAME	Fee	Member_Type	Metadata\$action	Metadata\$isUpdate	Metadata\$row_id
1	1	Joe	0	games	INSERT	FALSE	5906c9ee8834c4270...
2	2	Jane	0	music	INSERT	FALSE	c82f86ba1993d34118...
3	3	George	20	games	INSERT	FALSE	ce44f9dd68ee4cb47...

To get newly inserted records alone:

```

24 SELECT * FROM members_raw CHANGES(INFORMATION => append_only) at(TIMESTAMP =>$ts1);
25
26
27 SET ts2 = (SELECT CURRENT_TIMESTAMP());
28

```

results Data Preview

✓ [Query ID](#) [SQL](#) 188ms 3 rows [Copy](#)

Filter result... [Columns](#)

Row	ID	NAME	FEE	MEMBER_TYPE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	1	Joe	0	games	INSERT	FALSE	5906c9ee8834c4270...
2	2	Jane	0	music	INSERT	FALSE	c82f86ba1993d34118...
3	3	George	0	games	INSERT	FALSE	ce44f9dd68ee4cb47...

```

26
27 SET ts2 = (SELECT CURRENT_TIMESTAMP());
28
29 INSERT INTO members_raw (id,name,fee,member_type)
30 VALUES
31 (4,'Betty',0,'games')

```

results Data Preview

✓ [Query ID](#) [SQL](#) 100ms 1 rows [Copy](#)

Filter result... [Columns](#)

Row	status
1	Statement executed successfully.

```

28
29 INSERT INTO members_raw (id,name,fee,member_type)
30 VALUES
31 (4,'Betty',0,'games'),
32 (5,'Sally',0,'music');
33
34

```

Results Data Preview

✓ [Query ID](#) [SQL](#) 514ms 1 rows [Copy](#)

Filter result... [Columns](#)

Row
1

The screenshot shows two separate queries run in the Snowflake Data Preview interface.

Query 1:

```

34 SELECT * FROM members_raw CHANGES(INFORMATION => append_only) at(TIMESTAMP =>$ts1);
35
36
37
38

```

Results:

Row	ID	NAME	FEE	MEMBER_TYPE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	4	Betty	0	games	INSERT	FALSE	1eed0ba4a0b42ad0d...
2	5	Sally	0	music	INSERT	FALSE	72b53c0ecd267cc88...
3	1	Joe	0	games	INSERT	FALSE	5906c9ee8834c4270...
4	2	Jane	0	music	INSERT	FALSE	c82f86ba1993d34118...
5	3	George	0	games	INSERT	FALSE	ce44f9dd68ee4cb47...

Query 2:

```

37 SELECT * FROM members_raw CHANGES(INFORMATION => default) at(TIMESTAMP => $ts2);
38

```

Results:

Row	ID	NAME	FEE	MEMBER_TYPE	METADATA\$ACTION	METADATA\$ISUPDATE	METADATA\$ROW_ID
1	4	Betty	0	games	INSERT	FALSE	1eed0ba4a0b42ad0d...
2	5	Sally	0	music	INSERT	FALSE	72b53c0ecd267cc88...

Streams Project solution:

Without Snowpipe (manually inserting records to raw table from stage)

```

use role accountadmin;

CREATE OR REPLACE STORAGE INTEGRATION aws_sf_data
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = S3
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN =
  'arn:aws:iam::682470999581:role/aws-saml-bsys-ems-dev-developer'
  STORAGE_ALLOWED_LOCATIONS = ('s3://ems-pt-result/ecommerce_dev/');

grant usage on integration aws_sf_data to role sysadmin;

grant execute task on account to role sysadmin;

grant create stage on schema "ECOMMERCE_DB"."ECOMMERCE_DEV" to role sysadmin;

use role sysadmin;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

desc INTEGRATION aws_sf_data;

```

```

CREATE OR REPLACE FILE FORMAT json_load_format TYPE = 'JSON' ;

create or replace stage stg_lineitem_json_dev
storage_integration = aws_sf_data
url = 's3://ems-pt-result/ecommerce_devstreams_dev/'
file_format = json_load_format;

list @stg_lineitem_json_dev;

create or replace table lineitem_raw_json (src variant );

CREATE OR REPLACE STREAM lineitem_std_stream ON TABLE lineitem_raw_json;

create or replace task lineitem_load_tsk
warehouse = compute_wh
schedule = '1 minute'
when system$stream_has_data('lineitem_std_stream')
as
merge into lineitem as li
using
(
  select
    SRC:L_ORDERKEY as L_ORDERKEY,
    SRC:L_PARTKEY as L_PARTKEY,
    SRC:L_SUPPKEY as L_SUPPKEY,
    SRC:L_LINENUMBER as L_LINENUMBER,
    SRC:L_QUANTITY as L_QUANTITY,
    SRC:L_EXTENDEDPRICE as L_EXTENDEDPRICE,
    SRC:L_DISCOUNT as L_DISCOUNT,
    SRC:L_TAX as L_TAX,
    SRC:L_RETURNFLAG as L_RETURNFLAG,
    SRC:L_LINESTATUS as L_LINESTATUS,
    SRC:L_SHIPDATE as L_SHIPDATE,
    SRC:L_COMMITDATE as L_COMMITDATE,
    SRC:L_RECEIPTDATE as L_RECEIPTDATE,
    SRC:L_SHIPINSTRUCT as L_SHIPINSTRUCT,
    SRC:L_SHIPMODE as L_SHIPMODE,
    SRC:L_COMMENT as L_COMMENT
  from
    lineitem_std_stream
  where metadata$action='INSERT'
) as li_stg
on li.L_ORDERKEY = li_stg.L_ORDERKEY and li.L_PARTKEY = li_stg.L_PARTKEY and
li.L_SUPPKEY = li_stg.L_SUPPKEY
when matched then update
set
  li.L_PARTKEY = li_stg.L_PARTKEY,
  li.L_SUPPKEY = li_stg.L_SUPPKEY,

```

```

    li.L_LINENUMBER = li_stg.L_LINENUMBER,
    li.L_QUANTITY = li_stg.L_QUANTITY,
    li.L_EXTENDEDPRICE = li_stg.L_EXTENDEDPRICE,
    li.L_DISCOUNT = li_stg.L_DISCOUNT,
    li.L_TAX = li_stg.L_TAX,
    li.L_RETURNFLAG = li_stg.L_RETURNFLAG,
    li.L_LINESTATUS = li_stg.L_LINESTATUS,
    li.L_SHIPDATE = li_stg.L_SHIPDATE,
    li.L_COMMITDATE = li_stg.L_COMMITDATE,
    li.L_RECEIPTDATE = li_stg.L_RECEIPTDATE,
    li.L_SHIPINSTRUCT = li_stg.L_SHIPINSTRUCT,
    li.L_SHIPMODE = li_stg.L_SHIPMODE,
    li.L_COMMENT = li_stg.L_COMMENT
when not matched then insert
(
    L_ORDERKEY,
    L_PARTKEY,
    L_SUPPKEY,
    L_LINENUMBER,
    L_QUANTITY,
    L_EXTENDEDPRICE,
    L_DISCOUNT,
    L_TAX,
    L_RETURNFLAG,
    L_LINESTATUS,
    L_SHIPDATE,
    L_COMMITDATE,
    L_RECEIPTDATE,
    L_SHIPINSTRUCT,
    L_SHIPMODE,
    L_COMMENT
)
values
(
    li_stg.L_ORDERKEY,
    li_stg.L_PARTKEY,
    li_stg.L_SUPPKEY,
    li_stg.L_LINENUMBER,
    li_stg.L_QUANTITY,
    li_stg.L_EXTENDEDPRICE,
    li_stg.L_DISCOUNT,
    li_stg.L_TAX,
    li_stg.L_RETURNFLAG,
    li_stg.L_LINESTATUS,
    li_stg.L_SHIPDATE,
    li_stg.L_COMMITDATE,
    li_stg.L_RECEIPTDATE,
    li_stg.L_SHIPINSTRUCT,
    li_stg.L_SHIPMODE,
    li_stg.L_COMMENT
)

```

```

);
show tasks;
alter task lineitem_load_tsk resume;
copy into lineitem_raw_json from @stg_lineitem_json_dev ON_ERROR =
ABORT_STATEMENT;
select *
from table(information_schema.task_history(
    scheduled_time_range_start=>dateadd('hour',-1,current_timestamp()),
    result_limit => 100));

```

```

2   CREATE OR REPLACE STORAGE INTEGRATION aws_sf_data
3     TYPE = EXTERNAL_STAGE
4       STORAGE_PROVIDER = S3
5         ENABLED = TRUE
6           STORAGE_AWS_ROLE_ARN = 'arn:aws:iam::682470999581:role/aws-saml-bsys-ems-dev-developer'
7             STORAGE_ALLOWED_LOCATIONS = ('s3://ems-pt-result/ecommerce_dev/');
8
9
10  grant usage on integration aws_sf_data to role sysadmin;
11

```

↳ Results **Chart**

status		Query Details
1	Integration AWS_SF_DATA successfully created.	Query duration 13:

```

10  grant usage on integration aws_sf_data to role sysadmin;
11

```

↳ Results **Chart**

status		Query Details
1	Statement executed successfully.	Query duration

```

12  grant execute task on account to role sysadmin;
13

```

↳ Results **Chart**

status		Query Details
1	Statement executed successfully.	Query duration

```

14 | grant create stage on schema "ECOMMERCE_DB"."ECOMMERCE_DEV" to role sysadmin;
15 |
16 | use role sysadmin;
17 |
18 | use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";
19 |
20 | desc INTEGRATION aws_sf_data;

```

Results

status	
1	Statement executed successfully.

Query Details

Query duration

Results

status	
1	Statement executed successfully.

Query Details

Query duration

Results

status	
1	Statement executed successfully.

Query Details

Query duration

Results

property	property_type	property_value
5	String	arn:aws:iam::180294178125:user/w57r00
6	String	arn:aws:iam::682470999581:role/aws-sam
7	String	JXB38456_SFCRole=5_HZRbs+cZXWLEoA5

Query Details

Query duration

Rows

Query ID 01b7

Add the external id and user arn in role policy.

IAM > Roles > snowflake-aws-role > Edit trust policy

Edit trust policy

```
1▼ {
2    "Version": "2012-10-17",
3▼   "Statement": [
4▼     {
5         "Effect": "Allow",
6▼         "Principal": {
7             "AWS": "arn:aws:iam::206954943453:user/hndy-s-eusb7117"
8         },
9         "Action": "sts:AssumeRole",
10▼        "Condition": {
11▼            "StringEquals": {
12                "sts:ExternalId": "AZ32171_SFCRole=3/_jzjDtdUZ5+GIxI3GWrbZmGYayc-"
13            }
14        }
15    }
16 ]
17 }
```

```
21 | CREATE OR REPLACE FILE FORMAT json_load_format TYPE = 'JSON' ;
22 |
23 |
```

Results		Chart	Query Details
	status		Query duration
1	File format JSON_LOAD_FORMAT successfully created.		

```
23  
24 | create or replace stage stg_lineitem_json_dev  
25 | storage_integration = aws_sf_data  
26 | url = 's3://ems-pt-result/ecommerce_devstreams_dev/'  
27 | file_format = json_load_format;  
28
```

Results							
	Chart						
	<table><thead><tr><th></th><th>status</th><th>Query Details</th></tr></thead><tbody><tr><td>1</td><td>Stage area STG_LINEITEM_JSON_DEV successfully created.</td><td>Query duration</td></tr></tbody></table>		status	Query Details	1	Stage area STG_LINEITEM_JSON_DEV successfully created.	Query duration
	status	Query Details					
1	Stage area STG_LINEITEM_JSON_DEV successfully created.	Query duration					

```

36
37 create or replace task lineitem_load_tsk
38 warehouse = compute_wh
39 schedule = '1 minute'
40 when system$stream_has_data('lineitem_std_stream')
41 as
42 merge into lineitem as li
43 using
44 (
45     select
46         SRC:L_ORDERKEY as L_ORDERKEY,
47         SRC:L_PARTKEY as L_PARTKEY,
48         SRC:L_SUPPKEY as L_SUPPKEY,
49         SRC:L_LINENUMBER as L_LINENUMBER,
50         SRC:L_QUANTITY as L_QUANTITY,
51         SRC:L_EXTENDEDPRICE as L_EXTENDEDPRICE,
52         SRC:L_DISCOUNT as L_DISCOUNT,
53         SRC:L_TAX as L_TAX,
54         SRC:L_RETURNFLAG as L_RETURNFLAG.

```

Results Data Preview

✓ Query ID SQL 101ms 1 rows

Filter result... [Download](#) [Copy](#)

Row	status
1	Task LINEITEM_LOAD_TSK successfully created.

```

124
125 alter task lineitem_load_tsk resume;
126
127 copy into lineitem_raw_json from @stg_lineitem_json_dev ON_ERROR = ABORT_STATEMENT;
128
129 select *
130   from table(information_schema.task_history(
131     scheduled_time_range_start=>dateadd('hour', -1, current_timestamp()))

```

Results Data Preview

✓ Query ID SQL 89ms 1 rows

Filter result... [Download](#) [Copy](#)

Row	status
1	Statement executed successfully.

```

126
127 copy into lineitem_raw_json from @stg_lineitem_json_dev ON_ERROR = ABORT_STATEMENT;
128
129 select * from lineitem_std_stream limit 10;
130
131 select *
132   from table(information_schema.task_history(
133     scheduled_time_range_start=>dateadd('hour', -1, current_timestamp()),
134     result_limit => 100));
135

```

Results Data Preview [Open Hist](#)

✓ Query ID SQL 2s 1 rows

Filter result... [Download](#) [Copy](#) Columns ▾

Row	file	status	rows_parsed	rows_loaded	↓ error_limit	errors_seen	first_error	first_error_line	first_error_character	first_error_col
1	s3://sid-snow...	LOADED	7936	7936	1	0	NULL	NULL	NULL	NULL

```

131 select * from lineitem_std_stream limit 10;
132
133 select *
134   from table(information_schema.task_history(
135     scheduled_time_range_start=>dateadd('hour', -1, current_timestamp()),
136     result_limit => 100));
137

```

Results Data Preview [Open History](#)

Query_ID SQL 137ms 10 rows

Filter result... [Copy](#) Columns ▾

Row	Src	Metadata\$action	Metadata\$isupdate	Metadata\$Row_id
1	{"L_COMMENT": " across the busy reque..."}	INSERT	FALSE	3ca90d48ba01dde1be388816269bf38b...
2	{"L_COMMENT": "ts; packages haggle fu..."}	INSERT	FALSE	b2251c198691112d7274a28306bde0f8f...
3	{"L_COMMENT": "wake slyly permanen", ...}	INSERT	FALSE	e53ee75925636ea8cdf492a88ec52cd90...


```

132
133 select *
134   from table(information_schema.task_history(
135     scheduled_time_range_start=>dateadd('hour', -1, current_timestamp()),
136     result_limit => 10));
137

```

Results Data Preview [Open History](#)

Query_ID SQL 614ms 10 rows

Filter result... [Copy](#) Columns ▾

Row	Query_id	Name	Database_name	Schema_name	Query_Text	Condition_Te	State	Error_Code	Error_Messag	Scheduled_Ti	Query_L
1	NULL	LINEITEM_L...	ECOMMERCE...	ECOMMERCE...	merge into li...	system\$stre...	SCHEDULED	NULL	NULL	2022-03-05...	
2	01a2b900-3...	LINEITEM_L...	ECOMMERCE...	ECOMMERCE...	merge into li...	system\$stre...	SUCCEEDED	NULL	NULL	2022-03-05...	2022-03
3	NULL	LINEITEM_L...	ECOMMERCE...	ECOMMERCE...	merge into li...	system\$stre...	SKIPPED	0010000	0010000	2022-03-05...	

With snowpipe(automated inserting records into raw table from stage)

```

create or replace task lineitem_load_tsk
warehouse = compute_wh
schedule = '1 minute'
when system$stream_has_data('lineitem_std_stream')
as
merge into lineitem as li
using
(
  select
    SRC:L_ORDERKEY as L_ORDERKEY,
    SRC:L_PARTKEY as L_PARTKEY,
    SRC:L_SUPPKEY as L_SUPPKEY,
    SRC:L_LINENUMBER as L_LINENUMBER,
    SRC:L_QUANTITY as L_QUANTITY,
    SRC:L_EXTENDEDPRICE as L_EXTENDEDPRICE,
    SRC:L_DISCOUNT as L_DISCOUNT,
    SRC:L_TAX as L_TAX,
    SRC:L_RETURNFLAG as L_RETURNFLAG,
    SRC:L_LINESTATUS as L_LINESTATUS,
    SRC:L_SHIPDATE as L_SHIPDATE,
    SRC:L_COMMITDATE as L_COMMITDATE,
    SRC:L_RECEIPTDATE as L_RECEIPTDATE,
    SRC:L_SHIPINSTRUCT as L_SHIPINSTRUCT,

```

```

SRC:L_SHIPMODE as L_SHIPMODE,
SRC:L_COMMENT as L_COMMENT
from
lineitem_std_stream
where metadata$action='INSERT'
) as li_stg
on li.L_ORDERKEY = li_stg.L_ORDERKEY and li.L_PARTKEY = li_stg.L_PARTKEY and
li.L_SUPPKEY = li_stg.L_SUPPKEY
when matched then update
set
    li.L_PARTKEY = li_stg.L_PARTKEY,
    li.L_SUPPKEY = li_stg.L_SUPPKEY,
    li.L_LINENUMBER = li_stg.L_LINENUMBER,
    li.L_QUANTITY = li_stg.L_QUANTITY,
    li.L_EXTENDEDPRICE = li_stg.L_EXTENDEDPRICE,
    li.L_DISCOUNT = li_stg.L_DISCOUNT,
    li.L_TAX = li_stg.L_TAX,
    li.L_RETURNFLAG = li_stg.L_RETURNFLAG,
    li.L_LINESTATUS = li_stg.L_LINESTATUS,
    li.L_SHIPDATE = li_stg.L_SHIPDATE,
    li.L_COMMITDATE = li_stg.L_COMMITDATE,
    li.L_RECEIPTDATE = li_stg.L_RECEIPTDATE,
    li.L_SHIPINSTRUCT = li_stg.L_SHIPINSTRUCT,
    li.L_SHIPMODE = li_stg.L_SHIPMODE,
    li.L_COMMENT = li_stg.L_COMMENT
when not matched then insert
(
    L_ORDERKEY,
    L_PARTKEY,
    L_SUPPKEY,
    L_LINENUMBER,
    L_QUANTITY,
    L_EXTENDEDPRICE,
    L_DISCOUNT,
    L_TAX,
    L_RETURNFLAG,
    L_LINESTATUS,
    L_SHIPDATE,
    L_COMMITDATE,
    L_RECEIPTDATE,
    L_SHIPINSTRUCT,
    L_SHIPMODE,
    L_COMMENT
)
values
(
    li_stg.L_ORDERKEY,
    li_stg.L_PARTKEY,
    li_stg.L_SUPPKEY,
    li_stg.L_LINENUMBER,

```

```

    li_stg.L_QUANTITY,
    li_stg.L_EXTENDEDPRICE,
    li_stg.L_DISCOUNT,
    li_stg.L_TAX,
    li_stg.L_RETURNFLAG,
    li_stg.L_LINESTATUS,
    li_stg.L_SHIPDATE,
    li_stg.L_COMMITDATE,
    li_stg.L_RECEIPTDATE,
    li_stg.L_SHIPINSTRUCT,
    li_stg.L_SHIPMODE,
    li_stg.L_COMMENT
);

show tasks;

alter task lineitem_load_tsk resume;

copy into lineitem_raw_json from @stg_lineitem_json_dev ON_ERROR =
ABORT_STATEMENT;

select *
from table(information_schema.task_history(
scheduled_time_range_start=>dateadd('hour',-1,current_timestamp()),
result_limit => 100));

```

create or replace pipe lineitem_pipe auto_ingest=true as
copy into lineitem from @stg_lineitem_json_dev ON_ERROR = ABORT_STATEMENT;

```

144 create or replace pipe lineitem_pipe auto_ingest=true as
145 copy into lineitem_raw_json from @stg_lineitem_json_dev;
146
147 show pipes;
148
149 select count(1) from lineitem_raw_json ;
150
151 select * from information_schema.load_history where table_name='LINEITEM' order by last_load_time desc limit

```

Results Data Preview

✓ Query ID SQL 1.13s 1 rows

Filter result... Copy

Row	status
1	Pipe LINEITEM_PIPE successfully created.

Create event notification [Info](#)

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

General configuration

Event name

Event name can contain up to 255 characters.

Prefix - optional

Limit the notifications to objects with key starting with specified characters.

Suffix - optional

Limit the notifications to objects with key ending with specified characters.

Event types

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

Destination

 Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. [Learn more](#)

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:eu-central-1:206954943453:sf-snowpipe-AIDATAL3Z2PO3TJHHUZYK-brA
```

After some json file insertion and few seconds, we can see the records added to raw table.

```
149 select count(1) from lineitem_raw_json;
150
151 select
152
153
154 from cube_lineitem_sf_snowpipe_history(
155 date_range_start, date_end,
156 pipe_name)
157
158
159
```

results Data Preview ← Open History

Query ID	SQL	101ms	1 rows
✓	<code>select count(1) from lineitem_raw_json;</code>		

Filter result... Columns ▾

Row	COUNT(1)
1	1936

UDF:

SQL(Scalar/Tabular) UDF:

```
use role accountadmin;
use warehouse compute_wh;
use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";
```

```

-- Simple UDF ----
CREATE or replace FUNCTION sum_values(a number,b number)
RETURNS number
LANGUAGE SQL
AS
$$
  SELECT a+b as res
$$;

select sum_values(2,3);

SELECT SUM(l_quantity) as total_quantity_shipped
  FROM "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM"
 where L_SHIPDATE ='1996-02-06' and l_suppkey=2892071;

---Scalar Function to calculate sales quantity by Supplier ---
CREATE or replace FUNCTION sales_qty_by_supplier(ship_date date, supplier_key integer)
RETURNS NUMERIC(11,2)
AS
$$
  SELECT SUM(l_quantity) as total_quantity_shipped
    FROM "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM"
   where L_SHIPDATE =ship_date and l_suppkey=supplier_key
$$;

--- Use the above Scalar function in a query ----
select sales_qty_by_supplier('1996-02-06',2892071);

select
  sales_qty_by_supplier('1996-02-06',l_suppkey) as supplier_sales,
  l_suppkey
from
  "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM"
where supplier_sales>0;

select
  lt.L_SUPPKEY ,
  sum(l_QUANTITY) as qty_sold
from
  "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."SUPPLIER" sp
join
  "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" lt on sp.S_SUPPKEY =
lt.L_SUPPKEY
  where lt.l_shipdate='1997-03-19' and lt.L_SUPPKEY=4072277
  group by 1
  having qty_sold > 0;

--- Tabular UDF ----

```

```

CREATE or replace FUNCTION sales_qty_by_supplier(ship_date varchar,supplier_key
number)
RETURNS table(supplier_key number,qty_sold number)
AS
$$
select
    lt.L_SUPPKEY ,
    sum(L_QUANTITY) as qty_sold
from
    "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."SUPPLIER" sp
join
    "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF10"."LINEITEM" lt on sp.S_SUPPKEY =
lt.L_SUPPKEY
where lt.l_shipdate=ship_date and lt.L_SUPPKEY=supplier_key
group by 1
having qty_sold > 0
$$;

-- Call the above tabular UDTF ---
select * from table (sales_qty_by_supplier('1996-06-27',23661));

```

```

5   -- Simple UDF ----
6   CREATE or replace FUNCTION sum_values(a number ,b number )
7       RETURNS number
8       LANGUAGE SQL
9       AS
10      $$
11         SELECT a+b as res
12      $$;
13

```

	status	Query Det
1	Function SUM_VALUES successfully created.	Query dur:

```

13
14   select sum_values(2,3);
15

```

	SUM_VALUES(2,3)
1	5

```

20 ---Scalar Function to calculate sales quantity by Supplier ---
21 CREATE or replace FUNCTION sales_qty_by_supplier(ship_date date,supplier_key integer)
22 RETURNS NUMERIC(11,2)
23 AS
24 $$
25     SELECT SUM(l_quantity) as total_quantity_shipped
26         FROM "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM"
27     where L_SHIPDATE =ship_date and l_suppkey=supplier_key
28 $$;
29
30

```

Results		Chart	Query Details
	status	Query duration	
1	Function SALES_QTY_BY_SUPPLIER successfully created.		

```

30 select sales_qty_by_supplier('1996-02-06',2892071);

```

Results		Chart	Query Details
	SALES_QTY_BY_SUPPLIER('1996-02-06',2892071)	Query duration	
1	12.00		

```

36 select
37     sales_qty_by_supplier('1996-02-06',l_suppkey) as supplier_sales,
38     l_suppkey
39 from
40     "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM"
41 where supplier_sales>0;

```

	SUPPLIER_SALES	L_SUPPKEY	Query Details
	Query duration	Rows	Query ID
1	22.00	7444146	
2	12.00	3313933	
3	45.00	2104048	
4	31.00	1357672	
5	29.00	5654620	
6	41.00	7500950	
7	4.00	1145801	

```

43  select
44      lt.L_SUPPKEY ,
45      sum(L_QUANTITY)  as qty_sold
46  from
47      "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."SUPPLIER" sp
48  join
49      "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."LINEITEM" lt on sp.S_SUPPKEY = lt.L_SUPPKEY
50  where lt.l_shipdate='1997-03-19' and lt.L_SUPPKEY=4072277
51  group by 1
52  having qty_sold > 0;

```

↳ Results ↵ Chart 🔍 📈

	L_SUPPKEY	QTY SOLD	Query Details
1	4072277	32.00	Query duration

```

54  --- Tabular UDF ---
55  CREATE or replace FUNCTION sales_qty_by_supplier(ship_date varchar,supplier_key number)
56  RETURNS table(supplier_key number,qty_sold number)
57  AS
58  $$
59  select
60      lt.L_SUPPKEY ,
61      sum(L_QUANTITY) as qty_sold
62  from
63      "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF1000"."SUPPLIER" sp
64  join

```

↳ Results ↵ Chart 🔍 📈 ⏴ ⏵

	status	Query Details
1	Function SALES_QTY_BY_SUPPLIER successfully created.	Query duration 15

```

71  -- Call the above tabular UDTF ---
72  | select * from table (sales_qty_by_supplier('1996-06-27',23661));

```

↳ Results ↵ Chart 🔍 📈 ⏴ ⏵ ⏴ ⏵

	SUPPLIER_KEY	QTY SOLD	Query Details
1	23661	20	Query duration 2.2s

Javascript UDF:

```

use role accountadmin;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

CREATE OR REPLACE FUNCTION test_date_js(A string)
RETURNS OBJECT
LANGUAGE JAVASCRIPT

```

```

STRICT
AS
$$
function test_js_func(x)
{
    var now = new Date(x);
    return now
}
return test_js_func(A);
$$;

select test_date_js('2022-02-01');

CREATE OR REPLACE FUNCTION simple_js_func(A float)
RETURNS OBJECT
LANGUAGE JAVASCRIPT
STRICT
AS
$$
function test_function(x){
    var lr = {};
    lr['a'] = x*2;
    lr['b'] = x*3;
    return lr;
}
return test_function(A);
$$;

SELECT simple_js_func(2.2);

CREATE OR REPLACE FUNCTION range_to_values(PREFIX VARCHAR, RANGE_START
FLOAT, RANGE_END FLOAT)
RETURNS TABLE (IP_ADDRESS VARCHAR)
LANGUAGE JAVASCRIPT
AS $$
{
    processRow: function f(row, rowWriter, context) {
        var suffix = row.RANGE_START;
        while (suffix <= row.RANGE_END) {
            rowWriter.writeRow( {IP_ADDRESS: row.PREFIX + "." + suffix} );
            suffix = suffix + 1;
        }
    }
}
$$;
SELECT * FROM TABLE(range_to_values('192.168.1', 42::FLOAT, 45::FLOAT));

```

```

4
5   CREATE OR REPLACE FUNCTION test_date_js(A string)
6     RETURNS OBJECT
7     LANGUAGE JAVASCRIPT
8     STRICT
9     AS
10    $$
11    function test_js_func(x)
12    {
13      var now = new Date(x);
14      return now

```

↳ Results ⚡ Chart

status
1 Function TEST_DATE_JS successfully created.

```

18
19   | select test_date_js('2022-02-01');
20

```

↳ Results ⚡ Chart

TEST_DATE_JS('2022-02-01')
1 "2022-01-31 16:00:00.000 -0800"

```

22   CREATE OR REPLACE FUNCTION simple_js_func(A float)
23     RETURNS OBJECT
24     LANGUAGE JAVASCRIPT
25     STRICT
26     AS
27     $$
28     function test_function(x){
29       var lr = {};
30       lr['a'] = x*2;
31       lr['b'] = x*3;
32       return lr;

```

↳ Results ⚡ Chart

status
1 Function SIMPLE_JS_FUNC successfully created.

Query Details ...
Query duration 80ms

```

37 | SELECT simple_js_func(2.2);
38 |

```

Results Chart Query Details ...
Query duration 118ms

	SIMPLE_JS_FUNC(2.2)
1	{ "a": 4.40000000000000e+00, "b": 6.600000000000001e+00}


```

58
39 | CREATE OR REPLACE FUNCTION range_to_values(PREFIX VARCHAR, RANGE_START FLOAT, RANGE_END FLOAT)
40 | RETURNS TABLE (IP_ADDRESS VARCHAR)
41 | LANGUAGE JAVASCRIPT
42 | AS $$
43 | {
44 |     processRow: function f(row, rowWriter, context) {
45 |         var suffix = row.RANGE_START;
46 |         while (suffix <= row.RANGE_END) {
47 |             rowWriter.writeRow( {IP_ADDRESS: row.PREFIX + "." + suffix} );
48 |             suffix = suffix + 1;

```

Results Chart Query Details ...
Query duration 100ms

	status
1	Function RANGE_TO_VALUES successfully created.


```

52 | $$
53 | SELECT * FROM TABLE(range_to_values('192.168.1', 42::FLOAT, 45::FLOAT));

```

Results Chart Query Details ...
Query duration 157ms
Rows 01b2 Ask Copilot

	IP_ADDRESS
1	192.168.1.42
2	192.168.1.43
3	192.168.1.44
4	192.168.1.45

Secure Pushdown filter from exposing data:

```

use role accountadmin;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

create role udf_role;

grant usage on warehouse compute_wh to role udf_role;

grant usage on database ECOMMERCE_DB to role udf_role;

grant usage on schema ECOMMERCE_DEV to role udf_role;

grant select on all tables in schema ECOMMERCE_DEV to role udf_role;

```

```
grant all privileges on function sales_qty_by_supplier(date,number) to role udf_role;
```

```
grant role udf_role to user udf_developer;
```

--- By Default User get public role. For this section, we need to avoid that. ---

```
alter user udf_developer set default_role=udf_role;
```

```
4
5   create role udf_role;
6
```

Results		Chart	Query Details	...
	status		Query duration	74ms
1	Role UDF_ROLE successfully created.			

```
14
15   grant all privileges on function sales_qty_by_supplier(date,number) to role udf_role;
16
```

Results		Chart	Query Details	...
	status		Query duration	92ms
1	Statement executed successfully.			

Users Roles

2 Users

NAME ↑

MOHAMED

User Name

udf_developer

Email

SNOWFLAKE

New User

Creating as ACCOUNTADMIN

Password

.....

Confirm password

.....

Comment (optional)

UDF Developer

Force user to change password on first time login

Advanced User Options ^

Cancel

Create User

Search Owner All

OWNER

ACCOUNTADMIN

```

16
17 | grant role udf_role to user udf_developer;
18
19 --- By Default User get public role. For this section, we need to avoid that. ---
20 | alter user udf_developer set default_role=udf_role;
21

```

Results

status	
1	Statement executed successfully.

Query Details

Query duration 74ms

Results

status	
1	Statement executed successfully.

Query Details

Query duration 45ms

Log in to that user:

```
-- Test the GET_DDL using the new user account created above -----
use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";
select get_ddl('function','get_total_qty_shipped(date,number)');
select get_total_qty_shipped('1997-03-19',4072277) as total_qty_shipped;
```

The screenshot shows the AWS Cost Management console. The top navigation bar includes 'AI & MI', 'Mohamed Zuhair' (profile icon), and tabs for 'Account Overview', 'Consumption', and 'Resource Monitors'. A sidebar on the left lists 'Switch Role' (ACCOUNTADMIN), 'Account' (TXB80537), and links for 'My profile', 'Support', 'Appearance', 'Client download', 'Documentation', 'Privacy notice', and 'Sign Out'. The main content area displays 'Cost Management' for account TXB80537, which is associated with MOHAMEDZUHAIRKA - US West (Oregon). A modal window is open over the main content, showing 'Top warehouses by cost' with two entries: COMPUTE_WH and CLOUD_SERVICES_ONLY. The modal also displays the account identifier TXB80537, average daily cost (\$2.21), and a list of insights: 'Large tables that are never queried', 'Tables where data written but not read', and 'Rarely used materialized views'.

He can able to see the query for the UDF(taken screenshot from tutorial):

The screenshot shows a database interface with a query editor. The code entered is:

```
1 use schema "ECOMMERCE_DB"."ECOMMERCE_LIV";
2
3 select get_ddl('function','get_total_qty_shipped(date,number)');
```

A modal window titled "Details" is open, displaying the generated DDL for the function:

```
1 CREATE OR REPLACE FUNCTION "GET_TOTAL_QTY_SHIPPED"("SHIPDATE" DATE,
2 "SUPPLIER_KEY" NUMBER(38,0))
3 RETURNS NUMBER(38,0)
4 LANGUAGE SQL
5 AS '
6 SELECT
7     SUM(l_quantity) as total_quantity_shipped
8 FROM
9     LINEITEM
10 where
11     L_SHIPDATE =shipdate and l_suppkey=supplier_key
12 ';
```

At the bottom right of the modal is a "Copy" button. Below the modal, there is a "Done" button.

At the bottom of the interface, there are tabs for "Results" and "Data Preview". The status bar shows "92ms" and "1 rows".

To protect that, create UDF with secure keyword:

```
12 create or replace secure function get_total_qty_shipped(shipdate date,supplier_key number)
13 returns number
14 language sql
15 as
16 $$
17 SELECT
18     SUM(l_quantity) as total_quantity_shipped
19 FROM
20     LINEITEM
21 where
22     L_SHIPDATE =shipdate and l_suppkey=supplier_key
23 $$;
24
25 select get_total_qty_shipped('1997-03-19',4072277) as total_qty_shipped;
```

The screenshot shows a database interface with a query editor. The code entered is the same as above, but it is now a secure function:

```
12 create or replace secure function get_total_qty_shipped(shipdate date,supplier_key number)
13 returns number
14 language sql
15 as
16 $$
17 SELECT
18     SUM(l_quantity) as total_quantity_shipped
19 FROM
20     LINEITEM
21 where
22     L_SHIPDATE =shipdate and l_suppkey=supplier_key
23 $$;
24
25 select get_total_qty_shipped('1997-03-19',4072277) as total_qty_shipped;
```

The results pane shows a single row of data:

Row	status
1	Function GET_TOTAL_QTY_SHIPPED successfully created.

Now, if that user tries to get_ddl for that UDF, it will throw error.

```
3 select get_ddl('function', 'get_total_qty_shipped(date,number)');
4
5
6 [REDACTED]
```

Results Data Preview

✗ Query ID SQL 30ms

SQL compilation error: Object 'get_total_qty_shipped(date,number)' does not exist or not authorized.

But he can use the UDF to check the results:

```
4
5 select get_total_qty_shipped('1997-03-19',4872277) as total_qty_shipped;
```

results Data Preview Open History

✓ Query ID SQL 1.3s 1 rows

Filter result... Copy Columns ▾

Row	TOTAL_QTY_SHIPPED
1	32.00

External Function:

Data Format Received by Snowflake

When the remote service finishes processing a batch, the remote service should send data to Snowflake in a format similar to the format of the data sent by Snowflake. The returned value is in JSON format. Here is an example of the data section of such a response:

```
{  
    "data":  
        [  
            [  
                [ 0, 1995 ],  
                [ 1, 1974 ],  
                [ 2, 1983 ],  
                [ 3, 2001 ]  
            ]  
        ]  
}
```

Create lambda function:

Basic information

Function name
Enter a name that describes the purpose of your function.
 C

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
 ▼ C

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).
 Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
 ▼ C

View the [EMSDevIAM-AppApiLambdaRole-PT](#) role on the IAM console.

Copy and paste the below code into lambda function:

```

import json
import urllib3

def lambda_handler(event, context):

    status_code = 200
    array_of_rows_to_return = [ ]
    http = urllib3.PoolManager()

    try:
        event_body = event["body"]
        payload = json.loads(event_body)
        rows = payload["data"]

        for row in rows:

            row_number = row[0]
            from_currency = row[1]
            to_currency = row[2]

            response = http.request('GET','https://open.er-api.com/v6/latest/'+from_currency)
            response_data = response.data.decode('utf8').replace("", "")
            data = json.loads(response_data)

            exchange_rate_value = data['rates'][to_currency]

            output_value = [exchange_rate_value]
            row_to_return = [row_number, output_value]
            array_of_rows_to_return.append(row_to_return)

        json_compatible_string_to_return = json.dumps({"data" : array_of_rows_to_return})

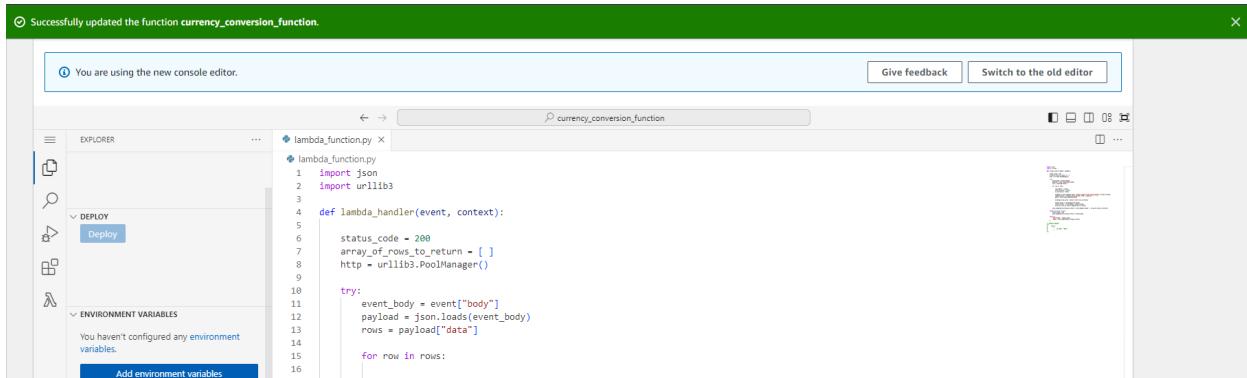
    except Exception as err:
        status_code = 400
        json_compatible_string_to_return = event_body

    return {
        'statusCode': status_code,
        'body': json_compatible_string_to_return
    }

# Example Request
# {
#   "data":
#   [
#     [0,"USD", "INR"]
#   ]
# }

```

Deploy the function:



```
lambda_function.py
1 import json
2 import urllib3
3
4 def lambda_handler(event, context):
5
6     status_code = 200
7     array_of_rows_to_return = []
8     http = urllib3.PoolManager()
9
10    try:
11        event_body = event["body"]
12        payload = json.loads(event_body)
13        rows = payload["data"]
14
15        for row in rows:
16            . . .
```

Create IAM role:

Trusted entity type

AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Leave the “Add permission” tab and click next:

<input type="checkbox"/>	<input checked="" type="checkbox"/> Amazon_EventBridge_Invoke_Ev...	Customer managed	-
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAPIGatewayAdministrat...	AWS managed	Provides full access to create/edit/delete...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAPIGatewayInvokeFul...	AWS managed	Provides full access to invoke APIs in A...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAPIGatewayPushToCl...	AWS managed	Allows API Gateway to push logs to us...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAppFlowFullAccess	AWS managed	Provides full access to Amazon AppFlo...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAppFlowReadOnlyAcc...	AWS managed	Provides read only access to Amazon A...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAppStreamFullAccess	AWS managed	Provides full access to Amazon AppStr...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAppStreamPCAcess	AWS managed	Amazon AppStream 2.0 access to AWS...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAppStreamReadOnly...	AWS managed	Provides read only access to Amazon A...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAppStreamServiceAcc...	AWS managed	Default policy for Amazon AppStream ...
<input type="checkbox"/>	<input checked="" type="checkbox"/> AmazonAthenaFullAccess	AWS managed	Provide full access to Amazon Athena ...

► Set permissions boundary - *optional*

[Cancel](#)

[Previous](#)

[Next](#)

Role details

Role name

Enter a meaningful name to identify this role.

currency_conversion_external_role

Maximum 64 characters. Use alphanumeric and '+=_,@-_` characters.

Description

Add a short explanation for this role.

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @~/[\{\}]!#\$%^*(;)";`"

Step 1: Select trusted entities

[Edit](#)

Trust policy

```

1 * [{ 
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "sts:AssumeRole"

```

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as

Step 3: Add tags

Add tags - optional [Info](#)

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create role](#)

Create API Gateway:

Choosing Your Endpoint Type: Regional Endpoint vs. Private Endpoint

You access a proxy service (such as Amazon API Gateway) via a URI, often referred to as an *endpoint*. The instructions for creating your Amazon API Gateway ask you to choose one of the following types of endpoints:

- A regional endpoint.
- A private endpoint.

The following information can help you choose the type of endpoint.

A regional endpoint can be accessed across AWS regions, or even across cloud platforms. Your Snowflake instance, your proxy service, and your remote service can all be in different regions or even on different cloud platforms. For example, a Snowflake instance running on Azure could send requests to an Amazon API Gateway regional endpoint, which in turn could forward data to a remote service running on GCP.

A private endpoint can be configured to allow access **only within a region**. For example, you can configure a private endpoint to allow access from only a Snowflake VPC (Virtual Private Cloud) in the same AWS region. Communication between a Snowflake VPC and a private endpoint uses AWS PrivateLink.

API Gateway X

APIs
Custom domain names
VPC links

WebSocket API

Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.

Works with the following:
Lambda, HTTP, AWS Services

Build

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

Import **Build** 

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

REST WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API Clone from existing API Import from Swagger or Open API 3 Example API

Settings

Choose a friendly name and description for your API.

API name*

Description

Endpoint Type  Regional 

* Required

Create API

Once create API gateway, create resource:

Create resource

Resource details

Proxy resource [Info](#)

Proxy resources handle requests to all sub-resources. To create a proxy resource use a path parameter that ends with a plus sign, for example {proxy+}.

Resource path

Resource name

CORS (Cross Origin Resource Sharing) [Info](#)

Create an OPTIONS method that allows all origins, all methods, and several common headers.

[Cancel](#)

[Create resource](#)

Resources

[API actions](#) [Deploy API](#)

[Create resource](#)

Resource details

[Delete](#)

[Update documentation](#)

[Enable CORS](#)

Path
/currency_conversion_api_resource

Resource ID
azh9sp

Methods (0)

[Delete](#)

[Create method](#)

Method type ▲ Integration type ▽ Authorization ▽ API key ▽

No methods

No methods defined.

Create method

Method details

Method type

POST

Integration type

Lambda function

Integrate your API with a Lambda function.



HTTP

Integrate with an existing HTTP endpoint.



Mock

Generate a response based on API Gateway mappings and transformations.



AWS service

Integrate with an AWS Service.



VPC link

Integrate with a resource that isn't accessible over the public internet.



Lambda proxy integration

Send the request to your Lambda function as a structured event.

Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-west-2

arn:aws:lambda:us-west-2:682470999581:function:curr

Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

yq5ervq4)

Deploy API

Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#)

Stage

New stage

Stage name

prod

Deployment description

A new stage will be created with the default settings. Edit your stage settings on the [Stage page](#).

Cancel **Deploy**

Method request settings

Successfully created deployment for test. This deployment is active for prod.

API Gateway > APIs > Resources - test (3uyq5ervq4)

Resources

Resource details		API actions	Deploy API
Path	/	Resource ID	akla3m38g5
Methods (1)	POST	Delete	Create method

Test API Gateway:

The screenshot shows the AWS Lambda Test API Gateway interface. On the left, a sidebar lists resources: Create resource, / (GET), /currency_conversion_api_resource (POST), and /event_license_response. The main area is titled 'Test method' with the sub-section 'Query strings'. It contains the URL parameter 'param1=value1¶m2=value2'. Below it is the 'Headers' section with 'header1:value1' and 'header2:value2'. The 'Client certificate' section indicates 'No client certificates have been generated.' The 'Request body' section displays a JSON object with a single key 'data' containing an array with one element [0, "USD", "INR"].

As we expect we get the response,

The screenshot shows the AWS Lambda Test API Gateway interface after a POST request. The sidebar remains the same. The main area has a 'Test' button highlighted in orange. Below it, the results for the POST method are shown: Request path '/currency_conversion_api_resource', Response body '{"data": [[0, [84.106102]]]}', Response headers including 'X-Amzn-Trace-Id: Root=1-670ac125-e6508f74d287889aebc396ec;Parent=278fc037772766c4;Sampled=0;Lineage=1:ded90721:0', and Status 200.

Secure API Gateway:

Check the Authorization is chosen as “AWS IAM”:

The screenshot shows the AWS Lambda Method request settings. The sidebar includes / (GET), /currency_conversion_api_resource (POST), and /event_license_response. The main area is titled 'Method request settings' and shows the 'Authorization' field set to 'AWS_IAM'. Other settings include 'Request validator' as 'None', 'API key required' as 'False', and 'SDK operation name' as 'Generated based on method and path'.

Then in “Resource Policy” tab, copy and paste below code and update it:

```
{  
    "Version": "2012-10-17",  
    "Statement":  
    [  
        {  
            "Effect": "Allow",  
            "Principal":  
                {  
                    "AWS":  
                        "arn:aws:sts::<12-digit-number>:assumed-role/<external_function_role>/snowflake"  
                },  
            "Action": "execute-api:Invoke",  
            "Resource": "<method_request_ARN>"  
        }  
    ]  
}
```

Edit resource policy [Info](#)

Amazon API Gateway resource policies are JSON policy documents that you attach to an API to control whether a specified principal (typically an IAM role or group) can invoke the API.

Policy details

Custom ▾

Policy examples

```
1▼ {  
2    "Version": "2012-10-17",  
3    "Statement":  
4▼ [  
5    {  
6        "Effect": "Allow",  
7        "Principal":  
8            {  
9                "AWS": "arn:aws:sts::██████████:assumed-role/currency_conversion_external_role/snowflake"  
10            },  
11        "Action": "execute-api:Invoke",  
12        "Resource": "arn:aws:execute-api:us-west-2:██████████:3uyq5ervq4/"  
13    }  
14]  
15 }
```

Cancel Save changes

Successfully updated resource policy for 'test'. Redeploy your API for the update to take effect.

API Gateway > APIs > test (3uyq5ervq4) > Resource policy

Resource policy Info

Use resource policies to configure access control to this API. You must redeploy your API for changes to this policy to take effect.

Policy details	
<pre>1 { 2 "Version": "2012-10-17", 3 "Statement": [4 { 5 "Effect": "Allow", 6 "Principal": { 7 "AWS": "arn:aws:sts::██:assumed-role/aws-saml-bsys-ems-dev-developer/snowflake" 8 }, 9 "Action": "execute-api:Invoke", 10 "Resource": "arn:aws:execute-api:us-west-2:██:3uyq5ervq4/*" 11 } 12] 13 }</pre>	<small>Edit</small>

As said, re-deploy the API.

Snowflake external function:

```
use role accountadmin;

use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";

create or replace api integration currency_conversion_int
    api_provider = aws_api_gateway
    api_aws_role_arn =
    api_allowed_prefixes = ()
    enabled = true;

desc integration currency_conversion_int;

create or replace external function currency_conversion_external_function(from_currency
    varchar,to_currency varchar)
    returns variant
    api_integration = external_function_integration
    as ";

select currency_conversion_external_function('USD','EUR')

select currency_conversion_external_function('USD','EUR')[0] as exchange_value;
```

```

5   create or replace api integration currency_conversion_int
6     api_provider = aws_api_gateway
7     api_aws_role_arn = 'arn:aws:iam::█████████████████████:role/████████████████████'
8     api_allowed_prefixes = ('https://3uyq5ervq4.execute-api.us-west-
9       2.amazonaws.com/prod//currency_conversion_api_resource')
10    enabled = true;
11
12  desc integration currency_conversion_int;
13

```

Results

	status
1	Integration CURRENCY_CONVERSION_INT successfully created.

Query Details ...
Query duration 265ms


```

11
12  desc integration currency_conversion_int;
13

```

Results

	property	property_type	property_value	property_defa
3	API_PROVIDER	String	AWS_API_GATEWAY	
4	API_AWS_IAM_USER_ARN	String	arn:aws:iam::180294178125:user/████████████████████	
5	API_AWS_ROLE_ARN	String	arn:aws:iam::682470999581:role/a	

Query Details ...
Query duration 266ms
Rows Ask Copilot

Then edit trust policy in AWS IAM role:

IAM > Roles > currency_conversion_ext_role > Edit trust policy

Edit trust policy

```

1▼ {
2  "Version": "2012-10-17",
3▼   "Statement": [
4▼     {
5       "Effect": "Allow",
6▼         "Principal": {
7           "AWS": "arn:aws:iam::████████████████████:user/hndy-s-eusb7117"
8         },
9         "Action": "sts:AssumeRole",
10        "Condition": {
11          "StringEquals": {
12            "sts:ExternalId": "AZ32171_SFCRole=3_q0KsT0zNTUxGf7/wMLy45r/Imt8="
13          }
14        }
15      }
16    ]
17 }

```

```
--  
14  create or replace external function currency_conversion_external_function(from_currency  
15    varchar,to_currency varchar)  
16      returns variant  
17      api_integration = currency_conversion_int  
18      as 'https://3uyq5ervq4.execute-api.us-west-2.amazonaws.com/prod//currency_conversion_api_resource';
```

↳ Results ↗ Chart

	status	Query Details	...
1	Function CURRENCY_CONVERSION_EXTERNAL_FUNCTION successfully created.	Query duration	117ms

```
18  
19 select currency_conversion_external_function('USD','EUR');  
20  
21 select currency_conversion_external_function('USD','EUR')[0] as exchange_value;
```

Results Data Preview

✓ Query ID SQL 1.16s 1 rows

Filter result...

Row	CURRENCY_CONVERSION_EXTERNAL_FUNCTION('USD','EUR')
1	[0.913]

```
21 select currency_conversion_external_function('USD','EUR')[0] as exchange_value;  
22  
23
```

Results Data Preview

✓ Query ID SQL 979ms 1 rows

Filter result...

Row	EXCHANGE_VALUE
1	0.913

Snowflake with Python, Pyspark and Airflow:

Connect python with Snowflake locally:

```
(pyenv) C:\Development\Python\pyenv\Scripts>pip install snowflake-connector-python
Collecting snowflake-connector-python
  Downloading snowflake_connector_python-3.12.2-cp39-cp39-win_amd64.whl (916 kB)
    |████████| 916 kB 1.7 MB/s
Requirement already satisfied: typing-extensions<5,>=4.3 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (4.9.0)
Requirement already satisfied: certifi>2017.4.17 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (2022.9.24)
Requirement already satisfied: urllib3<2.0.0,>=1.21.1; python_version < "3.10" in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (1.27.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (3.3.2)
Collecting filelock<4,>=3.5
  Downloading filelock-3.16.1-py3-none-any.whl (16 kB)
Collecting tomllib
  Downloading tomllib-0.13.2-py3-none-any.whl (37 kB)
Requirement already satisfied: pytz in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (2022.5)
Requirement already satisfied: sortedcontainers>=2.4.0 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (2.4.0)
Collecting asn1crypto<2.0.0,>=2.4.0
  Downloading asn1crypto-1.5.1-py2.py3-none-any.whl (105 kB)
    |████████| 105 kB 3.2 MB/s
Requirement already satisfied: cffi<2.0.0,>=1.9 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (1.15.1)
Requirement already satisfied: cryptography>=3.1.0 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (38.0.1)
Requirement already satisfied: pyjwt<3.0.0 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (2.8.0)
Requirement already satisfied: packaging in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (21.3)
Requirement already satisfied: idna<4,>=2.5 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (2.10)
Requirement already satisfied: requests<3.0.0 in c:\development\python\pyenv\lib\site-packages (from snowflake-connector-python) (2.31.0)
Collecting platformdirs<5.0.0,>=2.6.0
  Downloading platformdirs-4.3.6-py3-none-any.whl (18 kB)
Collecting pyOpenSSL<25.0.0,>=16.2.0
  Downloading pyOpenSSL-24.2.1-py3-none-any.whl (58 kB)
    |████████| 58 kB 1.6 MB/s
Requirement already satisfied: pycparser in c:\development\python\pyenv\lib\site-packages (from cffi<2.0.0,>=1.9->snowflake-connector-python) (2.21)
Requirement already satisfied: pyParsing!=3.0.5,>=2.0.2 in c:\development\python\pyenv\lib\site-packages (from packaging->snowflake-connector-python) (3.0.9)
Installing collected packages: filelock, tomllib, asn1crypto, platformdirs, pyOpenSSL, snowflake-connector-python
ERROR: After October 2020 you may experience errors when installing or updating packages. This is because pip will change the way that it resolves dependency conflicts.
We recommend you use --use-feature=2020-resolver to test your packages with the new resolver before it becomes the default.

pyopenssl 24.2.1 requires cryptography<44,>=41.0.5, but you'll have cryptography 38.0.1 which is incompatible.
Successfully installed asn1crypto-1.5.1 filelock-3.16.1 platformdirs-4.3.6 pyOpenSSL-24.2.1 snowflake-connector-python-3.12.2 tomllib-0.13.2
WARNING: You are using pip version 20.2.3; however, version 24.2 is available.
```

Run the below program in local machine:

```
import snowflake.connector

ctx = snowflake.connector.connect(
    user="",
    password="",
    account="", #Provide Locator ID
    warehouse="compute_wh",
    database="ecommerce_db",
    schema="ECOMMERCE_DEV",
    role='SYSADMIN',
    session_parameters={
        'TIMEZONE': 'UTC',
    }
)

cs = ctx.cursor()
try:
    sql = """select * from LINEITEM limit 10"""
    cs.execute(sql)
    # one_row = cs.fetchone()
    all_rows = cs.fetchall()
    # print(one_row[0])
    print(all_rows)

finally:
```

```
cs.close()
ctx.close()
```

```
(pyenv) C:\Development\Python\UtilityPrograms\Snowflake>py python snowflake_local.py
[(5698238502, 173336436, 8336471, 4, Decimal('44.00'), Decimal('54405.88'), Decimal('0.08'), Decimal('0.06'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1995, 11, 2
3), datetime.date(1996, 2, 27), 'DELIVER IN PERSON', 'FOB', 'l dependencies. deposits haggle across', (5698474147, 20961814, 3461817, 2, Decimal('41.00'), Decimal('76865.5
7'), Decimal('0.00'), Decimal('0.05'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1996, 1, 6), datetime.date(1996, 2, 23), 'DELIVER IN PERSON', 'MAIL', 'ly. ironic
deposits by the fluffy idle', (5698224512, 12733887, 5233889, 2, Decimal('34.00'), Decimal('65288.50'), Decimal('0.08'), Decimal('0.07'), 'N', 'O', datetime.date(1996,
2, 6), datetime.date(1995, 12, 8), datetime.date(1996, 3, 5), 'COLLECT COD', 'SHIP', 'e carefully pending accounts boo', (5698388194, 126980983, 1981008, 3, Decimal('34.00
'), Decimal('69959.76'), Decimal('0.05'), Decimal('0.03'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1996, 3, 27), datetime.date(1996, 2, 18), 'COLLECT COD', 'SHIP'
, 'promise blithely unusual sen', (5698087040, 174543546, 9543581, 1, Decimal('6.00'), Decimal('9484.92'), Decimal('0.00'), Decimal('0.01'), 'N', 'O', datetime.date(1996,
2, 6), datetime.date(1996, 2, 27), datetime.date(1996, 2, 15), 'NONE', 'SHIP', 'bold, ironi', (5698072965, 13514687, 1014647, 2, Decimal('21.00'), Decimal('33913.53'), D
ecimal('0.00'), Decimal('0.02'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1996, 1, 21), datetime.date(1996, 2, 11), 'TAKE BACK RETURN', 'MAIL', 'wickly special re
qu', (5861758402, 159854621, 2354637, 5, Decimal('49.00'), Decimal('76813.87'), Decimal('0.00'), Decimal('0.07'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1996,
3, 16), datetime.date(1996, 2, 24), 'NONE', 'FOB', 'y slyly bold platelets. regular requests wa', (569838791, 104054815, 6554826, 1, Decimal('6.00'), Decimal('10587.66
'), Decimal('0.09'), Decimal('0.00'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1996, 3, 3), datetime.date(1996, 2, 11), 'DELIVER IN PERSON', 'TRUCK', 'the carefully
bold requests. blithe', (5698370563, 128183562, 3183587, 4, Decimal('2.00'), Decimal('3278.32'), Decimal('0.00'), Decimal('0.08'), 'N', 'O', datetime.date(1996, 2, 6), dat
etime.date(1995, 12, 29), datetime.date(1996, 3, 2), 'COLLECT COD', 'REG AIR', 'kly final packages haggle slyly ev', (5698146913, 86515792, 9015801, 3, Decimal('40.00'), D
ecimal('72138.80'), Decimal('0.09'), Decimal('0.08'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1996, 2, 16), datetime.date(1996, 2, 19), 'NONE', 'TRUCK', 't the f
uriously final plat')]
```

Run python scripts in Glue:

Download whl file from the github repo:

https://github.com/sidd88/snowflake-aws-udemy/blob/main/Section%2010-AWS-Python-SF/wl/snowflake_connector_python-2.3.8-py3-none-any.whl

Add the file in S3:

The screenshot shows the AWS S3 console interface. The path 'Amazon S3 > Buckets > ems-migration-bucket-us-west-2-qa > test_pt_ > snowflake/' is visible. In the 'Objects' tab, there is one object listed: 'snowflake_connector_python-2.3.8-py3-none-any.whl'. The file is a 'whl' file, last modified on October 14, 2024, at 20:43:15 (UTC+05:30), and has a size of 332.6 KB.

Create Glue Job

Copy the below code in ETL job:

```
import snowflake.connector
from awsglue.utils import getResolvedOptions
```

```
con = snowflake.connector.connect(  
    user="siddharth",  
    password="Udemy@123",  
    account="az32171.eu-central-1",  
    warehouse="compute_wh",  
    database="ecommerce_db",  
    schema="ECOMMERCE_DEV",  
    role='SYSADMIN',  
    session_parameters={  
        'TIMEZONE': 'UTC',  
    }  
)  
  
cursor = con.cursor()  
  
sql_query = """  
    select * from lineitem limit 100  
"""  
  
try :  
    cursor.execute(sql_query)  
    query_id = cursor.sfqid  
    cursor.get_results_from_sfqid(query_id)  
    results = cursor.fetchall()  
    print(f'{results[0]}'  
  
except Exception as e:  
    con.rollback()  
    raise e  
  
finally:  
    con.close()
```

snowflake_glue

Script **Job details** Runs Data quality Schedules Version Control

Name
snowflake_glue

Description - *optional*
Glue job to execute snowflake queries

Descriptions can be up to 2048 characters long.

IAM Role
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.
ems-migration-GlueServiceRole-test

Type
The type of ETL job. This is set automatically based on the types of data sources you have selected.
Python Shell

Python version
Python 3.9

Load common analytics libraries (recommended)
Include common Python libraries from [pypi.org](#) such as pandas, numpy and s3fs. Uncheck if you are loading your own libraries, and wish to avoid version conflicts.

Libraries [Info](#)
Python library path
s3://ems-migration-bucket-us-west-2-qa/test_pt/snowflake/snowflake_connector_python-2.3.8-py3-none-any.whl

Referenced files path

Once done, save the job and run the job. Once job completed, we can able to see the output in logs.

snowflake_glue

Last modified on 10/14/2024, 8:48:05 PM Actions ▾

Script | Job details | **Runs** | Data quality | Schedules | Version Control

Job runs (1/3) Info Last updated (UTC) October 14, 2024 at 15:19:58 View details Stop job run Table View

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Worker type	Glue vers
Succeeded	0	10/14/2024 20:48:08	10/14/2024 20:49:19	1 m 2 s	0.0625 DPUs	-	3.0
Failed	0	10/14/2024 20:46:32	10/14/2024 20:47:07	31 s	0.0625 DPUs	-	3.0

Run details Input arguments (7) Continuous logs Run insights Metrics Spark UI

Job name snowflake_glue Start time (Local) 10/14/2024 20:48:08 Glue version 3.0 Last modified on (Local) 10/14/2024 20:49:19
 Id jid_4c392194f7586ade315414fea80b2dd3620965244145bce8a43581 End time (Local) Log group name /aws-glue/python-jobs
 62f4133cc1 Run status Succeeded Start-up time 9 seconds Max capacity 0.0625 DPUs Number of workers -
 Retry attempt number 0 Execution time Standard Timeout Initial run 1 minute 2 seconds Cloudwatch logs 20 minutes Trigger name Security configuration Usage profile -
 - -
 Job run queuing False Output logs Error logs

```

▶ 2024-10-14T15:18:54.582Z Installing collected packages: pytz, pyjwt, chardet, certifi, azure-common, asn1crypto, urllib3, typing-extensions, six, setuptools, pycryptodomex, pyparser, oscrypto
▶ 2024-10-14T15:18:54.017Z Successfully installed asn1crypto-1.5.1 azure-common-1.1.28 azure-core-1.31.0 azure-storage-blob-12.23.1 boto3-1.35.39 botocore-1.35.39 certifi-2020.12.5 cffi-1.17.1
▶ 2024-10-14T15:18:54.170Z [notice] A new release of pip available: 22.1.2 -> 24.2 [notice] To update, run: pip install --upgrade pip
▶ 2024-10-14T15:18:54.636Z Setup complete. Starting script execution: ----
▼ 2024-10-14T15:19:08.500Z (5698238502, 173336436, 8336471, 4, Decimal('44.00'), Decimal('64405.88'), Decimal('0.08'), Decimal('0.06'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1995, 12, 23), datetime.date(1996, 2, 27), 'DELIVER IN PERSON', 'FOB', '1 dependencies. deposits haggle across')
  (5698238502, 173336436, 8336471, 4, Decimal('44.00'), Decimal('64405.88'), Decimal('0.08'), Decimal('0.06'), 'N', 'O', datetime.date(1996, 2, 6), datetime.date(1995, 11, 23), datetime.date(1996, 2, 27), 'DELIVER IN PERSON', 'FOB', '1 dependencies. deposits haggle across')

```

Run job with params:

Copy below code and update the credentials to connect to snowflake:

```

import sys
import snowflake.connector
from awsglue.utils import getResolvedOptions
from datetime import datetime,timedelta

args = getResolvedOptions(sys.argv, ['supplier_key','ship_date'])

supplier_key = args['supplier_key']
ship_date = args['ship_date']

con = snowflake.connector.connect(
    user="",
    password="",
    account="",
    warehouse="compute_wh",
    database="ecommerce_db",
    schema="ECOMMERCE_DEV",
    role='SYSADMIN',
    session_parameters={
        'TIMEZONE': 'UTC',
    }
)

```

```

)
cursor = con.cursor()

sql_query = """
    select
    *
    from lineitem
    where l_shipdate='{0}' and l_suppkey='{1}'
""".format(ship_date,supplier_key)

try :
    cursor.execute(sql_query)
    query_id = cursor.sfqid
    cursor.get_results_from_sfqid(query_id)
    results = cursor.fetchall()
    print(f'{results[0]}')

except Exception as e:
    con.rollback()
    raise e

finally:
    con.close()

```

⌚ Successfully updated job
Successfully updated job snowflake_glue. To run the job choose the Run Job button.

snowflake_glue

[Script](#) | [Job details](#) | [Runs](#) | [Data quality](#) | [Schedules](#) | [Version Control](#)

[Script](#) [Info](#)

```

1 import sys
2 import snowflake.connector
3 from awsglue.utils import getResolvedOptions
4 from datetime import datetime,timedelta
5
6 args = getResolvedOptions(sys.argv, ['supplier_key','ship_date'])
7
8 supplier_key = args['supplier_key']
9 ship_date = args['ship_date']
10
11 con = snowflake.connector.connect(
12     user='[REDACTED]',#
13     password='[REDACTED]',#
14     account='[REDACTED]', #Provide Locator Id
15     warehouse="compute_WH",
16     database="ECOMMERCE_DB",
17     schema="ECOMMERCE_DEV",
18     role='ACCOUNTADMIN',
19     session_parameters={
20         'TIMEZONE': 'UTC',
21     }
22 )
23
24 cursor = con.cursor()

```

Now pass the params and run the job:

Run parameters (optional)

Review and override parameter values, as needed, before running this job. Changes affect this run only. Edit a job to change default parameter values.

⚠ Only job snowflake_glue is run. Jobs dependent on the completion of job snowflake_glue will not be run. To run a job and trigger dependent jobs, define an on-demand trigger.

- ▶ Advanced properties
- ▶ Monitoring options
- ▶ Security configuration
- ▶ Script libraries
- ▼ Job parameters

Key	Value	Remove
--supplier_key	2764188	<button>Remove</button>
--ship_date	1995-09-07	<button>Remove</button>
Add new parameter		

[Cancel](#)
Run job

snowflake_glue

Last modified: [REDACTED]

Script	Job details	Runs	Data quality	Schedules	Version Control																		
Job runs (1/5) Info <div style="display: flex; justify-content: space-between; align-items: center;"> Last updated (UTC) October 14, 2024 at 15:34:32 G View details </div> <table border="1"> <thead> <tr> <th>Run status</th> <th>Retries</th> <th>Start time (Local)</th> <th>End time (Local)</th> <th>Duration</th> <th>Capacity (DPUs)</th> </tr> </thead> <tbody> <tr> <td>Succeeded</td> <td>0</td> <td>10/14/2024 21:02:04</td> <td>10/14/2024 21:03:07</td> <td>53 s</td> <td>0.0625 DPUs</td> </tr> <tr> <td>Failed</td> <td>0</td> <td>10/14/2024 20:57:47</td> <td>10/14/2024 20:58:53</td> <td>56 s</td> <td>0.0625 DPUs</td> </tr> </tbody> </table> <div style="margin-top: 10px;"> <pre> ▶ 2024-10-14T15:32:46.245Z [notice] A new release of pip available: 22.1.2 -> 24.2 [notice] To update, run: pip install --upgrade pip ▶ 2024-10-14T15:32:46.749Z Setup complete. Starting script execution: ---- ▼ 2024-10-14T15:32:56.846Z (3316514693, 147549268, 2549297, 4, Decimal('19.00'), Decimal('24887.91'), Decimal('0.04'), Decimal('0.05'), 'A', 'F', datetime.date(1992, 2, 28), datetime.date(1992, 3, 18), 'DELIVER IN PERSON', 'TRUCK', ' deposits alongside of the deposits cajole') </pre> </div>						Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)	Succeeded	0	10/14/2024 21:02:04	10/14/2024 21:03:07	53 s	0.0625 DPUs	Failed	0	10/14/2024 20:57:47	10/14/2024 20:58:53	56 s	0.0625 DPUs
Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)																		
Succeeded	0	10/14/2024 21:02:04	10/14/2024 21:03:07	53 s	0.0625 DPUs																		
Failed	0	10/14/2024 20:57:47	10/14/2024 20:58:53	56 s	0.0625 DPUs																		

Run snowflake query with pandas:

Copy below code and fill the connection configuration. Run the job.

```
import pandas as pd
import sys
import snowflake.connector
# from awsglue.utils import getResolvedOptions

conn = snowflake.connector.connect(
    user="",
    password="",
    account="",
    warehouse="compute_wh",
    database="ecommerce_db",
    schema="ECOMMERCE_DEV",
    role='SYSADMIN',
    session_parameters={
        'TIMEZONE': 'UTC',
    }
)

try:
    sql = """
        select * from lineitem limit 10
    """
    data_agg = pd.read_sql(sql, conn)
    print(data_agg.head())
finally:
    conn.close()
```

snowflake_glue

Script Job details Runs Data quality Schedules Version Control

Script Info

```
1 import pandas as pd
2 import sys
3 import snowflake.connector
4 from awsglue.utils import getResolvedOptions
5
6 con = snowflake.connector.connect(
7     user='[REDACTED]',
8     password='[REDACTED]',
9     account='[REDACTED]', #Provide Locator Id
10    warehouse="compute_wh",
11    database="ECOMMERCE_DB",
12    schema="ECOMMERCE_DEV",
13    role='ACCOUNTADMIN',
14    session_parameters={
15        'TIMEZONE': 'UTC',
16    }
17)
18
19 try:
20     sql = """
21         select * from lineitem limit 10
22     """
23     data_agg = pd.read_sql(sql, con)
24     print(data_agg.head())
25 finally:
26     con.close()
```

snowflake_glue

Last updated (UTC) October 14, 2024 at 15:41:20 [View details](#)

Job runs (1/8) Info

Filter job runs by property

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPU)
● Succeeded	0	10/14/2024 21:10:37	10/14/2024 21:11:16	33 s	0.0625 DPU
○ Failed	0	10/14/2024 21:09:07	10/14/2024 21:10:06	51 s	0.0625 DPU

Run details Input arguments (7) Continuous logs Run insights Metrics Spark UI

Job name Start time (Local) Glue version
snowflake_glue 10/14/2024 21:10:37 3.0
Id End time (Local) Worker type
jr_0e73576a3f9cd04a1667c84d1a97925a6a1f08cc53261ef204755a7 10/14/2024 21:11:16 -
37aa4bb17 Run status Start-up time Max capacity
● Succeeded 5 seconds 0.0625 DPU
Retry attempt number Execution time Execution class
Initial run 33 seconds Standard
Trigger name Security configuration Cloudwatch logs
- -

- Output logs
- Error logs

Job run queuing
False

```

▶ 2024-10-14T15:41:01.309Z [notice] A new release of pip available: 22.1.2 => 24.2 [notice] To update, run: pip install --upgrade pip
▶ 2024-10-14T15:41:01.998Z Setup complete. Starting script execution: ----
▼ 2024-10-14T15:41:05.312Z L_ORDERKEY L_PARTKEY ... L_SHIPMODE L_COMMENT 0 5698238502 173336436 ... FOB 1 dependencies. deposits haggle across 1 5698474147 20961814 ... MAIL ly. ironic d
L_ORDERKEY L_PARTKEY ... L_SHIPMODE L_COMMENT
0 5698238502 173336436 ... FOB 1 dependencies. deposits haggle across
1 5698474147 20961814 ... MAIL ly. ironic deposits by the fluffly idle
2 5698224512 12733887 ... SHIP e carefully pending accounts boo
3 5698388194 126980983 ... SHIP promise blithely unusual sen
4 5698087040 174543546 ... SHIP bold, ironi
[5 rows x 16 columns]

```

Run pyspark in Glue:

Download the jar files from github repo:

<https://github.com/siddd88/snowflake-aws-udemy/blob/main/Section%2010-AWS-Python-SF/spark-jars/Jar-Archive.zip>

Upload in s3 bucket:

The screenshot shows the AWS S3 console interface. The path is: Amazon S3 > Buckets > ems-migration-bucket-us-west-2-qa > test_pt > snow_spark/. The 'Objects' tab is selected. There are two objects listed:

Name	Type	Last modified	Size
snowflake-jdbc-3.13.15.jar	jar	October 14, 2024, 21:19:58 (UTC+05:30)	
spark-snowflake_2.12-2.9.2-spark_3.1.jar	jar	October 14, 2024, 21:20:00 (UTC+05:30)	

Create spark job:

snowflake_spark

Script **Job details** Runs Data quality Schedules Version Control

Basic properties Info

Name
snowflake_spark

Description - optional
Snowflake in spark

Descriptions can be up to 2048 characters long.

IAM Role
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.

EMSDevIAM-emsMigrationGlueRole-0Szg9xWktOq2

Type
The type of ETL job. This is set automatically based on the types of data sources you have selected.

Spark

Glue version Info
Glue 3.0 - Supports spark 3.1, Scala 2, Python 3

Language
Python 3

Worker type
Set the type of predefined worker that is allowed when a job runs.

G 1X
(4vCPU and 16GB RAM)

Libraries Info

Python library path

s3://ems-migration-bucket-us-west-2-qa/test_pt/snow_spark/snowflake-jdbc-3.13.15.jar,s3://ems-migration-bucket

Dependent JARs path

Referenced files path

Copy the below code and paste it. Run the job.

```
from pyspark.sql import SparkSession
```

```
from pyspark import SparkContext

spark = SparkSession \
    .builder \
    .appName("Glue-pyspark-test") \
    .getOrCreate()

SNOWFLAKE_SOURCE_NAME = "net.snowflake.spark.snowflake"
snowflake_database="ECOMMERCE_DB"
snowflake_schema="ECOMMERCE_DEV"
source_table_name="LINEITEM"

snowflake_options = {
    "sfUrl": "",
    "sfUser": "",
    "sfPassword": "",
    "sfDatabase": snowflake_database,
    "sfSchema": snowflake_schema,
    "sfWarehouse": "COMPUTE_WH"
}

df = spark.read \
    .format(SNOWFLAKE_SOURCE_NAME) \
    .options(**snowflake_options) \
    .option("dbtable",source_table_name) \
    .option("autopushdown", "on") \
    .load()

df.write.format("snowflake") \
    .options(**snowflake_options) \
    .option("dbtable","spark_output_table").mode("overwrite") \
    .save()

#
s3://sf-glue-jobs/jars/snowflake-jdbc-3.13.15.jar,s3://sf-glue-jobs/jars/spark-snowflake_2.12-2.9.2-spark_3.1.jar
```

After execution, we can see a new table in snowflake.

snowflake_spark

Last modified: October 14, 2024 at 16:05:07

Script | Job details | **Runs** | Data quality | Schedules | Version Control

Job runs (1/3) [Info](#)

Filter job runs by property

Run status	Retries	Start time (Local)	End time (Local)	Duration	Capacity (DPUs)
Succeeded	0	10/14/2024 21:33:24	10/14/2024 21:35:09	1 m 41 s	2 DPUs
Failed	0	10/14/2024 21:31:15	10/14/2024 21:32:50	47 s	2 DPUs

Run details | Input arguments (8) | Continuous logs | Run insights | Metrics | Spark UI

Job name: snowflake_spark

Start time (Local): 10/14/2024 21:33:24

Glue version: 3.0

Id: jr_8e5e8e7c133e5a47115014ff05f35fa812c7bbf7a4aea8942fcba521

End time (Local): 10/14/2024 21:35:09

Worker type: G.1X

Run status: Succeeded

Start-up time: 3 seconds

Max capacity: 2 DPUs

Retry attempt number: 0

Execution time: 1 minute 41 seconds

Execution class: Standard

Initial run: Standard

Trigger name: Cloudwatch logs

-: Security configuration

- Output logs
- Error logs

Job run queuing: False

Load sample data with SQ...

2024-10-05 3:20pm

2024-10-11 10:11pm

2024-10-12 11:25pm

Databases Worksheets

Search objects

- LINEITEM
- LINEITEM_CLONE
- ORDERS
- SPARK_OUTPUT_TABLE**

ECOMMERCE_DB.ECOMMERCE_DEV Settings

```

1 use role accountadmin;
2
3 use schema "ECOMMERCE_DB"."ECOMMERCE_DEV";
4
5 select * from lineitem;

```

Results Chart

	L_ORDERKEY	L_PARTKEY
1	3316514693	147549268
2	1746217926	166973548
3	1746202240	140408439
4	2460173152	158943595
5	2460109959	149453493
6	2460135297	17591932

SPARK_OUTPUT_TABLE 2M Rows

#	L_ORDERKEY	NUMBER(38,0)
#	L_PARTKEY	NUMBER(38,0)
#	L_SUPPKEY	NUMBER(38,0)
#	L_LINENUMBER	NUMBER(38,0)
#	L_QUANTITY	NUMBER(12,2)
#	L_EXTENDEDPRICE	NUMBER(12,2)
#	L_DISCOUNT	NUMBER(12,2)
#	L_TAX	NUMBER(12,2)
A	L_RETURNFLAG	VARCHAR(16777216)
A	L_LINESTATUS	VARCHAR(16777216)

Managed Airflow cluster on AWS:

Amazon MWAA > Environments > Create environment

Specify details

Step 1
Specify details

Step 2
Configure advanced settings

Step 3
Review and create

▼ How Amazon MWAA works



Create an environment
An environment contains your Airflow cluster, including your scheduler, workers, and web server.

Upload your DAGs to Amazon S3
Package and upload your DAG (Directed Acyclic Graph) code to Amazon S3. Amazon MWAA loads the code into Airflow.

Run your DAGs in Airflow
Run your DAGs from the Airflow UI or CLI. Monitor your environment with Amazon CloudWatch.

New to MWAA? [Read the overview](#)

Environment details [Info](#)

Name: Use only letters, numbers, dashes, or underscores. Max 80 characters.

Airflow version:

Weekly maintenance window start (UTC):

Create new s3 bucket for airflow dags and put requirement.txt in root folder:

Amazon S3 > sid-sf-airflow

sid-sf-airflow [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#)

[Upload](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	dags/	Folder	-	-	-
<input type="checkbox"/>	requirements.txt	txt	March 14, 2022, 10:58:26 (UTC+04:00)	55.0 B	Standard

Requirement.txt

```
snowflake-connector-python==2.5.1
```

ⓘ Create or specify an S3 bucket to store your DAG code. The bucket name must have versioning enabled. You can create a new bucket in the [Amazon S3 console](#)

S3 Bucket

The S3 bucket where your source code is stored. Enter an S3 URI or browse and select a bucket.

[View](#)[Browse S3](#)

Format: s3://mybucketname

DAGs folder

The S3 bucket folder that contains your DAG code. Enter an S3 URI or browse and select a folder.

[View](#)[Browse S3](#)

Format: s3://mybucketname/mydagfolder

Plugins file - *optional*

The S3 bucket ZIP file that contains your DAG plugins. Enter an S3 URI or browse and select a file object and version.

[Choose a version](#)[View](#)[Browse S3](#)

Format: s3://mybucketname/myplugins.zip

Requirements file - *optional*

The S3 bucket file that contains your DAG requirements.txt. Enter an S3 URI or browse and select a file object and version.

[Choose a version](#)[View](#)[Browse S3](#)

Create MWAA VPC and provide that:

Configure advanced settings

Networking [Info](#)

Virtual private cloud (VPC)

Defines the networking infrastructure setup of your Airflow environment. An environment needs 2 private subnets in different availability zones. To create a new VPC with private subnets, choose Create MWAA VPC. [Learn more](#)

[Create MWAA VPC](#)

Subnet 1

Private subnet for the first availability zone. Each environment occupies 2 availability zones.



eu-central-1a

Private

Subnet 2

Private subnet for the second availability zone. Each environment occupies 2 availability zones.



eu-central-1b

Private

Environment class Info

Each Amazon MWAA environment includes the scheduler, web server, and 1 worker. Workers auto-scale up and down according to system load. You can monitor the load on your environment and modify its class at any time.

DAG capacity*	Scheduler CPU	Worker CPU	Web server CPU
<input checked="" type="radio"/> mw1.small Up to 50	1 vCPU	1 vCPU	0.5 vCPU
<input type="radio"/> mw1.medium Up to 250	2 vCPU	2 vCPU	1 vCPU
<input type="radio"/> mw1.large Up to 1000	4 vCPU	4 vCPU	2 vCPU

*under typical usage

Maximum worker count

The maximum number of workers your environment is permitted to scale up to.

Must be between 1 and 25

Minimum worker count

The minimum number of workers always present in your environment.

Must be less than or equal to maximum workers. Minimum 1 worker

Amazon MWAA > Environments

Airflow environments

Next steps		Actions													
<p>While you wait for your environment to be created, learn about how to access the Airflow UI and work with your DAGs.</p>		Learn more													
<hr/>															
<h3>Environments (2)</h3>															
<table><thead><tr><th>Name</th><th>Status</th><th>Created date</th><th>Airflow version</th><th>Airflow UI</th></tr></thead><tbody><tr><td>snowflake-airflow</td><td>Creating</td><td>Mar 14, 2022 22:37:04 (UTC+04:00)</td><td>1.10.12</td><td>Open Airflow UI</td></tr></tbody></table>						Name	Status	Created date	Airflow version	Airflow UI	snowflake-airflow	Creating	Mar 14, 2022 22:37:04 (UTC+04:00)	1.10.12	Open Airflow UI
Name	Status	Created date	Airflow version	Airflow UI											
snowflake-airflow	Creating	Mar 14, 2022 22:37:04 (UTC+04:00)	1.10.12	Open Airflow UI											

Then open Airflow UI and add new connection for snowflake:

Edit Connection	
Conn Id *	snowflake_conn
Conn Type	Snowflake
Host	tn37104.eu-central-1.snowflakecomputing.com
Schema	
Login	siddharth
Password
Port	
Extra	{"account": "tn37104.eu-central-1", "warehouse": "compute_wh", "database": "ecommerce_db", "schema": "public", "user": "siddharth", "password": "*****", "private_key": null}

In this tutorial, what has been done:

- Load data from S3 to Snowflake
- Trigger AWS glue
- Execute both the above tasks in parallel

To trigger AWS glue, please add permissions to trigger from AWS Airflow

IAM > Roles > AmazonMWAA-snowflake-airflow-xd1MWG > Add permissions

Attach policy to AmazonMWAA-snowflake-airflow-xd1MWG

▶ Current permissions policies (1)

Other permissions policies (Selected 1/737)

Filter policies by property or policy name and press enter

"glue" X Clear filters

Policy name	Type	Description
AwsGlueDataBrewServicePolicyForInputS3Object-test	Customer managed	
AWSGlueServiceNotebookRole	AWS managed	Policy for AWS Glue service
<input checked="" type="checkbox"/> AWSGlueServiceRole	AWS managed	Policy for AWS Glue service
<input type="checkbox"/> AWSGlueConsoleSageMakerNotebookFullAccess	AWS managed	Provides full access to

Create Glue job:

Copy and paste the code into AWS Glue:

```
from pyspark.sql import SparkSession
from pyspark import SparkContext
from pyspark.sql.functions import count, avg,sum

spark = SparkSession \
    .builder \
    .appName("Glue-pyspark-test") \
    .getOrCreate()

SNOWFLAKE_SOURCE_NAME = "net.snowflake.spark.snowflake"
snowflake_database="ECOMMERCE_DB"
snowflake_schema="ECOMMERCE_DEV"
source_table_name="ORDERS"

snowflake_options = {
    "sfUrl": "https://myorganization-myaccount.snowflakecomputing.com/",
    "sfUser": "mohamedzuhairka",
    "sfPassword": "",
    "sfDatabase": snowflake_database,
    "sfSchema": snowflake_schema,
    "sfWarehouse": "COMPUTE_WH"
}

# df = spark.read \
#     .format(SNOWFLAKE_SOURCE_NAME) \
#     .options(**snowflake_options) \
#     .option("dbtable",source_table_name) \
#     .option("autopushdown", "on") \
#     .load()

fetch_orderes_sql = """
select
O_ORDERKEY,
O_CUSTKEY,
O_ORDERSTATUS,
O_TOTALPRICE,
O_ORDERDATE
from orders
"""

fetch_lineitems_sql = """
select
L_ORDERKEY,
L_SHIPDATE,
L_SHIPMODE
```

```
from LINEITEM
where L_SHIPDATE = '1996-02-13'
"""

df_orders = spark.read \
    .format(SNOWFLAKE_SOURCE_NAME) \
    .options(**snowflake_options) \
    .option("query",fetch_orderes_sql) \
    .option("autopushdown", "on") \
    .load()

df_lineitems = spark.read \
    .format(SNOWFLAKE_SOURCE_NAME) \
    .options(**snowflake_options) \
    .option("query",fetch_lineitems_sql) \
    .option("autopushdown", "on") \
    .load()

df =
df_lineitems.join(df_orders,df_orders.O_ORDERKEY==df_lineitems.L_ORDERKEY,"inner")

df_agg = df.groupBy("L_SHIPMODE","L_SHIPDATE").agg(sum("O_TOTALPRICE"),
count("O_ORDERKEY"))

df_agg.write.format("snowflake") \
    .options(**snowflake_options) \
    .option("dbtable","aggregated_daily_sales").mode("overwrite") \
    .save()

#s3://sf-glue-jobs/spark-jars/snowflake-jdbc-3.13.15.jar,s3://sf-glue-jobs/spark-jars/spark-snow
flake_2.12-2.9.2-spark_3.1.jar
```

Job: pyspark_sales_data_agg

Action ▾ Save Run job Generate diagram ⓘ Insert template at cursor ⓘ Source Target Target Location Transform Spigot ? X

The diagram cannot be generated. Check the annotations in your script.

```

56     .load()
57
58 df_lineitems = spark.read \
59   .format(SNOWFLAKE_SOURCE_NAME) \
60   .options(**snowflake_options) \
61   .option("query",fetch_lineitems_sql) \
62   .option("autopushdown", "on") \
63   .load()
64
65 df = df_lineitems.join(df_orders,df_orders.O_ORDERKEY==df_lineitems.L_ORDERKEY,"inner")
66
67 df_agg = df.groupBy("L_SHIPMODE","L_SHIPDATE").agg(sum("O_TOTALPRICE"), count("O_ORDERKEY"))
68
69
70 df_agg.write.format("snowflake") \
71   .options(**snowflake_options) \
72   .option("dbtable","aggregated_daily_sales").mode("overwrite") \
73   .save()
74
75
76 #s3://sf-glue-jobs/spark-jars/snowflake-jdbc-3.13.15.jar,s3://sf-glue-jobs/spark-jars/spark-snowflake

```

Logs Schema

Create a DAG:

```

import logging
import airflow
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
import boto3
from airflow.contrib.operators.snowflake_operator import SnowflakeOperator
from airflow.contrib.hooks.snowflake_hook import SnowflakeHook
from datetime import datetime, timedelta
from botocore.exceptions import ClientError

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

client = boto3.client('glue',region_name='eu-central-1')

args = {"owner": "Airflow", "start_date": airflow.utils.dates.days_ago(2)}

dag = DAG(
    dag_id="snowflake_automation_dag", default_args=args, schedule_interval=None
)

snowflake_query = [
    """
    use role sysadmin;
    """,
    "use schema \"ECOMMERCE_DB\".\"ECOMMERCE_LIV\";",
]

```

```

    """
    """
    copy into lineitem
    from @stg_lineitem_csv_dev
    file_format = csv_load_format
    ON_ERROR = ABORT_STATEMENT;
"""

]

def execute_job(**kwargs):
    status = client.start_job_run(JobName = "pyspark_sales_data_agg")
    logging.info("GLUE Job Status: %s Execution Time: %s",status['JobRun']['JobRunState'],status['JobRun']['ExecutionTime'])

    while True:
        try:
            status = client.get_job_run(JobName=kwargs['glueJobName'],
                                         RunId=status['JobRunId'])
            if status['JobRun']['JobRunState'] == 'SUCCEEDED':
                break
        except ClientError as e:
            logging.info(e)

with dag:

    copy_data = SnowflakeOperator(
        task_id="insert_snowflake_data",
        sql=snowflake_query ,
        snowflake_conn_id="snowflake_conn"
    )

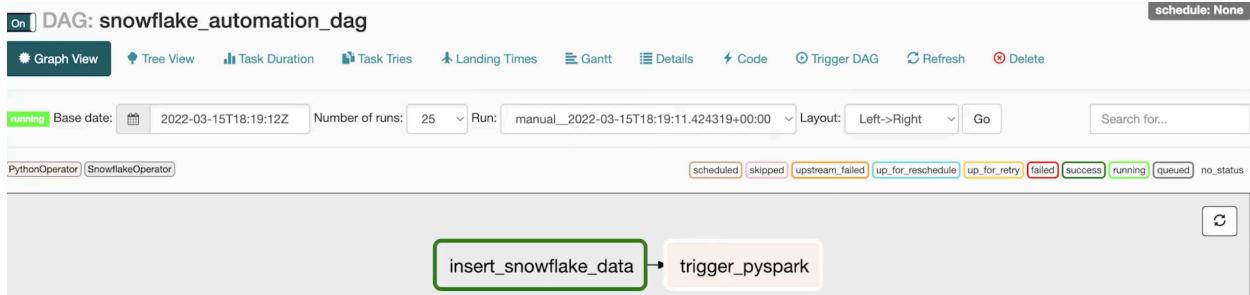
glue_task=PythonOperator(task_id="trigger_pyspark",python_callable=execute_job,execution_timeout=timedelta(minutes=15))

copy_data >> glue_task

```

DAGs

All 1	Active 1	Paused 0	Filter dags			Filter tags	Reset	Search:
			Schedule	Owner	Recent Tasks 1	Last Run 1	DAG Runs 1	Links
	DAG							
	snowflake_automation_dag			Airflow		1 1	2022-03-15, 18:19:11 1	
«	«	1	»	»				Showing 1 to 1 of 1 entries



Snowflake-Kafka Integration:

Download and install kafka:

Download kafka: <https://kafka.apache.org/downloads>

3.7.1

- Released Jun 28, 2024
- [Release Notes](#)
- Docker image: [apache/kafka:3.7.1](#).
- Source download: [kafka-3.7.1-src.tgz \(asc, sha512\)](#)
- Binary downloads:
 - Scala 2.12 - [kafka_2.12-3.7.1.tgz \(asc, sha512\)](#)
 - Scala 2.13 - [kafka_2.13-3.7.1.tgz \(asc, sha512\)](#)

We build for multiple versions of Scala. This only matters if you are using Scala 2.12. Otherwise any version should work (2.13 is recommended).

Kafka 3.7.1 includes a significant number of new features and fixes. For more information, see the [Notes](#).

Extract the TGZ File:

- Right-click the `.tgz` file.
- Hover over the **7-Zip** option in the context menu.
- Choose **Extract Here** to extract in the current folder or **Extract to [folder]** to extract in a subfolder.

This will first extract the `.tar` file inside the `.tgz`, and then you can extract the `.tar` file again the same way to get its contents.

File Explorer				
	Name	Date modified	Type	Size
▶	bin	6/19/2024 3:02 AM	File folder	
▶	config	6/19/2024 3:02 AM	File folder	
▶	libs	6/19/2024 3:02 AM	File folder	
▶	licenses	6/19/2024 3:02 AM	File folder	
▶	site-docs	6/19/2024 3:02 AM	File folder	
▶	LICENSE	6/19/2024 2:57 AM	File	15 KB
▶	NOTICE	6/19/2024 2:57 AM	File	28 KB

Download snowflake kafka connector:

<https://mvnrepository.com/artifact/com.snowflake/snowflake-kafka-connector>



Snowflake Kafka Connector » 2.3.0

Snowflake Kafka Connect Sink Connector

License	Apache 2.0
Tags	streaming kafka connector connection snowflake
HomePage	https://www.snowflake.com/
Date	Jul 10, 2024
Files	pom (23 KB) jar (143.6 MB) View All
Repositories	Central
Ranking	#599522 in MvnRepository (See Top Artifacts)
Vulnerabilities	Vulnerabilities from dependencies: CVE-2024-47561

Add the jar under /libs folder:

is PC > Windows (C:) > Development > Kafka > kafka_2.12-3.7.1 > kafka_2.12-3.7.1 > libs

Name	Date modified	Type	Size
paranamer-2.8	5/23/2022 2:24 AM	Executable Jar File	34 KB
pcollections-4.0.1	4/21/2023 4:44 PM	Executable Jar File	72 KB
plexus-utils-3.3.1	2/16/2024 6:41 PM	Executable Jar File	256 KB
protobuf-java-3.23.4	11/2/2023 7:37 PM	Executable Jar File	1,702 KB
reflections-0.10.2	7/24/2023 7:40 PM	Executable Jar File	128 KB
reload4j-1.2.25	5/30/2023 3:46 AM	Executable Jar File	326 KB
rocksdbjni-7.9.2	8/25/2023 7:48 PM	Executable Jar File	56,641 KB
scala-collection-compat_2.12-2.10.0	3/12/2024 5:10 AM	Executable Jar File	283 KB
scala-java8-compat_2.12-1.0.2	3/12/2024 5:10 AM	Executable Jar File	1,146 KB
scala-library-2.12.18	3/12/2024 5:09 AM	Executable Jar File	5,307 KB
scala-logging_2.12-3.9.4	3/12/2024 5:10 AM	Executable Jar File	58 KB
scala-reflect-2.12.18	3/12/2024 5:09 AM	Executable Jar File	3,585 KB
slf4j-api-1.7.36	4/11/2023 4:38 PM	Executable Jar File	41 KB
slf4j-reload4j-1.7.36	4/11/2023 4:39 PM	Executable Jar File	10 KB
snappy-java-1.1.10.5	10/7/2023 7:52 AM	Executable Jar File	2,246 KB
snowflake-kafka-connector-2.3.0	10/15/2024 9:44 PM	Executable Jar File	147,039 KB
swagger-annotations-2.2.8	4/21/2023 4:47 PM	Executable Jar File	42 KB

Follow these steps to complete kafka setup:

<https://www.geeksforgeeks.org/how-to-install-and-run-apache-kafka-on-windows/>

Setup kafka in local system:

Execute the commands:

```
# Kafka steps

Step-1 - .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
Step-2 - .\bin\windows\kafka-server-start.bat .\config\server.properties
Step-3 - .\bin\windows\kafka-topics.bat --create --topic sales-data --bootstrap-server
localhost:9092

# Delete a topic
- .\bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 --delete --topic test-data

# list all topics
- .\bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092

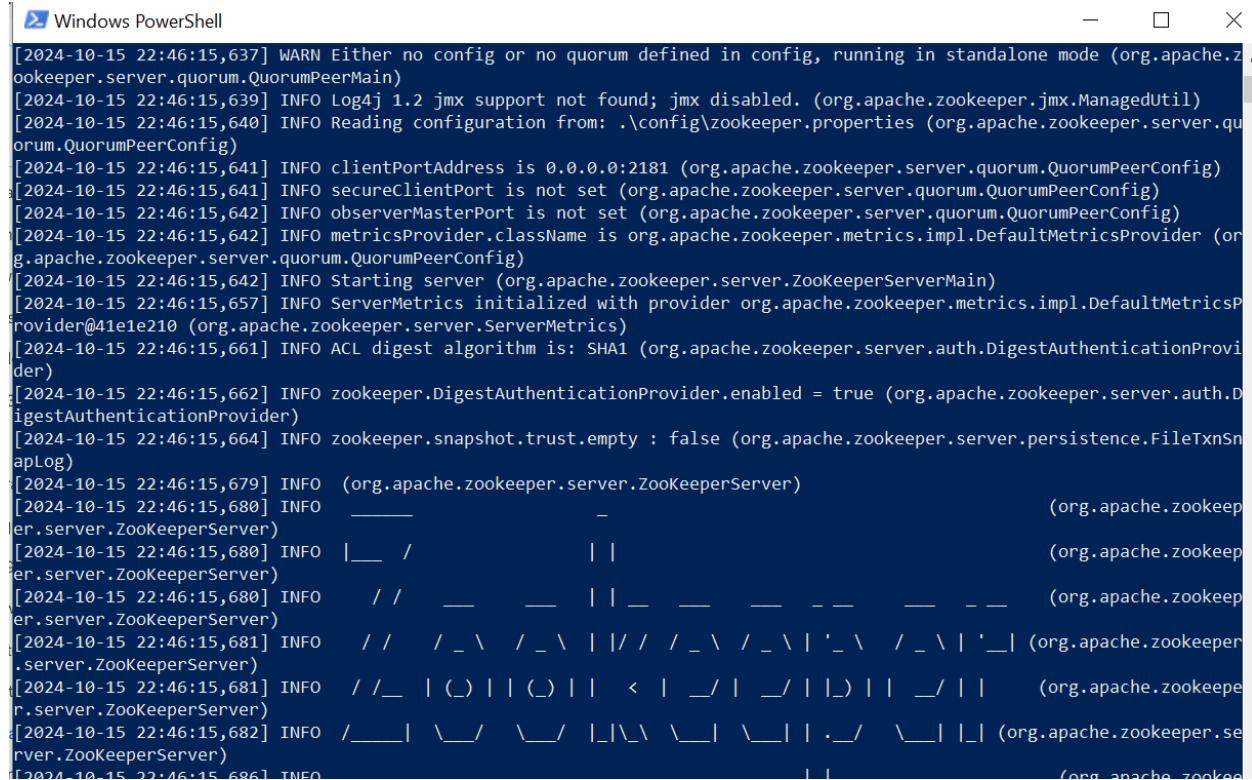
Step-4 - .\bin\windows\kafka-console-consumer.bat --topic sales-data --from-beginning
--bootstrap-server localhost:9092
```

Step-5 .\bin\windows\connect-standalone.bat .\config\connect-standalone.properties
.config\SF_connect.properties

You may face the problem like below in command prompt:

```
C:\Development\Kafka\kafka_2.12-3.7.1\kafka_2.12-3.7.1>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
The input line is too long.
The syntax of the command is incorrect.
```

Try moving the Kafka folder to a location with a shorter path, such as C:\kafka. Then, execute the command in powershell instead of cmd.



A screenshot of a Windows PowerShell window. The title bar says "Windows PowerShell". The window contains a large amount of text output from a Zookeeper server. The log entries are timestamped and show various INFO and WARN messages related to the server's configuration and startup process. The text is mostly in blue and black, with some red highlights for error messages.

```
[2024-10-15 22:46:15,637] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)
[2024-10-15 22:46:15,639] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)
[2024-10-15 22:46:15,640] INFO Reading configuration from: .\config\zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-10-15 22:46:15,641] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-10-15 22:46:15,641] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-10-15 22:46:15,642] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-10-15 22:46:15,642] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-10-15 22:46:15,642] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-10-15 22:46:15,657] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@41e1e210 (org.apache.zookeeper.server.ServerMetrics)
[2024-10-15 22:46:15,661] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-10-15 22:46:15,662] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)
[2024-10-15 22:46:15,664] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-10-15 22:46:15,679] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-10-15 22:46:15,680] INFO _____ - (org.apache.zookeeper.server.ZooKeeperServer)
[2024-10-15 22:46:15,680] INFO |__ / ||| (org.apache.zookeeper.server.ZooKeeperServer)
[2024-10-15 22:46:15,680] INFO / / __ _ ||_ _ _ _ _ _ _ _ _ _ (org.apache.zookeeper.server.ZooKeeperServer)
[2024-10-15 22:46:15,681] INFO / / / _ \ / _ \ | | / / / _ \ / _ \ | ' _ \ / _ \ | ' _ | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-10-15 22:46:15,681] INFO / / _ | ( ) | | ( ) | | < | _ / | _ / | | _ / | | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-10-15 22:46:15,682] INFO / ____| \ _/ \ _/ | | \ _/ \ _/ | | . _/ \ _/ | | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-10-15 22:46:15,686] INFO _____ | | (org.apache.zookeeper.server.ZooKeeperServer)
```

Copy the below python program and run it.

```
import pandas as pd
import random
from faker import Faker
from random import randrange
from datetime import datetime
from time import sleep
from json import dumps
from kafka import KafkaProducer
```

```

nr_of_customers = 100

fake = Faker('de_DE')

customers = dict()

topic_name='sales-data'

producer = KafkaProducer(bootstrap_servers=['localhost:9092'],value_serializer=lambda x:
dumps(x).encode('utf-8'))

for customers_id in range(nr_of_customers):

    # Create transaction date
    d1 = datetime.strptime('1/1/2021', '%m/%d/%Y')
    d2 = datetime.strptime('8/10/2021', '%m/%d/%Y')
    transaction_date = fake.date_between(d1, d2)

    #create customer's name
    name = fake.name()

    # Create gender
    gender = random.choice(["M", "F"])

    # Create email
    email = fake.ascii_email()

    #Create city
    city = fake.city()

    #create product ID in 8-digit barcode
    product_id = fake.ean(length=8)

    #create amount spent
    amount_spent = fake.pyfloat(right_digits=2, positive=True, min_value=1, max_value=100)

    customers={
        'transaction_date':str(transaction_date) ,
        'name':name,'gender':gender,'city':city,
        'email':email,'product_id':product_id,
        'amount_spent':amount_spent
    }
    print(customers)

    producer.send(topic_name, value=dumps(customers))
    sleep(5)

```

```
transaction_date': '2021-05-19', 'name': 'Friedhilde Knappe', 'gender': 'M', 'city': 'Hohenstein-Ernstthal', 'email': 'umielcarek@web.de', 'product_id': '33382630', 'amount_spent': 39.91}
```

Kafka Snowflake integration:

Copy the below file and place it inside the config folder as “SF_connect.properties”

```
connector.class=com.snowflake.kafka.connector.SnowflakeSinkConnector
tasks.max=8
topics=fake-data
snowflake.topic2table.map=fake-data:sales_data
buffer.count.records=10000
buffer.flush.time=60
buffer.size.bytes=5000000
snowflake.url.name=
snowflake.user.name=mohamedzuhairka
snowflake.private.key=
snowflake.database.name=ecommerce_db
snowflake.schema.name=kafka_liv_schema
key.converter=com.snowflake.kafka.connector.records.SnowflakeJsonConverter
value.converter=com.snowflake.kafka.connector.records.SnowflakeJsonConverter
name=fakedata_kafka_test
```

is PC > Windows (C:) > Development > Kafka > kafka_2.12-3.7.1 > kafka_2.12-3.7.1 > config

Name	Date modified	Type	Size
kraft	6/19/2024 2:57 AM	File folder	
connect-console-sink	6/19/2024 2:57 AM	PROPERTIES File	1 KB
connect-console-source	6/19/2024 2:57 AM	PROPERTIES File	1 KB
connect-distributed	6/19/2024 2:57 AM	PROPERTIES File	6 KB
connect-file-sink	6/19/2024 2:57 AM	PROPERTIES File	1 KB
connect-file-source	6/19/2024 2:57 AM	PROPERTIES File	1 KB
connect-log4j	6/19/2024 2:57 AM	PROPERTIES File	3 KB
connect-mirror-maker	6/19/2024 2:57 AM	PROPERTIES File	3 KB
connect-standalone	6/19/2024 2:57 AM	PROPERTIES File	3 KB
consumer	6/19/2024 2:57 AM	PROPERTIES File	2 KB
log4j	6/19/2024 2:57 AM	PROPERTIES File	5 KB
producer	6/19/2024 2:57 AM	PROPERTIES File	3 KB
server	10/15/2024 9:47 PM	PROPERTIES File	7 KB
SF_connect	10/15/2024 10:37 PM	PROPERTIES File	1 KB
tools-log4j	6/19/2024 2:57 AM	PROPERTIES File	2 KB
trgdor	6/19/2024 2:57 AM	CONF File	2 KB
zookeeper	10/15/2024 9:48 PM	PROPERTIES File	2 KB

The below three properties will decide the frequency at which data will be inserted into snowflake:

buffer.count.records=10000

buffer.flush.time=60

buffer.size.bytes=5000000

If any one of the condition is met, then it will insert the records into snowflake.

Also, in connect-standalone.properties file, provide the pluginpath to the libs folder.

```
# Set to a list of filesystem paths separated by commas (,) to enable class loading isolation for plugins
# (connectors, converters, transformations). The list should consist of top level directories that include
# any combination of:
# a) directories immediately containing jars with plugins and their dependencies
# b) uber-jars with plugins and their dependencies
# c) directories immediately containing the package directory structure of classes of plugins and their dependencies
# Note: symlinks will be followed to discover dependencies or plugins.
# Examples:
# plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors,
#plugin.path=C:/Development/Kafka/libs/
```

Create encryption key for connection:

Download openssl and execute the commands

```
# Create an unencrypted private key
openssl genrsa -out rsa_key.pem 2048
```

```
# Create a public key referencing the above private key
openssl rsa -in rsa_key.pem -pubout -out rsa_key.pub
```

```
siddharthagunath@siddharths-MBP kafka_2.12-3.1.0 % openssl genrsa -out rsa_key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
siddharthagunath@siddharths-MBP kafka_2.12-3.1.0 %
```

```
siddharthagunath@siddharths-MBP kafka_2.12-3.1.0 % ls -l
total 96
-rw-r--r--@ 1 siddharthagunath staff 14536 Jan 12 13:01 LICENSE
-rw-r--r--@ 1 siddharthagunath staff 28184 Jan 12 13:01 NOTICE
drwxr-xr-x@ 40 siddharthagunath staff 1280 Jan 12 13:04 bin
drwxr-xr-x@ 23 siddharthagunath staff 736 Mar 13 16:17 config
drwxr-xr-x@ 105 siddharthagunath staff 3360 Feb 7 23:17 libs
drwxr-xr-x@ 12 siddharthagunath staff 384 Jan 12 13:04 licenses
drwxr-xr-x 146 siddharthagunath staff 4672 Mar 13 16:03 logs
-rw-r--r-- 1 siddharthagunath staff 1675 Mar 13 19:29 rsa_key.pem
drwxr-xr-x@ 3 siddharthagunath staff 96 Jan 12 13:04 site-docs
siddharthagunath@siddharths-MBP kafka_2.12-3.1.0 %
```

```
siddharthagunath@siddharths-MBP kafka_2.12-3.1.0 % openssl rsa -in rsa_key.pem -pubout -out rsa_key.pub
writing RSA key
siddharthagunath@siddharths-MBP kafka_2.12-3.1.0 % ls -l
total 104
-rw-r--r--@ 1 siddharthagunath staff 14536 Jan 12 13:01 LICENSE
-rw-r--r--@ 1 siddharthagunath staff 28184 Jan 12 13:01 NOTICE
drwxr-xr-x@ 40 siddharthagunath staff 1280 Jan 12 13:04 bin
drwxr-xr-x@ 23 siddharthagunath staff 736 Mar 13 16:17 config
drwxr-xr-x@ 105 siddharthagunath staff 3360 Feb 7 23:17 libs
drwxr-xr-x@ 12 siddharthagunath staff 384 Jan 12 13:04 licenses
drwxr-xr-x 146 siddharthagunath staff 4672 Mar 13 16:03 logs
-rw-r--r-- 1 siddharthagunath staff 1675 Mar 13 19:29 rsa_key.pem
-rw-r--r-- 1 siddharthagunath staff 451 Mar 13 19:29 rsa_key.pub
drwxr-xr-x@ 3 siddharthagunath staff 96 Jan 12 13:04 site-docs
siddharthagunath@siddharths-MBP kafka_2.12-3.1.0 %
```

Copy paste the private key in SF_connect.properties (add backslash at every line)

```
Users > siddharthagunath > Documents > kafka-snowflake > kafka_2.12-3.1.0 > config > SF_connect.properties
7 buffer.size.bytes=5000000
8 snowflake.url.name=az32171.eu-central-1.snowflakecomputing.com
9 snowflake.user.name=siddharth
10 snowflake.private.key=MIIEdwIBAAKCAQEAt7c9HXV+qRQD3uB1RiJQ1BYaKFTLyb/eUCGkXpqhp/4/B0Qqy\YErXL9hqvWR13c10y0QChyKhWlq2ZJRKAYBTQu33KKBXfb2qRLwuoMgqI3moiml0\NgdY1ebj2t5Ec+P+ePY7z5eTH60DGEq04khS71mpZ7f92dGdEEvuPsxNTIYvHy\qRiq8DEbjS8sxYtUFzj1mzzYqGJZQ6WTdU86eeFQGdDcVQDsEV1mQuFwnqsENjbR\Zz0Szsh0KPDpibp0BogjWgSP/XcF7FUXuS+3P2EVqSAF29WBeVseYcTJCj15W1/L5\+1ZcuZ+qRgojsPM885YWjPq7Zi4qZAREX3txWQIDAQABAOIBAQCyN3aJ+VmHIPSE\XCRwMHGT1IHC5U1er7GWo/t/gBL/30IpINUQ6MVfK9bCXHkgYe480pj6Y09+4SBz\09ZDhYQgUmm5XoBxmWGY5Twkbh12PIQcRm861kKy+rejnJK8Lw7T4+Rq7HLrcQSb\vbRazEyz5w0F+4wWJoLHiCcAT5tu3X4E8pA+0XCyC0jc5p4Fbjtm+/sISqcdFNlN\uScUdY1vzuy2/rn2sln4Egbf01Vv0mY6BlwmHvgWbUYXW4D40p14v68pvXEzWDg0\PEtV5lRMN7IBM6w6n/wQ4i3w2w7x0fWZSeKcbTU00h9XhIdUFnYGAmvf1hNsM7wU\qsvylaUBAoGBAPcyYq7HHgezMVcuGXVfW00mq3Xc2Uf+yoSTImQWt8FdTCtiGSAA\uJwv6Qdy5B5abYB58v4s+06011e7004d155fxd5NcTYCWE+ELi2oMe7GdPbz4l
```

Update the public key in snowflake user:

```
1 use schema ecommerce_db.ecommerce_dev;
2
3
4 alter user siddharth set RSA_PUBLIC_KEY='MIIBIjANBgkqhkiG9w0BAQEFA0CAQ8AMIIBCgKCAQEA7c9HXV+qRQD3uB1RiJQ1
5 BYaKFTLyb/eUCGkXpqhp/4/B0QqyYErXL9hqvWR13c10y0QChyKhWlq2ZJRKAyBT
6 Qu33KKBXfB2qRLwoMqqI3moiml0NgdY1ebj2t5Ec+P+ePY7z5eTH60DGEq04khS
7 71mppZ7fq92dGdEEvuPsxNTIYvHyqRiq8DEbjS8sxYtUFzj1mzzYqGJZQ6WTdU86
8 eeFQGdCvQDsEV1mQuFwnqsENjbRZZoSzsH0KPDpipuBogjWgSP/XcF7FUxuS+3P
9 2EVqSAF29WBeVseYcTJCj15W1/L5+1ZcuZ+qRgojsPM885YWjPq7Z14qZAREX3tx
10 WQIDAQAB';
11
12 desc user siddharth;
13
14
```

Stream data in action:

Run the below commands:

Kafka steps

```
Step-1 - .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
Step-2 - .\bin\windows\kafka-server-start.bat .\config\server.properties
Step-3 - .\bin\windows\kafka-topics.bat --create --topic sales-data --bootstrap-server
localhost:9092

Step-4 - .\bin\windows\kafka-console-consumer.bat --topic sales-data --from-beginning
--bootstrap-server localhost:9092

Step-5 py fake_data.py

Step-6 .\bin\windows\connect-standalone.bat .\config\connect-standalone.properties
.\config\SF_connect.properties
```

Make sure to create database and schema in snowflake. (As provided names in “**SF_connect.properties**”.

17
18 `create schema ecommerce_db.kafka_live_streaming;`

Results Data Preview

✓ Query_ID SQL 49ms 1 rows

Filter result...

Row	status
1	Schema KAFKA_LIVE_STREAMING successfully created.

Start the python program(located under ([Setup Kafka in local systems](#)) to write data in topics:

```
ddharthaguntha@ddharthas-MBP:~/Section16-kafka-streaming$ python3 fake_data.py
transaction_date': '2021-01-13', 'name': 'Kathy Kostolzin', 'gender': 'M', 'city': 'Niesky', 'email': 'walterrossi@yahoo.de', 'product_id': '00050685', 'amount_spent': 45.9}
transaction_date': '2021-04-27', 'name': 'Dr. Philipp Heß B.Sc.', 'gender': 'F', 'city': 'Bad Freienwalde', 'email': 'gaetanokabus@scholz.net', 'product_id': '55611060', 'amount_spent': 64.23}
```

After start the connection, once any **three condition** met, it will flush the data into snowflake. And you can see the records in snowflake.

The screenshot shows the Snowflake interface. On the left, there's a sidebar with a tree view of databases, schemas, and tables. A yellow box highlights the 'SALES_DATA' table under the 'KAFKA_LIVE_STREAMING' schema. The main area shows a query editor with the following code:

```
17
18 create schema
19
20
21
22
```

Below the code, the results tab shows the creation of the schema. The data preview tab shows the contents of the 'SALES_DATA' table:

Table: ECOMMERCE_DB.KAFKA_LIVE_STREAMING.SALES_DATA

Row	RECORD_METADATA	RECORD_CONTENT
1	{ "CreateTime": 1647186009603, "key": {}, "key_schema_id": 0, "offset": 0, "partition": 0, ... }	{"transaction_date": "2021-02-04", "name": "Prof. Diana J\u00e4u00e4ntschi B.Eng.", "city": "Niesky", "email": "walterrossi@yahoo.de", "product_id": "00050685", "amount_spent": 45.9}
2	{ "CreateTime": 1647186014610, "key": {}, "key_schema_id": 0, "offset": 1, "partition": 0, ... }	{"transaction_date": "2021-08-04", "name": "Jorge Dehmel", "gender": "M", "city": "Bad Freienwalde", "email": "gaetanokabus@scholz.net", "product_id": "55611060", "amount_spent": 64.23}
3	{ "CreateTime": 1647186019615, "key": {}, "key_schema_id": 0, "offset": 2, "partition": 0, ... }	{"transaction_date": "2021-01-07", "name": "Sigmar Hermighausen", "gender": "M", "city": "Niesky", "email": "walterrossi@yahoo.de", "product_id": "00050685", "amount_spent": 45.9}
4	{ "CreateTime": 1647186024620, "key": {}, "key_schema_id": 0, "offset": 3, "partition": 0, ... }	{"transaction_date": "2021-06-22", "name": "Frieda Rudolph", "gender": "F", "city": "Bad Freienwalde", "email": "gaetanokabus@scholz.net", "product_id": "55611060", "amount_spent": 64.23}
5	{ "CreateTime": 1647186029626, "key": {}, "key_schema_id": 0, "offset": 4, "partition": 0, ... }	{"transaction_date": "2021-04-03", "name": "Kunibert Becker B.Sc.", "gender": "M", "city": "Niesky", "email": "walterrossi@yahoo.de", "product_id": "00050685", "amount_spent": 45.9}

Record metadata:

The screenshot shows the Snowflake interface. On the left, there's a sidebar with a tree view of databases, schemas, and tables. A yellow box highlights the 'RECORD_METADATA' column in the table data. The main area shows a query editor with the following code:

```
15
16
17
18 create schema
19
20
21
22
```

Below the code, the results tab shows the creation of the schema. The data preview tab shows the contents of the 'SALES_DATA' table:

Table: ECOMMERCE_DB.KAFKA_LIVE_STREAMING.SALES_DATA

Row	RECORD_METADATA
1	{ "CreateTime": 1647186009603, "key": {}, "key_schema_id": 0, "offset": 0, "partition": 0, ... }

A modal window titled 'Details' is open, showing the JSON structure of the first record:

```
1 {
2   "CreateTime": 1647186009603,
3   "key": {},
4   "key_schema_id": 0,
5   "offset": 0,
6   "partition": 0,
7   "topic": "sales-data"
8 }
```

```

15
16
17
18 create schema
19
20
21
22

```

Results Data Preview

Table: ECOMMERCE_DB.P

Filter result...

Row RECORD_M

1 {"CreateTir

Details

```
1 [{"transaction_date": "2021-08-04", "name": "Jorge Dehme", "gender": "M", "city": "Konstanz", "email": "foersterekkehard@winkler.de", "product_id": "57694450", "amount_spent": 23.8}
```

[Copy](#)

Columns ▾

Open History

Time Travel:

```

use role accountadmin;

Use database ECOMMERCE_DB;

Use SCHEMA ECOMMERCE_DEV;

SHOW TABLES;

alter table "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM" set
data_retention_time_in_days=90;

ALTER database ECOMMERCE_DB set data_retention_time_in_days=1;

ALTER schema ECOMMERCE_DB.ECOMMERCE_LIV set data_retention_time_in_days=1;

create table lineitem_test as select * from LINEITEM limit 10;

SELECT * FROM LINEITEM before(timestamp => '2024-10-17 11:00:56.786'::timestamp) limit
10;

SELECT * FROM LINEITEM at(timestamp => '2024-10-17 11:00:56.786'::timestamp) limit 10;

CREATE TABLE LINEITEM_restored CLONE LINEITEM at(timestamp => '2024-10-18
11:00:56.786'::timestamp);

create database testing_db;

alter database testing_db set data_retention_time_in_days=0;

drop database testing_db;

undrop database testing_db;

```

```

create database testing_db;

alter database testing_db set data_retention_time_in_days=1;

drop database testing_db;

undrop database testing_db;

```

↳ SHOW TABLES;

5 alter table "ECOMMERCE_DB"."ECOMMERCE_DEV"."LINEITEM" set data_retention_time_in_days=90;

	created_on	name	database_name
1	2024-10-01 23:27:28.302 -0700	LINEITEM	ECOMMERCE_DB
2	2024-10-05 02:52:12.794 -0700	LINEITEM_CLONE	ECOMMERCE_DB
3	2024-10-01 23:43:52.119 -0700	ORDERS	ECOMMERCE_DB
4	2024-10-14 09:04:47.104 -0700	SPARK_OUTPUT_TABLE	ECOMMERCE_DB
5	2024-10-11 11:20:15.596 -0700	SUPPLIER	ECOMMERCE_DB
6	2024-10-01 23:44:00.010 -0700	ORDERS	ECOMMERCE_DB

Query Details

- Query duration: 787ms
- Rows: 93
- Query ID: 01b7ca1d-0003-ecdc-0...

Ask Copilot

90 days retention period is for Snowflake enterprise edition.

database_name schema_name kind comment cluster_by rows bytes owner rete : time automatic_clust change_tracking sea

ECOMMERCE... ECOMMERCE... TABLE LINEAR(L_S... 154616896 4035046912 SYSADMIN 9 : ON OFF OFF

ECOMMERCE... ECOMMERCE... TABLE 1500000000 41566244864 SYSADMIN 1 OFF OFF OFF

7 ALTER database ECOMMERCE_DB set data_retention_time_in_days=1;

8

9 ALTER schema ECOMMERCE_DB.ECOMMERCE_LIV set data_retention_time_in_days=1;

10

	status		Query Details
1	Statement executed successfully.		Query duration: 34ms

	status		Query Details
1	Statement executed successfully.		Query duration: 65ms

```
15 | create table lineitem_test as select * from LINEITEM limit 10;
```

↳ Results ⚡ Chart

🔍 ⏪ ⏴ ⏵ ⏷ ⏸

	status
1	Table LINEITEM_TEST successfully created.

Query Details ...

Query duration 2.0s

Rows 1

Query ID 01b7ca23-0004-039e-0...

```
17 | SELECT * FROM LINEITEM before(timestamp => '2024-10-18 11:00:56.786'::timestamp) limit 10;
18 |
```

↳ Results ⚡ Chart

🔍 ⏪ ⏴ ⏵ ⏷ ⏸

	L_ORDERKEY	L_PARTKEY	L_SUPPKEY	L_LINENUMBER	L_QUANTITY	L_EXTENDEDPRICE
1	5698238502	173336436	8336471	4	44.00	1935.20
2	5698474147	20961814	3461817	2	41.00	1703.40
3	5698224512	12733887	5233889	2	34.00	1482.80
4	5698388194	126980983	1981008	3	34.00	1482.80
5	5698087040	174543546	9543581	1	6.00	27.44
6	5698072965	133514607	1014647	2	21.00	445.76

Query Details ...

Query duration 78ms

Rows 10

Query ID 01b7ca24-0004-039e-0...

Show more ▾

L_ORDERKEY Ask Copilot 

```
17 | SELECT * FROM LINEITEM before(timestamp => '2024-10-17 11:00:56.786'::timestamp) limit 10;
18 |
```

↳ Results ⚡ Chart

🔍 ⏪ ⏴ ⏵ ⏷ ⏸



Time travel data is not available for table LINEITEM. The requested time is either beyond the allowed time travel period or before the object creation time.

18

19 | SELECT * FROM LINEITEM at(timestamp => '2024-10-18 11:00:56.786'::timestamp) limit 10;

	L_ORDERKEY	L_PARTKEY	L_SUPPKEY	L_LINENUMBER	L_QUANTITY	L_EXTENDEDPRICE
1	5698238502	173336436	8336471	4	44.00	1932.00
2	5698474147	20961814	3461817	2	41.00	1701.00
3	5698224512	12733887	5233889	2	34.00	1399.00
4	5698388194	126980983	1981008	3	34.00	1399.00
5	5698087040	174543546	9543581	1	6.00	26.00
6	5698072965	133514607	1014647	2	21.00	483.00

18

19 | SELECT * FROM LINEITEM at(timestamp => '2024-10-17 11:00:56.786'::timestamp) limit 10;

	L_ORDERKEY	L_PARTKEY	L_SUPPKEY	L_LINENUMBER	L_QUANTITY	L_EXTENDEDPRICE
1	5698238502	173336436	8336471	4	44.00	1932.00
2	5698474147	20961814	3461817	2	41.00	1701.00
3	5698224512	12733887	5233889	2	34.00	1399.00
4	5698388194	126980983	1981008	3	34.00	1399.00
5	5698087040	174543546	9543581	1	6.00	26.00
6	5698072965	133514607	1014647	2	21.00	483.00

18

19 | SELECT * FROM LINEITEM at(timestamp => '2024-10-17 11:00:56.786'::timestamp) limit 10;

	L_ORDERKEY	L_PARTKEY	L_SUPPKEY	L_LINENUMBER	L_QUANTITY	L_EXTENDEDPRICE
1	5698238502	173336436	8336471	4	44.00	1932.00
2	5698474147	20961814	3461817	2	41.00	1701.00
3	5698224512	12733887	5233889	2	34.00	1399.00
4	5698388194	126980983	1981008	3	34.00	1399.00
5	5698087040	174543546	9543581	1	6.00	26.00
6	5698072965	133514607	1014647	2	21.00	483.00

⚠

Time travel data is not available for table LINEITEM. The requested time is either beyond the allowed time travel period or before the object creation time.

21 | CREATE TABLE LINEITEM_restored CLONE LINEITEM at(timestamp => '2024-10-18 11:00:56.786'::timestamp);

22

	status
1	Table LINEITEM_RESTORED successfully created.

23 | create database testing_db;

24

	status
1	Database TESTING_DB successfully created.

```
25 | alter database testing_db set data_retention_time_in_days=0;
26 |
```

↳ Results ↵ Chart

status	
1	Statement executed successfully.

Query Details ...
Query duration 53ms
Rows 1

```
27 | drop database testing_db;
```

↳ Results ↵ Chart

status	
1	TESTING_DB successfully dropped.

Query Details ...
Query duration 68ms
Rows 1

```
29 | undrop database testing_db;
```

↳ Results ↵ Chart

⚠

Database TESTING_DB did not exist or was purged.

```
23 | create database testing_db;
24 |
```

↳ Results ↵ Chart

status	
1	Database TESTING_DB successfully created.

Query Details ...
Query duration 210ms
Rows 1

```
25 | alter database testing_db set data_retention_time_in_days=1;
```

↳ Results ↵ Chart

status	
1	Statement executed successfully.

Query Details ...
Query duration 92ms
Rows 1

```

27 | drop database testing_db;
28 |

```

Results

	status
1	TESTING_DB successfully dropped.

Query Details ...
 Query duration 83ms
 Rows 1

```

29 | undrop database testing_db;

```

Results

	status
1	Database TESTING_DB successfully restored.

Query Details ...
 Query duration 73ms
 Rows 1

Masking Policy:

```

use role accountadmin;

use "ECOMMERCE_DB"."ECOMMERCE_DEV";

create table orders_test as select * from
"ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS" limit 20;

create or replace masking policy mask_order_value as (val NUMBER) returns number ->
  case
    when current_role() in ('reporting_intern','REPORTING_INTERNAL') then
      99999999999999999999
    else val
  end;

ALTER TABLE IF EXISTS "ECOMMERCE_DB"."ECOMMERCE_DEV"."ORDERS_TEST"
MODIFY COLUMN O_TOTALPRICE SET MASKING POLICY mask_order_value;

----- Create Role for Column-Level Access Policy - Data Masking -----

create or replace role reporting_intern;
grant usage on warehouse compute_wh to role reporting_intern;
grant usage on database ECOMMERCE_DB to role reporting_intern;
grant usage on schema ECOMMERCE_DB.ECOMMERCE_DEV to role reporting_intern;
grant select on ECOMMERCE_DB.ECOMMERCE_DEV.orders_test to role reporting_intern;
grant role reporting_intern to user siddharth;

```

```

---- Switch the role -----
use role reporting_intern;
use warehouse compute_wh;
---- Test the column level masking policy -----
select * from ECOMMERCE_DB.EMCERCE_DEV.orders_test;

---- Remove Column Level Masking -----

use role accountadmin;
ALTER TABLE orders_test MODIFY COLUMN O_TOTALPRICE UNSET MASKING POLICY

```

4
5 | create table orders_test as select * from "ECOMMERCE_DB"."EMCERCE_DEV"."ORDERS" limit 20;
6

↳ Results | Chart

	status	Query Details	...
1	Table ORDERS_TEST successfully created.	Query duration	1.7s
		Rows	1

6
7 create or replace masking policy mask_order_value as (val NUMBER) returns number ->
8 case
9 when current_role() in ('reporting_intern', 'REPORTING_INTERN') then 99999999999999999999
10 else val
11 end;
12
13 ALTER TABLE IF EXISTS "ECOMMERCE_DB"."EMCERCE_LIV"."ORDERS_TEST"
14 MODIFY COLUMN O_TOTALPRICE SET MASKING POLICY mask_order_value;

Results Data Preview

✓ Query ID SQL 92ms 1 rows

Filter result...

Row	status
1	Masking policy MASK_ORDER_VALUE successfully created.

```

12
13 ALTER TABLE IF EXISTS "ECOMMERCE_DB"."ECOMMERCE_LIV"."ORDERS_TEST"
14   MODIFY COLUMN O_TOTALPRICE SET MASKING POLICY mask_order_value;

```

Results Data Preview

✓ [Query ID](#) [SQL](#) 77ms 1 rows

Filter result...



[Copy](#)

Row	status
1	Statement executed successfully.

```

18
19 create or replace role reporting_intern;
20
21 grant usage on warehouse prod_xl to role reporting_intern;
22 grant usage on database ECOMMERCE_DB to role reporting_intern;
23 grant usage on schema ECOMMERCE_DB.ECOMMERCE_LIV to role reporting_intern;
24 grant select on ECOMMERCE_DB.ECOMMERCE_LIV.orders_test to role reporting_intern;
25 grant role reporting_intern to user siddharth;
26
27 ----- Switch the role -----
28 use role reporting_intern;
29 use warehouse prod_xl;
30 ----- Test the column level masking policy -----
31 select * from ECOMMERCE_DB.ECOMMERCE_LIV.orders_test;

```

Results Data Preview

✓ [Query ID](#) [SQL](#) 45ms 1 rows

Filter result...



[Copy](#)

Row	status
1	Statement executed successfully.

```

32
33 select * from ECOMMERCE_DB.ECOMMERCE_LIV.orders_test;

```

Results Data Preview

[Open History](#)

✓ [Query ID](#) [SQL](#) 647ms 20 rows

Filter result...



[Copy](#)

Columns ▾

Row	O_ORDERKEY	O_CUSTKEY	O_ORDERSTATUS	O_TOTALPRICE	O_ORDERDATE	O_ORDERPRIORITY	O_CLERK	O_SHIPPRIORITY	O_COMMENT
1	751786246	119293057	O	999999999999999999999999.00	1997-07-23	5-LOW	Clerk#00025...	0	furiously abov...
2	2515452133	17075774	O	999999999999999999999999.00	1997-07-23	3-MEDIUM	Clerk#00042...	0	y. bold pinto b...
3	750374880	139728394	O	999999999999999999999999.00	1997-07-23	5-LOW	Clerk#000199...	0	e final, bold p...
4	2515084836	53753054	O	999999999999999999999999.00	1997-07-23	1-URGENT	Clerk#00037...	0	structions. furi...
5	750743682	148530434	O	999999999999999999999999.00	1997-07-23	5-LOW	Clerk#00098...	0	according to t...

Row level access policy:

```
use role accountadmin;

create database test_db;

create schema test_schema;

create or replace table test_db.test_schema.item as
select * from "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."ITEM" where i_category
in ('Sports','Music') limit 100;

CREATE TABLE access_management (role string, category string);

INSERT INTO access_management VALUES ('SPORTS_ADMIN', 'Sports'),
('MUSIC_ADMIN', 'Music');

select * from access_management;

SELECT 1 FROM access_management
WHERE category = 'Sports'
AND role = 'SPORTS_ADMIN';

CREATE or replace ROW ACCESS POLICY category_level_access AS
(category_filter VARCHAR) RETURNS BOOLEAN ->
CURRENT_ROLE() = 'ACCOUNTADMIN'
OR EXISTS
(
    SELECT 1 FROM access_management
    WHERE category = category_filter
    AND role = CURRENT_ROLE()
);

ALTER TABLE item ADD ROW ACCESS POLICY category_level_access ON (i_category);

select * from item;

use role accountadmin;

create role SPORTS_ADMIN;

grant usage on warehouse compute_wh to role SPORTS_ADMIN;

grant usage on database TEST_DB to role SPORTS_ADMIN;

grant usage on schema TEST_DB.TEST_SCHEMA to role SPORTS_ADMIN;
```

```
grant select on TEST_DB.TEST_SCHEMA.ITEM to role SPORTS_ADMIN;
```

```
grant role SPORTS_ADMIN to user siddharth;
```

```
---- Switch the role -----
```

```
use role SPORTS_ADMIN;
```

```
use warehouse compute_wh;
```

```
---- Test the Row Level Access Policy -----
```

```
select * from TEST_DB.TEST_SCHEMA.ITEM ;
```

```
6  create or replace table test_db.test_schema.item as
7  select * from "SNOWFLAKE_SAMPLE_DATA"."TPCDS_SF100TCL"."ITEM" where i_category in ('Sports','Music') limit 100;
8
9
10 | CREATE TABLE access_management (role string, category string);
11
12 | INSERT INTO access_management VALUES ('SPORTS_ADMIN', 'Sports'), ('MUSIC_ADMIN', 'Music');
13
14 | select * from access_management;
15
```

Results

status
1 Table ITEM successfully created.

Query Details

Query duration 2.4s

Rows 1

Results

status
1 Table ACCESS_MANAGEMENT successfully created.

Query Details

Query duration 160ms

Rows 1

Results

number of rows inserted
2

Query Details

Query duration 650ms

Rows 1

Results

ROLE	CATEGORY
1 SPORTS_ADMIN	Sports
2 MUSIC_ADMIN	Music

Query Details

Query duration 151ms

Rows 2

Query ID 01e7ab26-0003-0000-0000-000000000000

```

17 CREATE OR REPLACE ROW ACCESS POLICY category_level_access AS
18 (category_filter VARCHAR) RETURNS BOOLEAN ->
19 CURRENT_ROLE() = 'ACCOUNTADMIN'
20 OR EXISTS
21 (
22     SELECT 1 FROM access_management
23     WHERE category = category_filter
24     AND role = CURRENT_ROLE()
25 );
26
27 ALTER TABLE item ADD ROW ACCESS POLICY category_level_access ON (i_category);
28
29 select * from item;
30

```

Results Data Preview

Query ID SQL 49ms 1 rows

Filter result... Copy

Row	status
-----	--------

1 Row access policy 'CATEGORY_LEVEL_ACCESS' is successfully created.

```

27 ALTER TABLE item ADD ROW ACCESS POLICY category_level_access ON (i_category);
28
29 select * from item;
30

```

Results Data Preview

Query ID SQL 91ms 1 rows

Filter result... Copy

Row	status
-----	--------

1 Statement executed successfully.

```

29 select count(1) from item;
30

```

Results Data Preview

Query ID SQL 190ms 1 rows

Filter result... Copy

Columns ▾

Row	
-----	--

COUNT(1)

1

100