

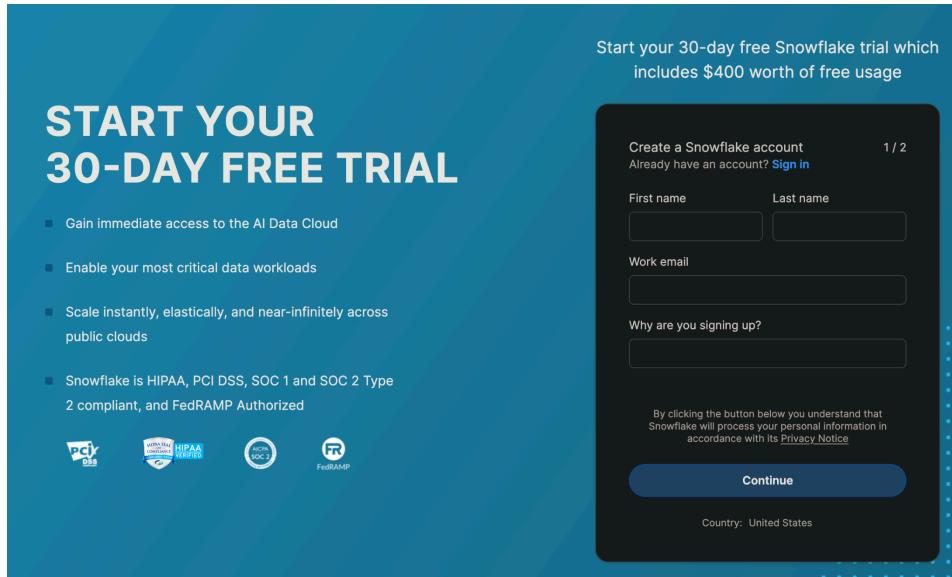
Table of Contents

Module 1: Setting Up.....	2
Signing-up for Snowflake Trail Account.....	2
Setting Up Account.....	5
Showcase setup.sql.....	5
Creating and Executing a SQL Files in Snowflake Workspace.....	6
Module 2: Dynamic Tables.....	10
Creating Dynamic Tables.....	10
Module 3: Chaining Dynamic Tables.....	14
Creating a Fact Table.....	14
Visualize the Pipeline.....	16
Module 4: Pipeline Monitoring.....	17
Showcase Pipeline Monitoring Techniques.....	17
Pipeline Management in Snowsight.....	19
Module 5: Building Intelligence.....	20
Module 6: Snowflake Badge.....	24
Setup Autograder.....	24
Run Autograding Scripts.....	24

Module 1: Setting Up

Signing-up for Snowflake Trail Account

1. Visit mlh.link/snowflake-de-trial



Start your 30-day free Snowflake trial which includes \$400 worth of free usage

CREATE A SNOWFLAKE ACCOUNT 1 / 2

Already have an account? [Sign in](#)

First name: Last name:

Work email:

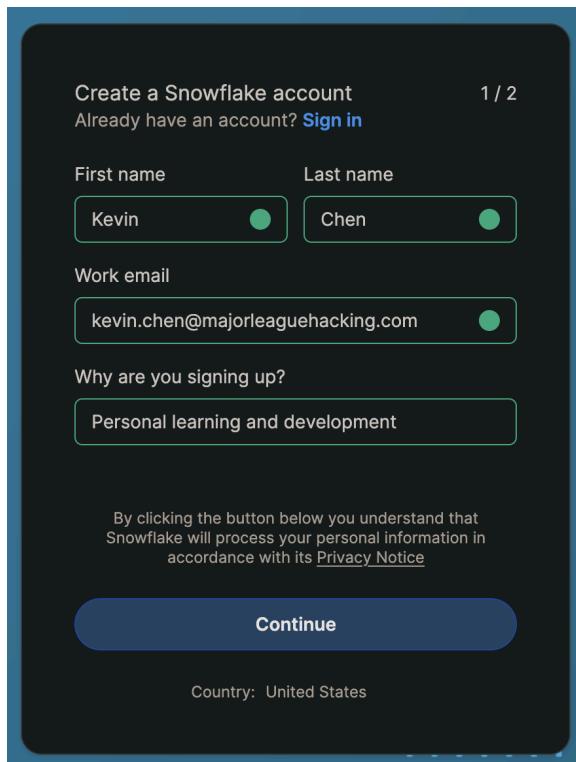
Why are you signing up?

By clicking the button below you understand that Snowflake will process your personal information in accordance with its [Privacy Notice](#)

Continue

Country: United States

2. Fill out all the information and select "Personal learning and development"



Create a Snowflake account 1 / 2

Already have an account? [Sign in](#)

First name: Kevin Last name: Chen

Work email: kevin.chen@majorleaguehacking.com

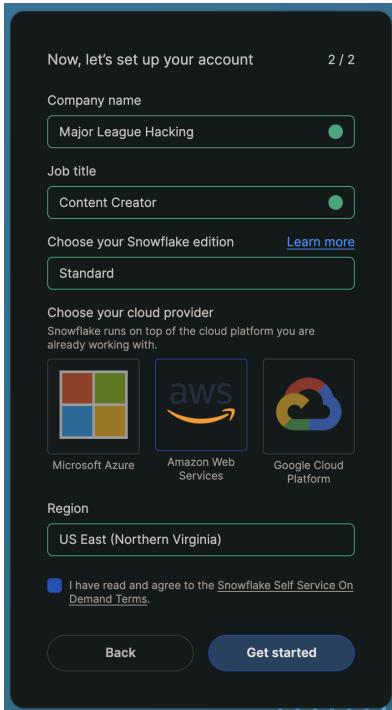
Why are you signing up?
 Personal learning and development

By clicking the button below you understand that Snowflake will process your personal information in accordance with its [Privacy Notice](#)

Continue

Country: United States

3. On the next page, enter your personal details (company name and role can be fictional). Choose the Standard edition, as it includes all AI/ML features at \$2/credit, so your trial credits will stretch further, but any edition will work for this workshop. You should see your cloud provider and region pre-selected for you, with the value of AWS and US West (Oregon). If for any reason you do not see that, select AWS and US West (Oregon) manually.



4. Answer the next two optional pages if desired
5. Check for an email and activate your account!

Check your inbox!
An email to activate your account has been sent to kevin.chen@majorleaguehacking.com. It may take a few minutes to arrive.
Meanwhile, here are a few resources to check out:

- Get started with [Snowflake documentation](#)
- Sign up for a free, instructor-led [Virtual Hands-On Lab](#)
- Explore industry-specific user cases and walkthroughs in our [Solutions Center](#)

Activate your Snowflake account ▶ MLH x

Snowflake Computing <no-reply@snowflake.net>
to kevin.chen ▾

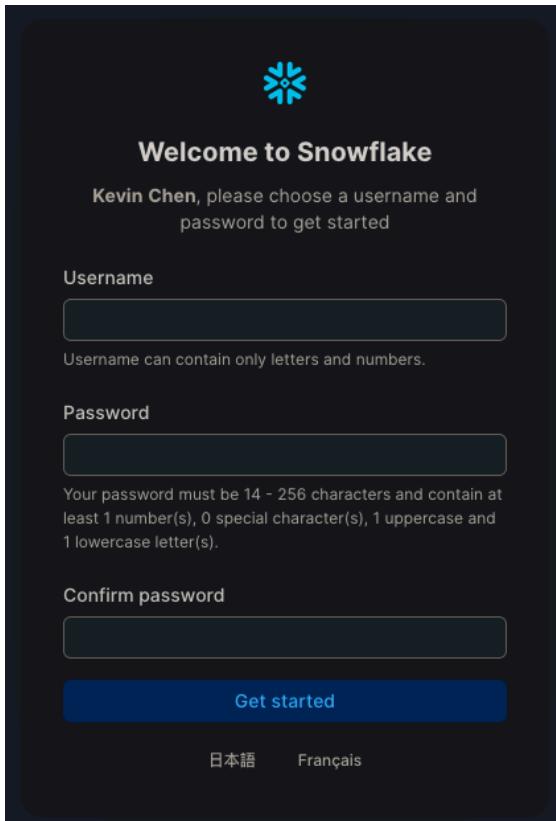
Congratulations on getting started with Snowflake! Click the button below to activate your account.

[CLICK TO ACTIVATE](#)

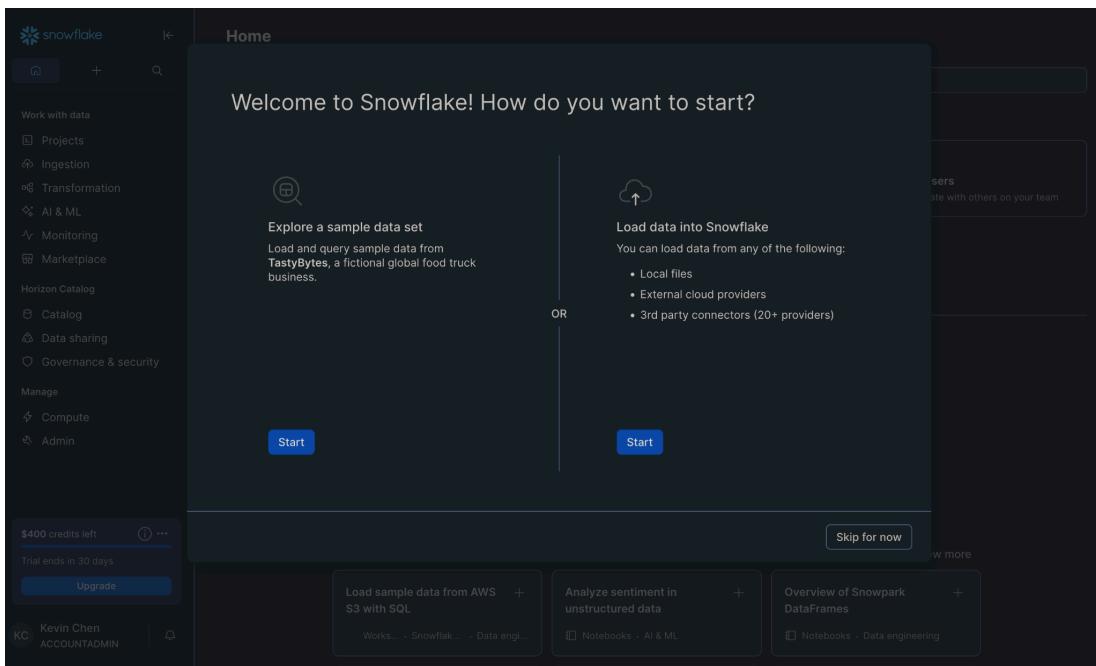
This activation link is temporary and will expire in 72 hours.

Save this for later
Once you activate your account, you can access it at
<https://apchri0-vac8103.snowflakecomputing.com/console/login>

6. Set up your user account for your new Snowflake account.



7. Perfect, leave that page up, and you are all setup to continue back to the slides!





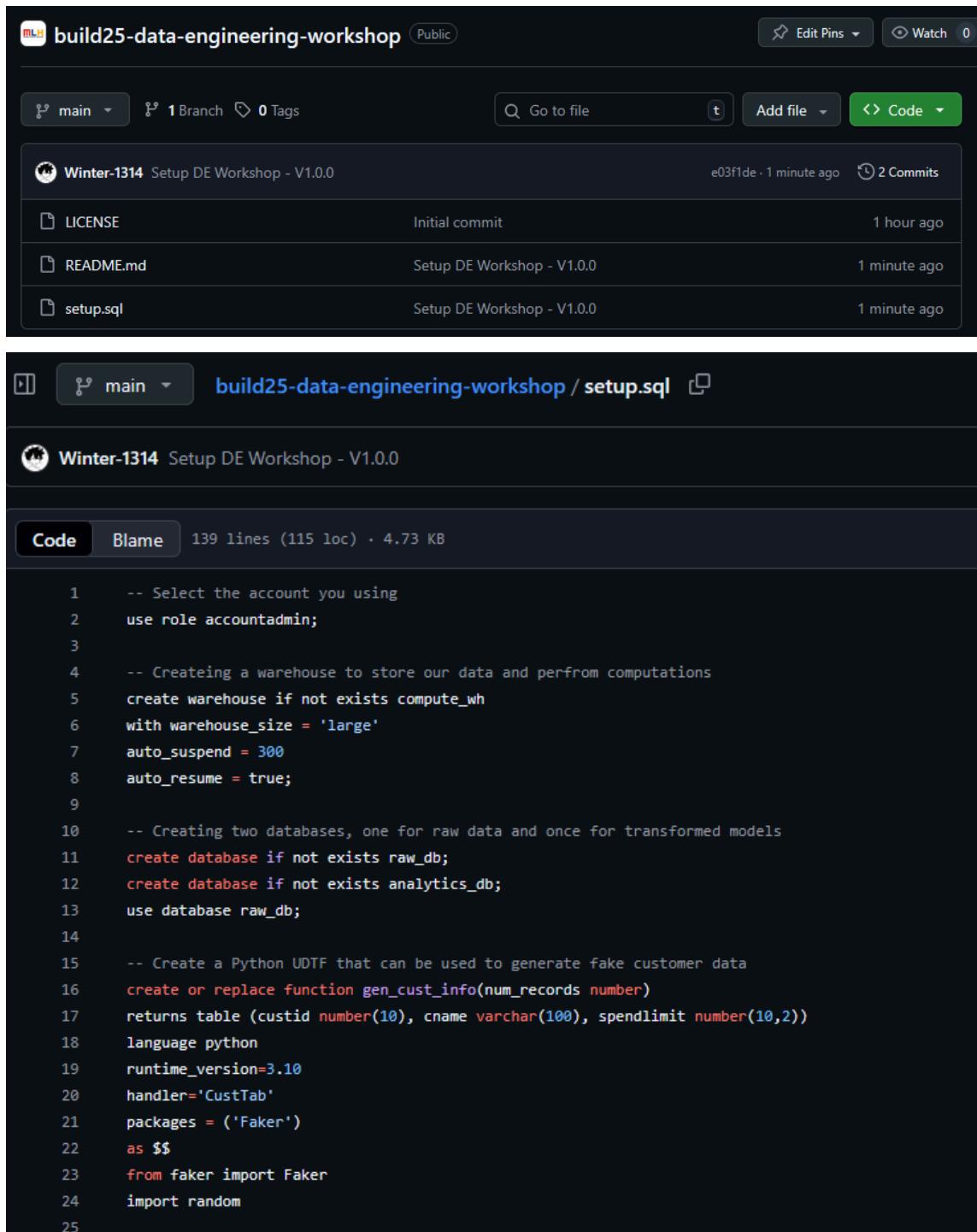
Loading in the Data

<https://github.com/MLH/build25-data-engineering-workshop>

Setting Up Account

Showcase `setup.sql`

1. Show the attendees the `setup.sql` script



The screenshot shows a GitHub repository named "build25-data-engineering-workshop". The repository has one branch ("main") and one tag ("0.0.0"). There are four files: LICENSE, README.md, and setup.sql. The setup.sql file was committed by "Winter-1314" at 1 minute ago. The code in setup.sql is as follows:

```

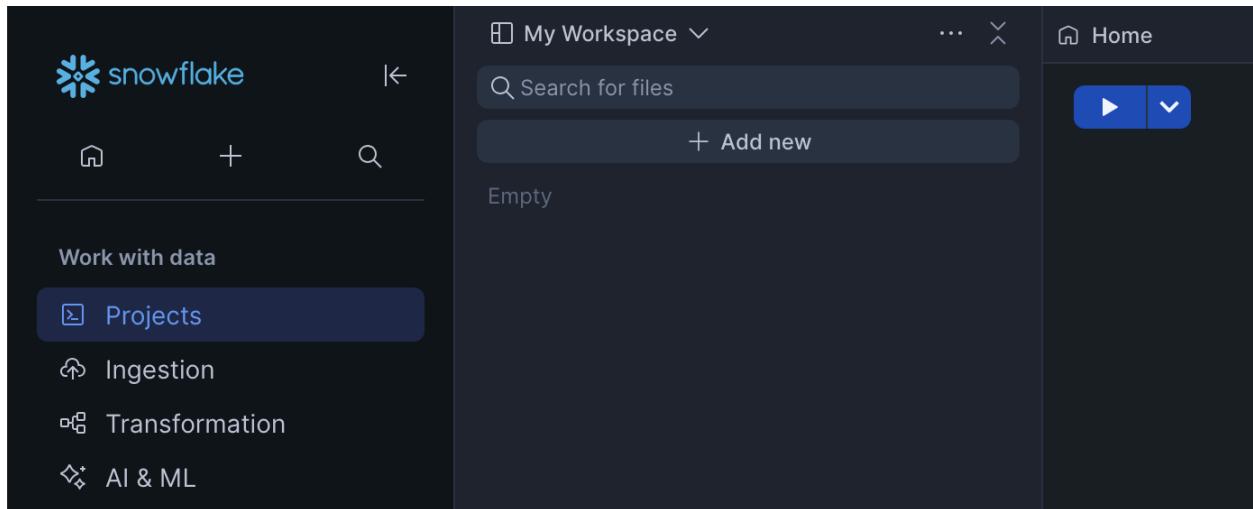
1 -- Select the account you using
2 use role accountadmin;
3
4 -- Creating a warehouse to store our data and perfrom computations
5 create warehouse if not exists compute_wh
6 with warehouse_size = 'large'
7 auto_suspend = 300
8 auto_resume = true;
9
10 -- Creating two databases, one for raw data and once for transformed models
11 create database if not exists raw_db;
12 create database if not exists analytics_db;
13 use database raw_db;
14
15 -- Create a Python UDTF that can be used to generate fake customer data
16 create or replace function gen_cust_info(num_records number)
17 returns table (custid number(10), cname varchar(100), spendlimit number(10,2))
18 language python
19 runtime_version=3.10
20 handler='CustTab'
21 packages = ('Faker')
22 as $$
23 from faker import Faker
24 import random
25

```

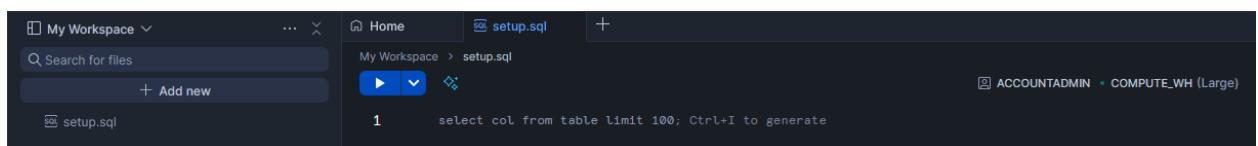
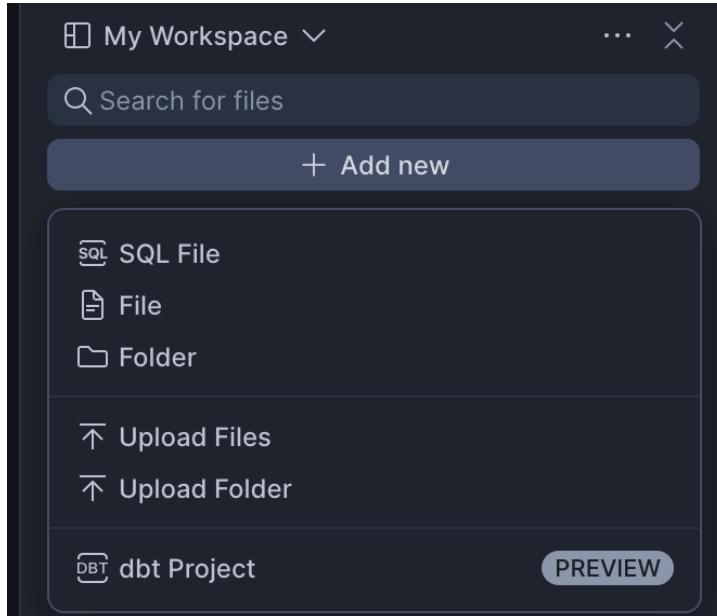
2. Scroll through and explain what the script is doing.
 - a. 1: Switch to the ACCOUNTADMIN role
 - b. 5-8: Create the Warehouse that will be used for compute/storage
 - c. 10-13: Create a database called RAW_DB for raw data and one called ANALYTICS_DB for transformed models
 - d. 16-122: Create three different User Defined Table Function that will generate mock data for a customer, products, and orders tables
 - e. 125-129: Create the three tables using the UDTF that was just created
 - f. 132-136: Allows you to run and validate the mock data in each of the three tables
 - g. 139: Display a setup completion confirmation message.

Creating and Executing a SQL Files in Snowflake Workspace

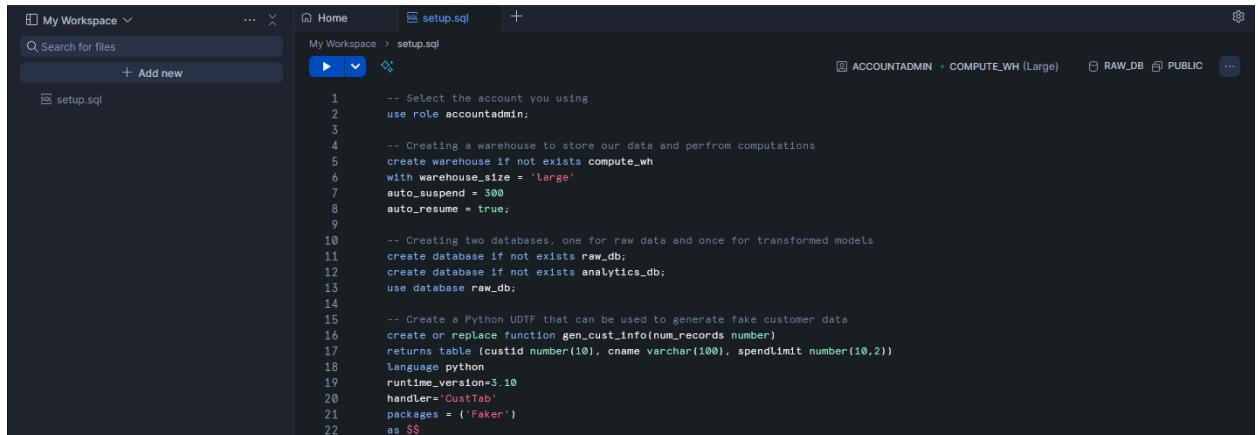
1. Go to "Projects" which should take you to "My Workspace"



- Click on “+ Add new” and select “SQL File” and name it `setup.sql`



- Navigate back to the GitHub repo, open `setup.sql`, copy its contents, and paste them into the SQL file you created in your workspace.

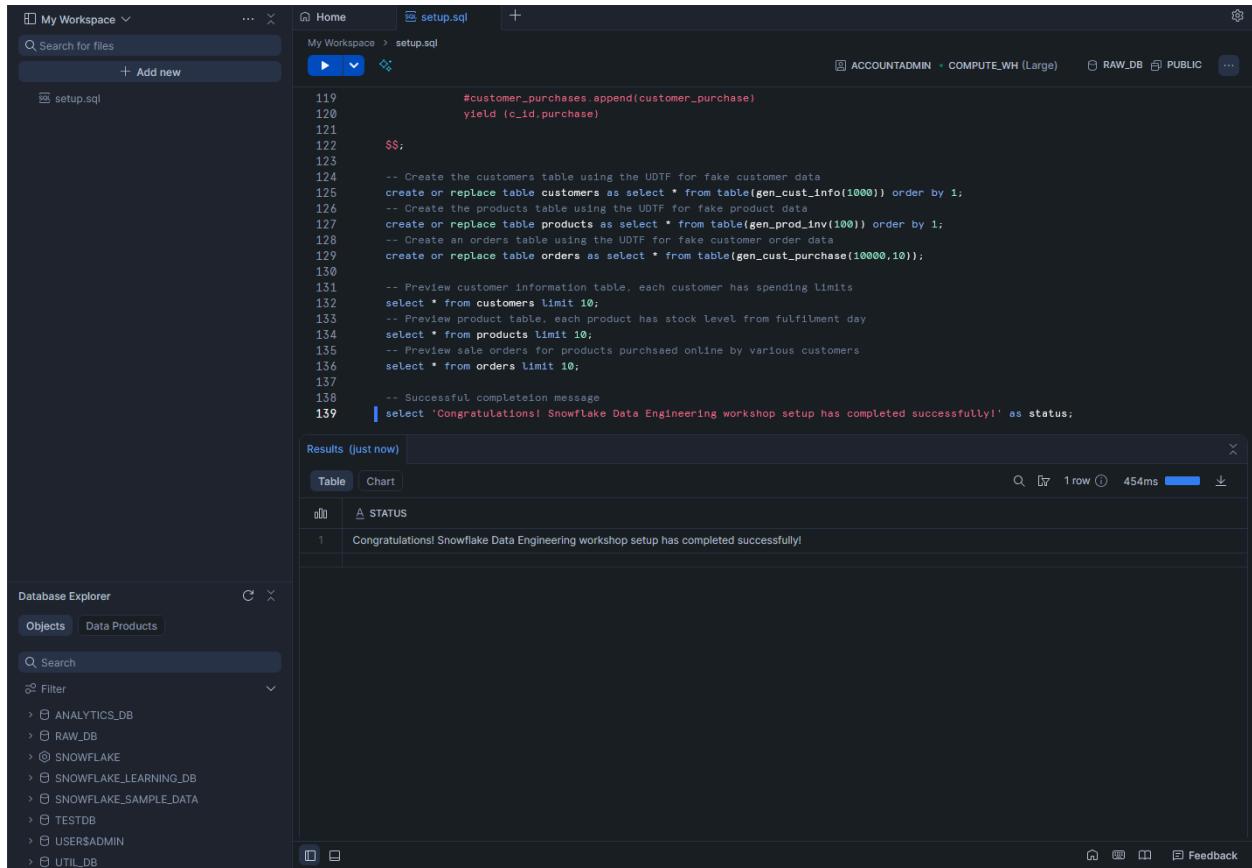


```

1 -- Select the account you using
2 use role accountadmin;
3
4 -- Creating a warehouse to store our data and perform computations
5 create warehouse if not exists compute_wh
6 with warehouse_size = 'Large'
7 auto_suspend = 300
8 auto_resume = true;
9
10 -- Creating two databases, one for raw data and one for transformed models
11 create database if not exists raw_db;
12 create database if not exists analytics_db;
13 use database raw_db;
14
15 -- Create a Python UDTF that can be used to generate fake customer data
16 create or replace function gen_cust_info(num_records number)
17 returns table (custid number(10), cname varchar(100), spendLimit number(10,2))
18 language python
19 runtime_version=3.10
20 handler='CustTab'
21 packages = ('Faker')
22 as $$
```

- If you haven't yet, briefly explain the purpose of `setup.sql`: it automates role, warehouse, database, UDTF, and table creation. Click “Run All” to execute the entire script. You should see: “Congratulations! Snowflake Data Engineering workshop

setup has completed successfully!"



The screenshot shows the Snowflake UI interface. At the top, there's a navigation bar with 'My Workspace', 'Home', and a file named 'setup.sql'. Below the navigation is a search bar and a '+' button for adding new files. The main area displays the 'setup.sql' script content:

```

119      #customer_purchases.append(customer_purchase)
120      yield (c_id,purchase)
121
122      $$;
123
124      -- Create the customers table using the UDTF for fake customer data
125      create or replace table customers as select * from table(gen_cust_info(1000)) order by 1;
126      -- Create the products table using the UDTF for fake product data
127      create or replace table products as select * from table(gen_prod_inv(100)) order by 1;
128      -- Create an orders table using the UDTF for fake customer order data
129      create or replace table orders as select * from table(gen_cust_purchase(10000,10));
130
131      -- Preview customer information table, each customer has spending limits
132      select * from customers limit 10;
133      -- Preview product table, each product has stock level from fulfilment day
134      select * from products limit 10;
135      -- Preview sale orders for products purchased online by various customers
136      select * from orders limit 10;
137
138      -- Successful completion message
139      | select 'Congratulations! Snowflake Data Engineering workshop setup has completed successfully!' as status;

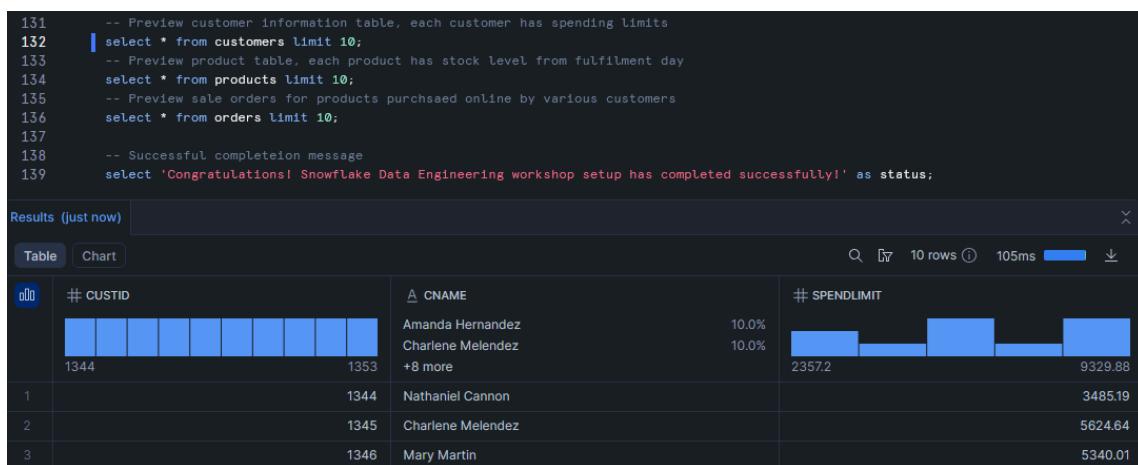
```

Below the script is a results table titled 'Results (just now)'. It has two tabs: 'Table' (selected) and 'Chart'. The 'Table' tab shows a single row with the status message. The 'Chart' tab shows a bar chart of spending limits for customers.

STATUS	SPENDLIMIT
Congratulations! Snowflake Data Engineering workshop setup has completed successfully!	9329.88

At the bottom left, there's a 'Database Explorer' sidebar with a tree view of databases like ANALYTICS_DB, RAW_DB, SNOWFLAKE, etc. On the right, there are various UI buttons and a feedback link.

5. Go back and run lines 132, 134, and 136 to preview all the data that was created for each of the three tables: CUSTOMERS, PRODUCTS, and ORDERS



The screenshot shows the results of running the previewed queries. The results table has three tabs: 'Table' (selected), 'Chart', and 'CSV'. The 'Table' tab shows the previewed data for each table:

# CUSTID		CNAME	SPENDLIMIT
1344	1344	Amanda Hernandez	10.0%
1344	1344	Charlene Melendez	10.0%
1345	1345	+8 more	2357.2
1346	1346	Nathaniel Cannon	3485.19
1346	1346	Charlene Melendez	5624.64
1346	1346	Mary Martin	5340.01

The 'Chart' tab shows a stacked bar chart of the spending limits for the top 8 customers.

```

131      -- Preview customer information table, each customer has spending limits
132      select * from customers limit 10;
133      -- Preview product table, each product has stock level from fulfilment day
134      | select * from products limit 10;
135      -- Preview sale orders for products purchased online by various customers
136      | select * from orders limit 10;
137
138      -- Successful completion message
139      | select 'Congratulations! Snowflake Data Engineering workshop setup has completed successfully!' as status;

```

Results (just now)

Table Chart

Q 10 rows 68ms

PID	PNAME	STOCK	STOCKDATE
114	Customizable actuating framework	502	8/14/2025
125	Extended fault-tolerant encoding	946	10/21/2025
+8 more			
1	Profit-focused radical encoding	770.00	2025-09-16
2	Reactive radical forecast	660.00	2025-10-02
3	Re-contextualized contextually-based budgetary manager	887.00	2025-10-22

```

131      -- Preview customer information table, each customer has spending limits
132      select * from customers limit 10;
133      -- Preview product table, each product has stock level from fulfilment day
134      | select * from products limit 10;
135      -- Preview sale orders for products purchased online by various customers
136      | select * from orders limit 10;
137
138      -- Successful completion message
139      | select 'Congratulations! Snowflake Data Engineering workshop setup has completed successfully!' as status;

```

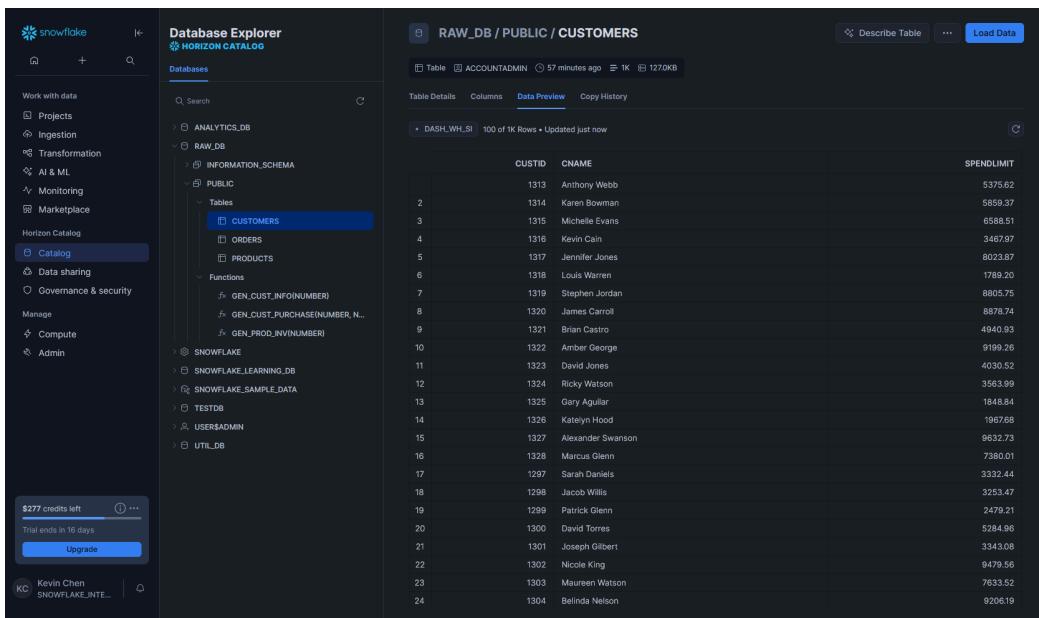
Results (just now)

Table Chart

Q 10 rows 51ms

CUSTID	PURCHASE
1190	100% filled
1803	{ "prodid": 143, "purchase_amount": 8.33460000000000e+02, "purchase_date": "2025-10-23", "quantity": 2 }
1953	{ "prodid": 154, "purchase_amount": 1.21730000000000e+02, "purchase_date": "2025-10-17", "quantity": 5 }
1630	{ "prodid": 168, "purchase_amount": 9.51730000000000e+02, "purchase_date": "2025-10-25", "quantity": 3 }

- Finally, verify the script ran successfully by going to "Catalog" and checking for "RAW_DB/PUBLIC" DB which should list three tables and three functions populated.

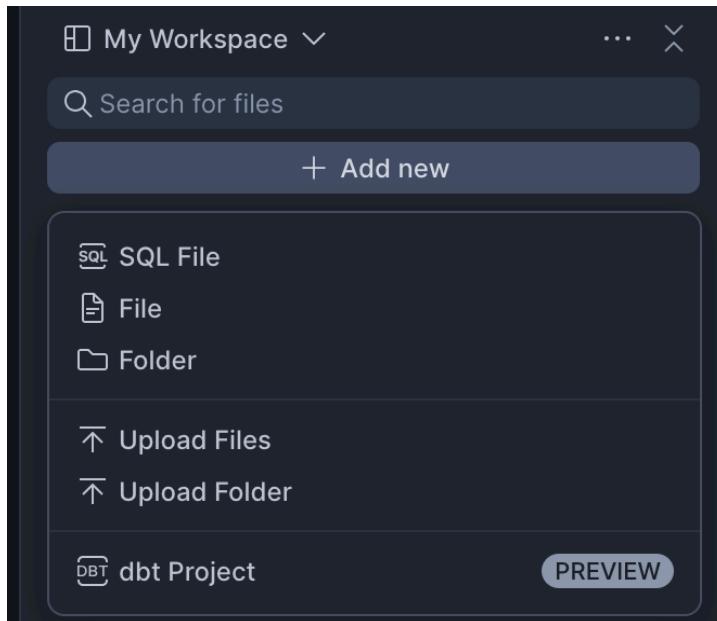


The screenshot shows the Snowflake Database Explorer interface. On the left, the sidebar includes sections for Work with data, Catalog, and Admin. The Catalog section is currently selected. In the center, the Database Explorer shows the HORIZON CATALOG. Under Databases, there is a tree view with ANALYTICS_DB, RAW_DB (selected), INFORMATION_SCHEMA, and PUBLIC. Under PUBLIC, there are Tables (CUSTOMERS, ORDERS, PRODUCTS) and Functions (GEN_CUST_INFO(NUMLIMIT), GEN_CUST_PURCHASE(NUMLIMIT, N_), GEN_PROD_INV(NUMLIMIT)). The CUSTOMERS table is selected. On the right, the details for the CUSTOMERS table are shown, including a preview of 100 rows of data. The columns are CUSTID, CNAME, and SPENDLIMIT. The data includes rows for Anthony Webb, Karen Bowman, Michelle Evans, Kevin Cain, Jennifer Jones, Louis Warren, Stephen Jordan, James Carroll, Brian Castro, Amber George, David Jones, Ricky Watson, Gary Aguilar, Katelyn Hood, Alexander Swanson, Marcus Glenn, Sarah Daniels, Jacob Wills, Patrick Glenn, David Torres, Joseph Gilbert, Nicole King, Maureen Watson, and Belinda Nelson.

Module 2: Dynamic Tables

Creating Dynamic Tables

1. Click on “+ Add new” and select “SQL File” and name it `create-dt.sql`



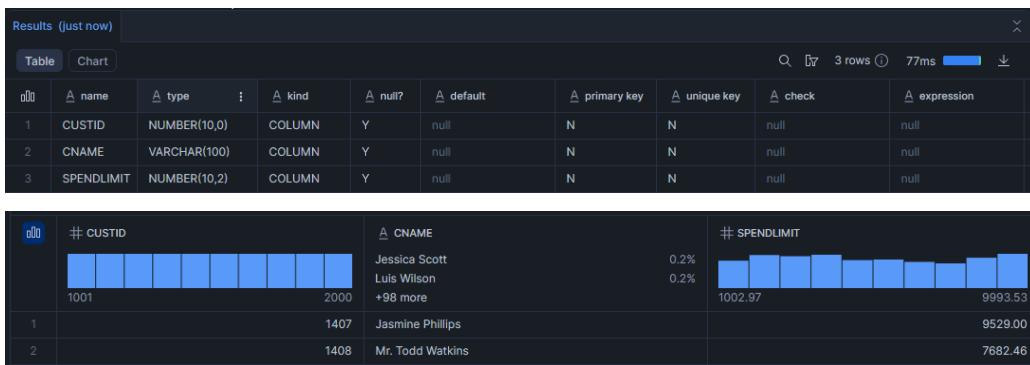
2. Type in the following lines to select the account, warehouse, and database you are using.

```
a. use role accountadmin;
b. use warehouse compute_wh;
c. use database analytics_db;
```

3. Type in the following lines to display what type of data and example data in the customers table. The goal here is to identify columns that we can perform light transformations like renaming columns and basic type casting.

```
a. desc table raw_db.public.customers;
b. select * from raw_db.public.customers;
```

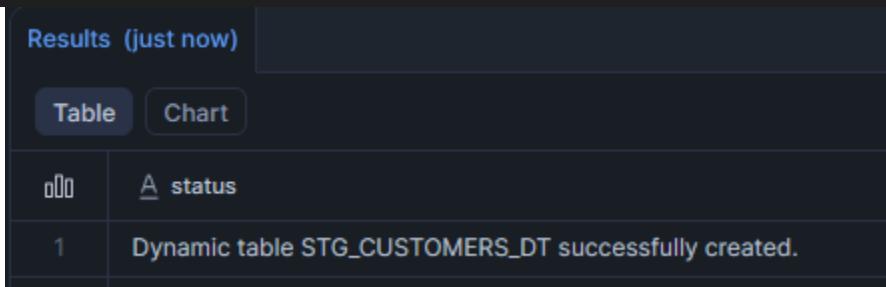
C.



#	A name	A type	A kind	A null?	A default	A primary key	A unique key	A check	A expression
1	CUSTID	NUMBER(10,0)	COLUMN	Y	null	N	N	null	null
2	CNAME	VARCHAR(100)	COLUMN	Y	null	N	N	null	null
3	SPENDLIMIT	NUMBER(10,2)	COLUMN	Y	null	N	N	null	null

4. Type in the following lines to create a Dynamic Table for CUSTOMERS table with the transformations that we selected.

```
a. create or replace dynamic table stg_customers_dt
b. target_lag=downstream
c. warehouse=compute_wh
d. as select
e.     custid as customer_id,
f.     cname as customer_name,
g.     cast(spendlimit as float) as spend_limit
h. from raw_db.public.customers;
```



#	A status
1	Dynamic table STG_CUSTOMERS_DT successfully created.

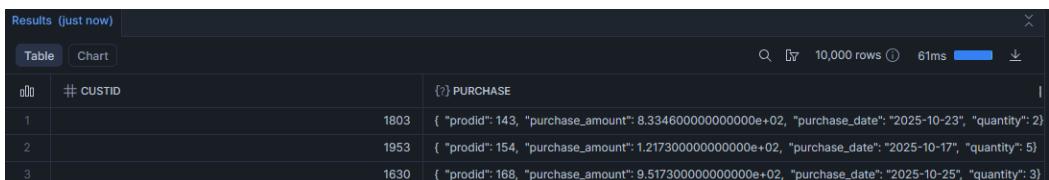
5. Type in the following lines to display what type of data and example data in the orders table. For this table, you will notice that the purchase column holds a JSON value. This means when we are doing the transformations, we can pull those values into their own columns

```
a. desc table raw_db.public.orders;
b. select* from raw_db.public.orders;
```

C.



id	A name	A type	A kind	A null?	A default	A primary key	A unique key	A check	A expression
1	CUSTID	NUMBER(10,0)	COLUMN	Y	null	N	N	null	null
2	PURCHASE	VARIANT	COLUMN	Y	null	N	N	null	null

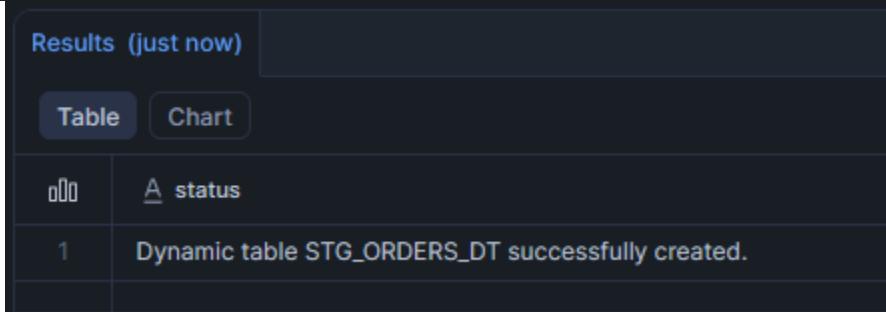


id	# CUSTID	PURCHASE
1		{ "prodid": 143, "purchase_amount": 8.33460000000000e+02, "purchase_date": "2025-10-23", "quantity": 2 }
2		{ "prodid": 154, "purchase_amount": 1.21730000000000e+02, "purchase_date": "2025-10-17", "quantity": 5 }
3		{ "prodid": 168, "purchase_amount": 9.51730000000000e+02, "purchase_date": "2025-10-25", "quantity": 3 }

6. Type in the following lines to create a dynamic table for orders table. Note the unpacking of the JSON. The following format is used

COLUMN:"VAR_NAME":TYPE_CASE as NEW_COL_NAME

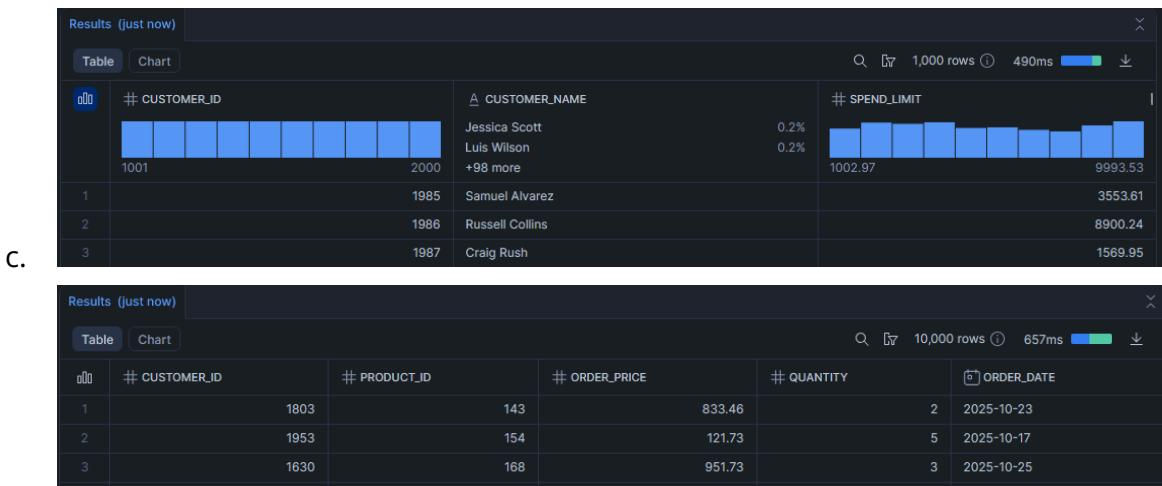
```
a. create or replace dynamic table stg_orders_dt
b. target_lag=downstream
c. warehouse=compute_wh
d. as select
e.     custid as customer_id,
f.     purchase:"prodid":number(5) as product_id,
g.     purchase:"purchase_amount":float(10) as order_price,
h.     purchase:"quantity":number(5) as quantity,
i.     purchase:"purchase_date":date as order_date
j. from raw_db.public.orders;
```



id	A status
1	Dynamic table STG_ORDERS_DT successfully created.

7. Type in the following lines to query the new Dynamic Tables that you created. You should see the new columns.

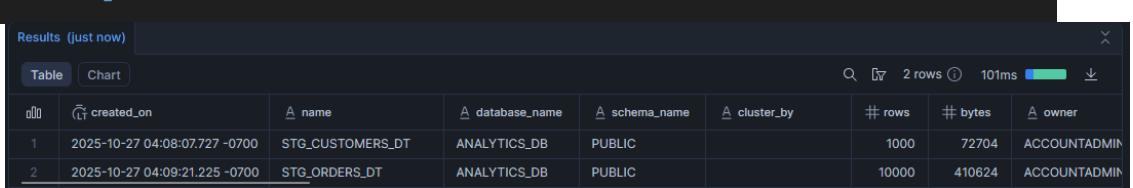
```
a. select * from analytics_db.public.stg_customers_dt;
b. select * from analytics_db.public.stg_orders_dt;
```



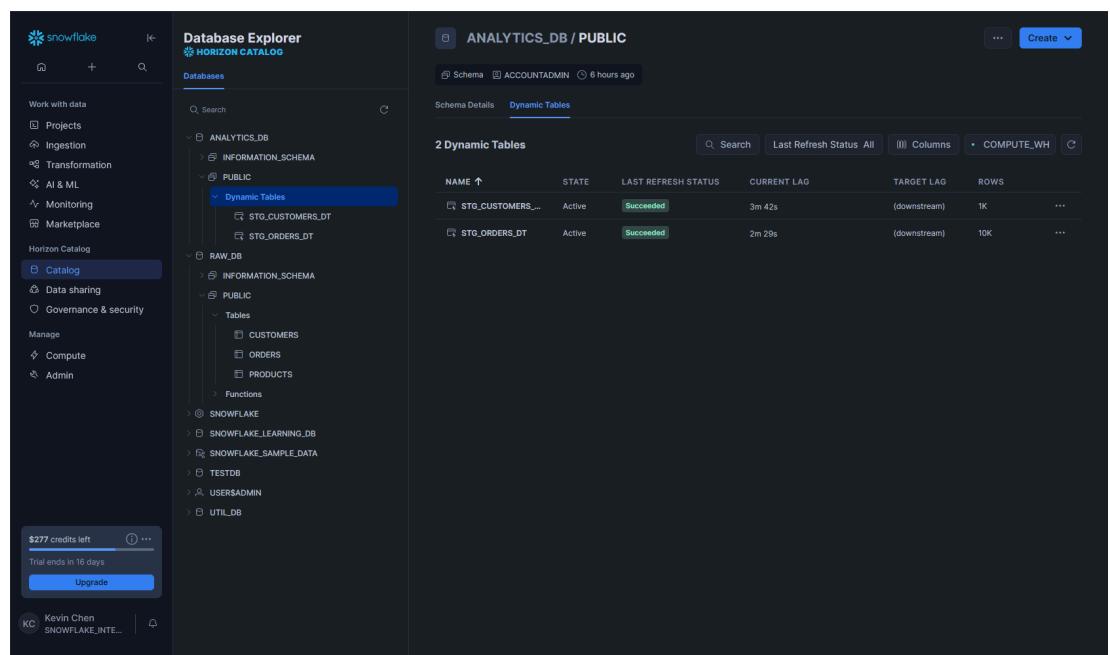
8. Type in the following lines to displays all the Dynamic Tables that we have created. Scroll to the right and find TARGET_LAG can make sure to call out that this how often the Dynamic Tables will refresh its data. In our case, it's set to DOWNSTREAM, which means whenever the downstream table is changed, our Dynamic Tables will refresh. You can also go to "Catalog" and see the two new Dynamic Tables.

a. `show dynamic tables;`

b.



The screenshot shows the Snowflake Catalog interface. It lists two dynamic tables: 'STG_CUSTOMERS_DT' and 'STG_ORDERS_DT', both located in the 'ANALYTICS_DB' database under the 'PUBLIC' schema. The 'Dynamic Tables' section in the catalog sidebar is also visible.

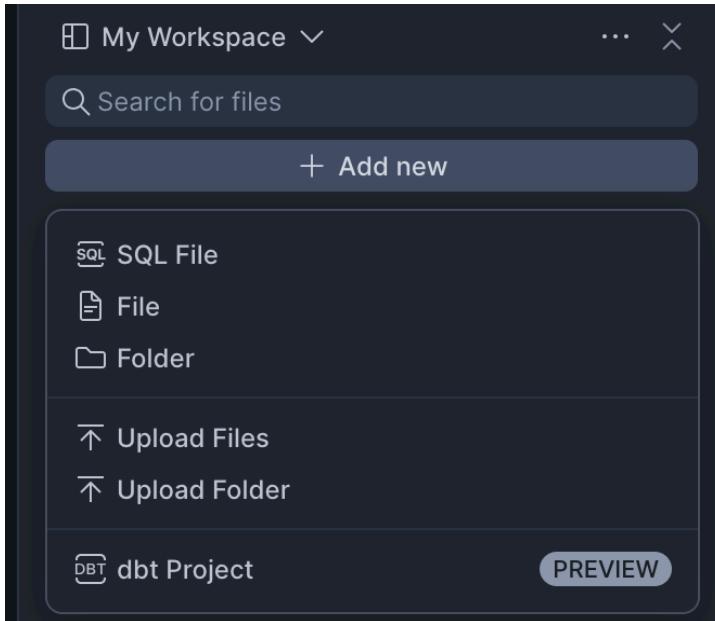


The screenshot shows the Snowflake Database Explorer. On the left, the 'Catalog' section is selected, showing the 'ANALYTICS_DB' schema with its tables ('CUSTOMERS', 'ORDERS', 'PRODUCTS') and dynamic tables ('STG_CUSTOMERS_DT', 'STG_ORDERS_DT'). On the right, the 'ANALYTICS_DB / PUBLIC' schema details are shown, specifically the 'Dynamic Tables' section, which lists the two dynamic tables with their current status and refresh information.

Module 3: Chaining Dynamic Tables

Creating a Fact Table

1. Click on “+ Add new” and select “SQL File” and name it chaining-dt.sql



2. Type in the following lines to select the account, warehouse, and database you are using.

```

a. use role accountadmin;
b. use warehouse compute_wh;
c. use database analytics_db;
```

3. Type in the following lines to check the values in each Dynamic Tables. You are doing this so you can decide what values you want to carry over into your Fact Table.

```

a. select * from analytics_db.public.stg_customers_dt;
b. select * from analytics_db.public.stg_orders_dt;
```

4. Type in the following lines to create a fact dynamic table for customer orders. We are going to take the `customer_id` and `customer_name` from the `STG_CUSTOMERS_DT` Dynamic Table and then the `product_id`, `order_price`, `quantity`, and `order_date` from the `STG_ORDERS_DT` Dynamic Table. Make sure to talk about how all you needed to do is simply reference the upstream Dynamic Tables by name and Snowflake automatically discovers the dependency needed.

```

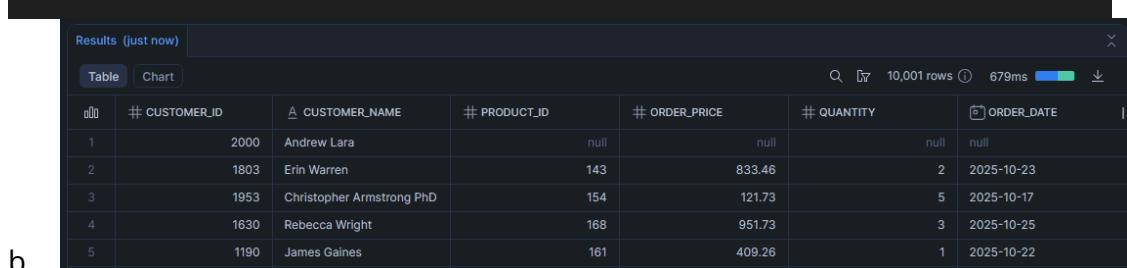
a. create or replace dynamic table fct_customer_orders_dt
b.      target_lag=downstream
c.      warehouse=compute_wh
d.      as select
e.          c.customer_id,
f.          c.customer_name,
g.          o.product_id,
h.          o.order_price,
i.          o.quantity,
j.          o.order_date
k.      from stg_customers_dt c
l.      left join stg_orders_dt o
m.          on c.customer_id = o.customer_id;

```

5. Type in the following lines to query the new Fact Model Dynamic Table that you created.

a. `select * from analytics_db.public.fct_customer_orders_dt;`

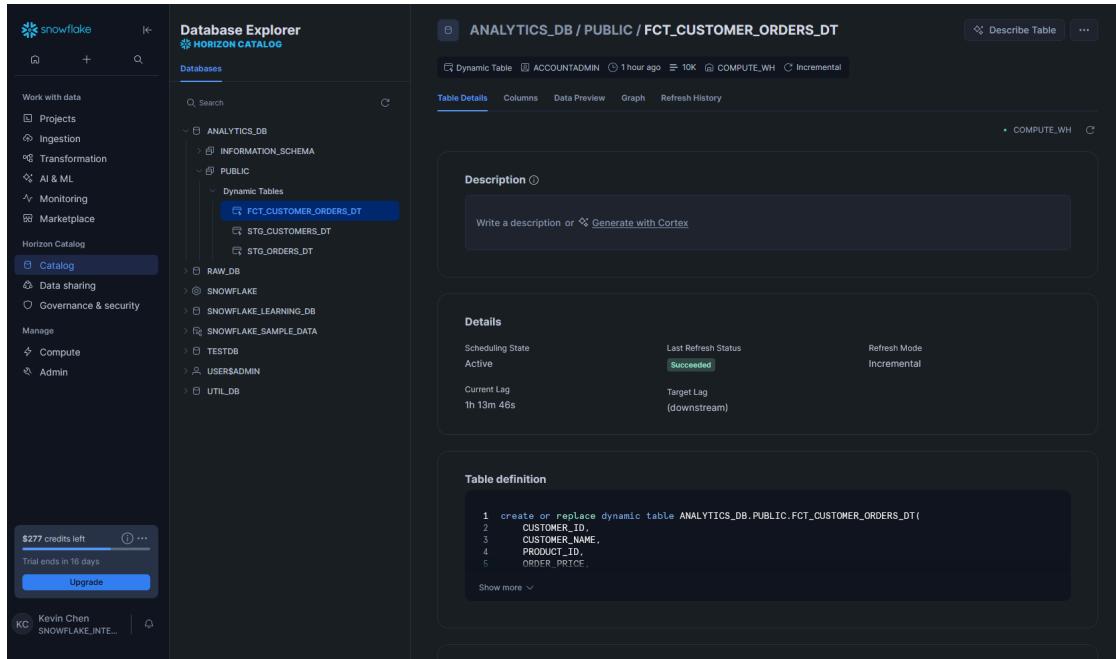
b.



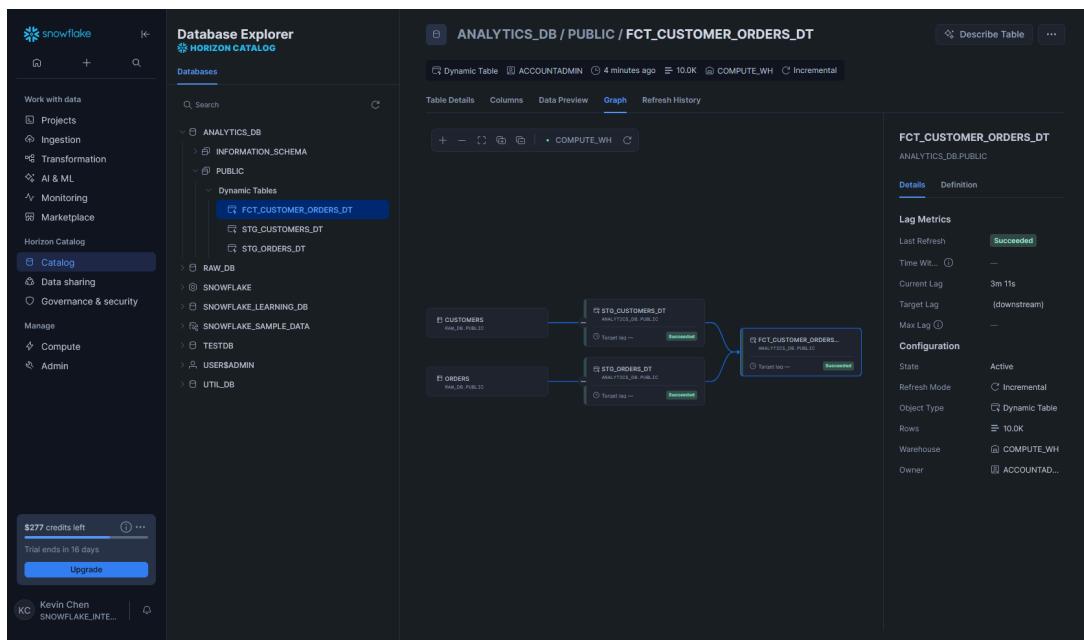
#	CUSTOMER_ID	CUSTOMER_NAME	PRODUCT_ID	ORDER_PRICE	QUANTITY	ORDER_DATE
1	2000	Andrew Lara	null	null	null	null
2	1803	Erin Warren	143	833.46	2	2025-10-23
3	1953	Christopher Armstrong PhD	154	121.73	5	2025-10-17
4	1630	Rebecca Wright	168	951.73	3	2025-10-25
5	1190	James Gaines	161	409.26	1	2025-10-22

Visualize the Pipeline

1. Navigate to “Catalog” and under “Dynamic Table” select FCT_CUSTOMER_ORDER_DT.



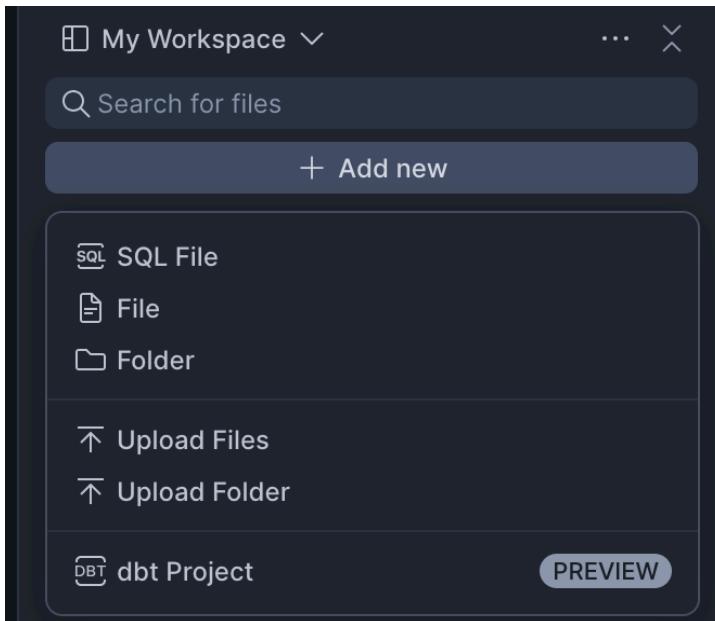
2. Click on “Graph”. You should now see a visual DAG (Directed Acyclic Graph) of the pipeline: Raw Tables -> Staging Dynamic Tables -> Fact Dynamic Table. If you need to click the + and zoom out.



Module 4: Pipeline Monitoring

Showcase Pipeline Monitoring Techniques

1. Click on “+ Add new” and select “SQL File” and name it pipeline.sql



2. Type in the following lines to select the account, warehouse, and database you are using.

```
a. use role accountadmin;
b. use warehouse compute_wh;
c. use database analytics_db;
```

3. Type in the following lines to displays all the Dynamic Tables that we have created. We just want to show the attendees that the current value for TARGET_LAG is DOWNSTREAM.

```
a. show dynamic tables;
```

4. Type in the following lines to adjust the freshness, meaning the TARGET_LAG for STG_ORDERS_DT. Explain that any downstream tables will automatically adjust to this new cadence, since they update whenever changes occur upstream.

```
a. alter dynamic table stg_orders_dt set target_lag = '5
minutes';
```

5. Type in the following lines to check for the altered Dynamic Table's new TARGET_LAG which now should say 5 minutes.

```
a. show dynamic tables;
```

6. Type in the following lines to monitoring pipeline health, this is how you inspect the history of refreshes, showing execution times, data changes, and potential errors. This is your built-in observability tool in SQL.

```
a. select * from
    table(information_schema.dynamic_table_refresh_history());
```

Results (just now)

	NAME	SCHEMA_NAME	DATABASE_NAME	STATE	STATE_CODE	STATE_MESSAGE	QUERY_ID
STG_ORDERS_DT	68.0%	PUBLIC	ANALYTICS_DB	SUCCE... 100.0%	SUCCESS	All values are null	01bffc15-0000-4214-00
STG_CUSTOMERS_DT	20.0%	100.0%	RAW_DB	1.3%	SUCCE... 100.0%		01bffc23-0000-4214-00
+1 more							+73 more
1	FCT_CUSTOMER_ORDERS_DT	PUBLIC	ANALYTICS_DB	SUCCEEDED	SUCCESS	null	01bffd21-0000-4231-00
2	STG_CUSTOMERS_DT	PUBLIC	ANALYTICS_DB	SUCCEEDED	SUCCESS	null	01bffd21-0000-4231-00
3	STG_ORDERS_DT	PUBLIC	ANALYTICS_DB	SUCCEEDED	SUCCESS	null	01bffd21-0000-4231-00
4	STG_ORDERS_DT	PUBLIC	ANALYTICS_DB	SUCCEEDED	SUCCESS	null	01bffd1d-0000-4214-00
5	STG_ORDERS_DT	PUBLIC	ANALYTICS_DB	SUCCEEDED	SUCCESS	null	01bffd1c-0000-41d4-00

7. Type in the following lines to query the Fact Dynamic Table and check for potential issues, we are looking to see if there is any null order, sometimes there is, sometimes there isn't. If there is, call it out, if not move to the next step but call out that there could be null for PRODUCT_ID if the user didn't purchase anything.

```
a. select * from analytics_db.public.fct_customer_orders_dt;
```

8. Type in the following lines to integrated data quality to remove null orders from the FCT_CUSTOMER_ORDERS_DT.

```
a. create or replace dynamic table fct_customer_orders_dt
b. target_lag=downstream
c. warehouse=compute_wh
d. as select
e.     c.customer_id,
f.     c.customer_name,
g.     o.product_id,
h.     o.order_price,
i.     o.quantity,
j.     o.order_date
k.     from stg_customers_dt c
l.     left join stg_orders_dt o
m.         on c.customer_id = o.customer_id
```

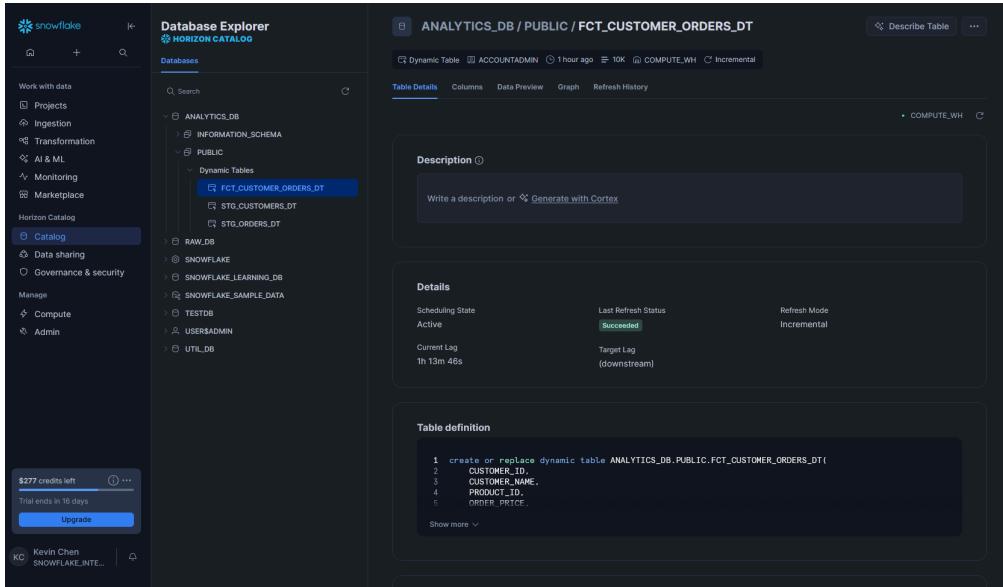
```
n.      where o.product_id is not null;
```

9. Type in the following lines to check if the data quality enforcement works

```
a. select * from analytics_db.public.fct_customer_orders_dt;
```

Pipeline Management in Snowsight

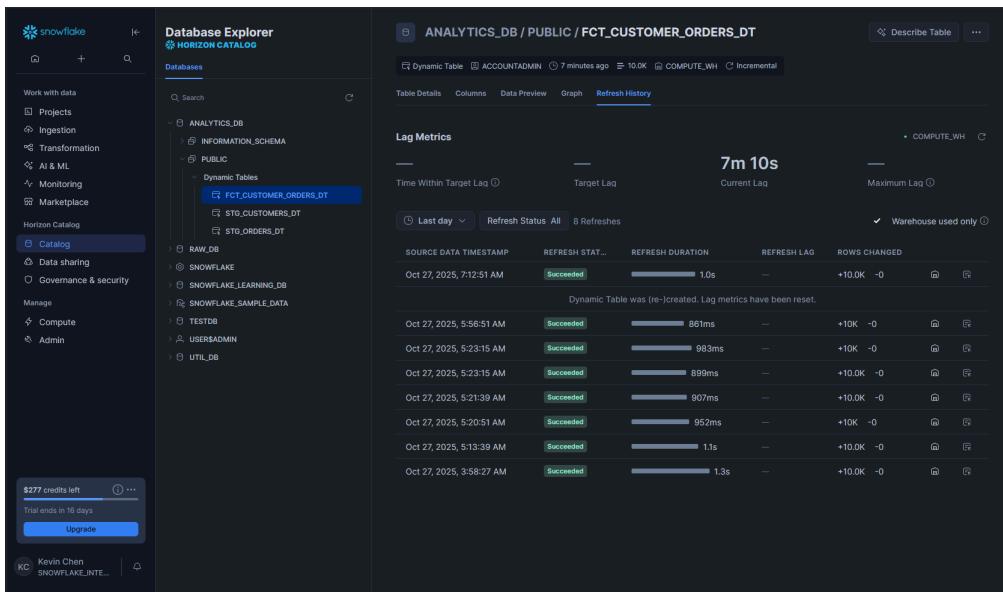
1. Navigate to “Catalog” and under “Dynamic Table” select FCT_CUSTOMER_ORDER_DT.



The screenshot shows the Snowsight interface. On the left, the Database Explorer displays the schema structure of the ANALYTICS_DB database, including the PUBLIC schema which contains the FCT_CUSTOMER_ORDER_DT dynamic table. This table is highlighted. The right side shows the detailed view for the FCT_CUSTOMER_ORDER_DT table, including its description, scheduling details (Active), and table definition. The table definition is as follows:

```
1 create or replace dynamic table ANALYTICS_DB.PUBLIC.FCT_CUSTOMER_ORDER_DT
2   CUSTOMER_ID,
3   CUSTOMER_NAME,
4   PRODUCT_ID,
5   ORDER_PRICE,
```

2. Click on “Refresh History”. You should now see a table of information on when the Dynamic Table was refreshed and the status of the refresh.



The screenshot shows the same Snowsight interface, but the Refresh History tab is selected in the top navigation bar. The main area displays a table of refresh history for the FCT_CUSTOMER_ORDER_DT table. The table includes columns for SOURCE DATA TIMESTAMP, REFRESH STATUS, REFRESH DURATION, REFRESH LAG, and ROWS CHANGED. The data shows several successful refreshes over a period of time, with the most recent refresh being at 7m 10s ago.

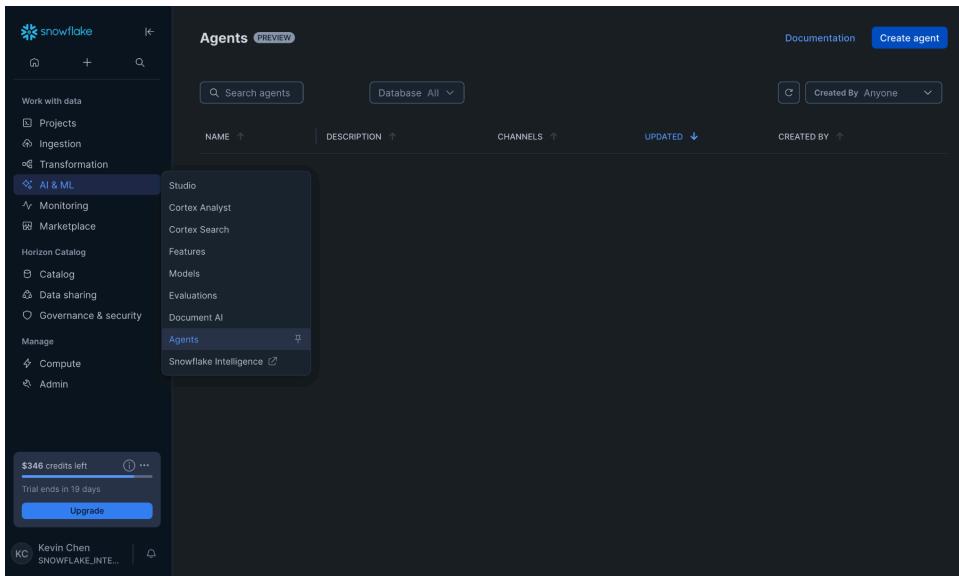
SOURCE DATA TIMESTAMP	REFRESH STATUS	REFRESH DURATION	REFRESH LAG	ROWS CHANGED
Oct 27, 2025, 7:12:51 AM	Succeeded	1.0s	—	+10.0K -0
Oct 27, 2025, 5:56:51 AM	Succeeded	861ms	—	+10K -0
Oct 27, 2025, 5:23:15 AM	Succeeded	983ms	—	+10K -0
Oct 27, 2025, 5:21:39 AM	Succeeded	899ms	—	+10.0K -0
Oct 27, 2025, 5:20:51 AM	Succeeded	952ms	—	+10K -0
Oct 27, 2025, 5:13:39 AM	Succeeded	1.1s	—	+10.0K -0
Oct 27, 2025, 3:58:27 AM	Succeeded	1.3s	—	+10.0K -0

Module 5: Snowflake Intelligence

1. Resources to explore Snowflake Intelligence

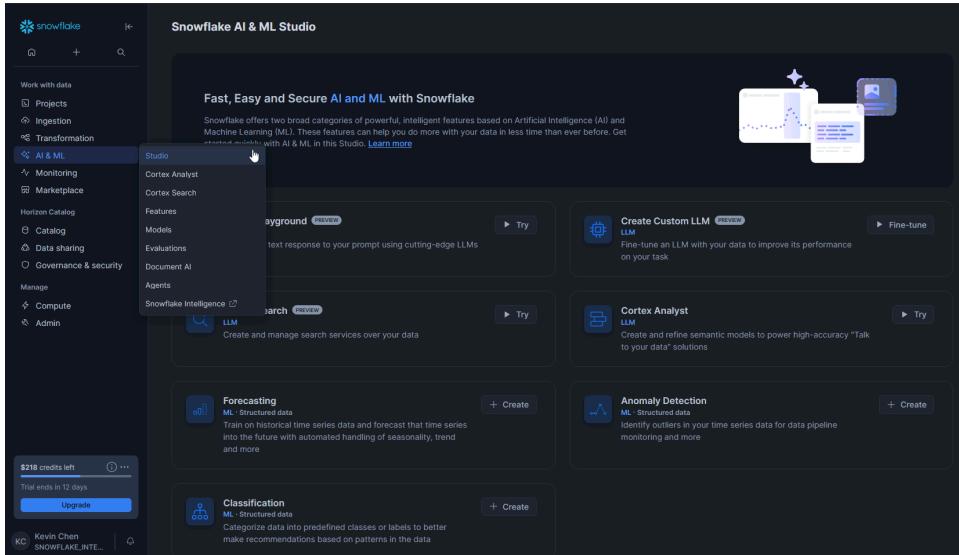
- a. [Snowflake Intelligence Overview](#)
- b. [Snowflake Cortex Agents](#)
- c. [Snowflake Cortex Analyst](#)
- d. [Snowflake Cortex Search](#)
- e. [Understanding Snowflake Cortex](#)
- f. [Tutorial: Getting Started with Snowflake Intelligence](#)

2. Show the attendees the “AI/ML” section



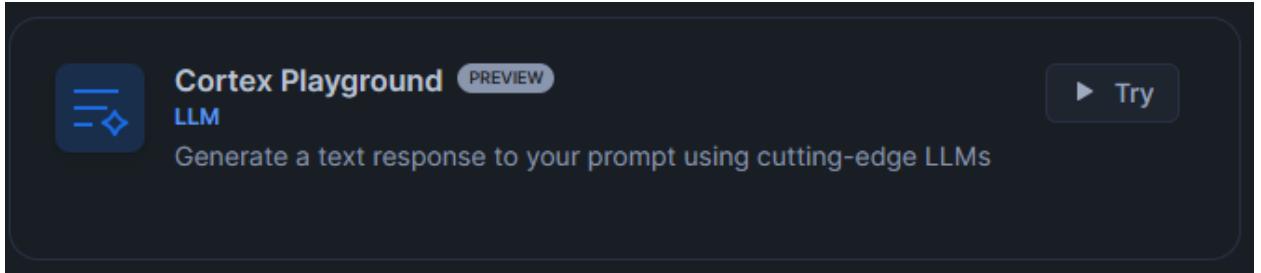
The screenshot shows the Snowflake AI & ML Agents page. The left sidebar has 'AI & ML' selected. The main area shows a table of agents with columns: NAME, DESCRIPTION, CHANNELS, UPDATED, and CREATED BY. The first row is 'Agents'.

3. Navigate to “Studio” where you can see the “Snowflake AI & ML Studio”

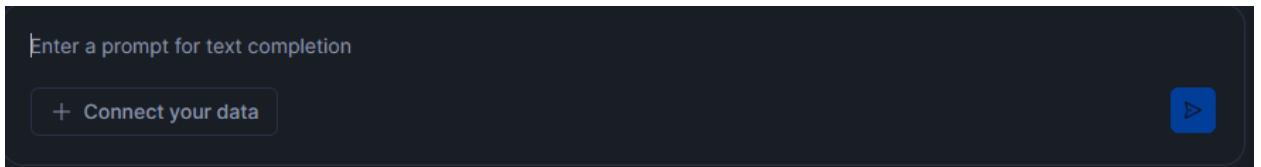


The screenshot shows the Snowflake AI & ML Studio page. The left sidebar has 'AI & ML' selected. The main area displays several AI and ML services: 'Fast, Easy and Secure AI and ML with Snowflake', 'aground', 'Create Custom LLM', 'Cortex Analyst', 'Forecasting', and 'Classification'.

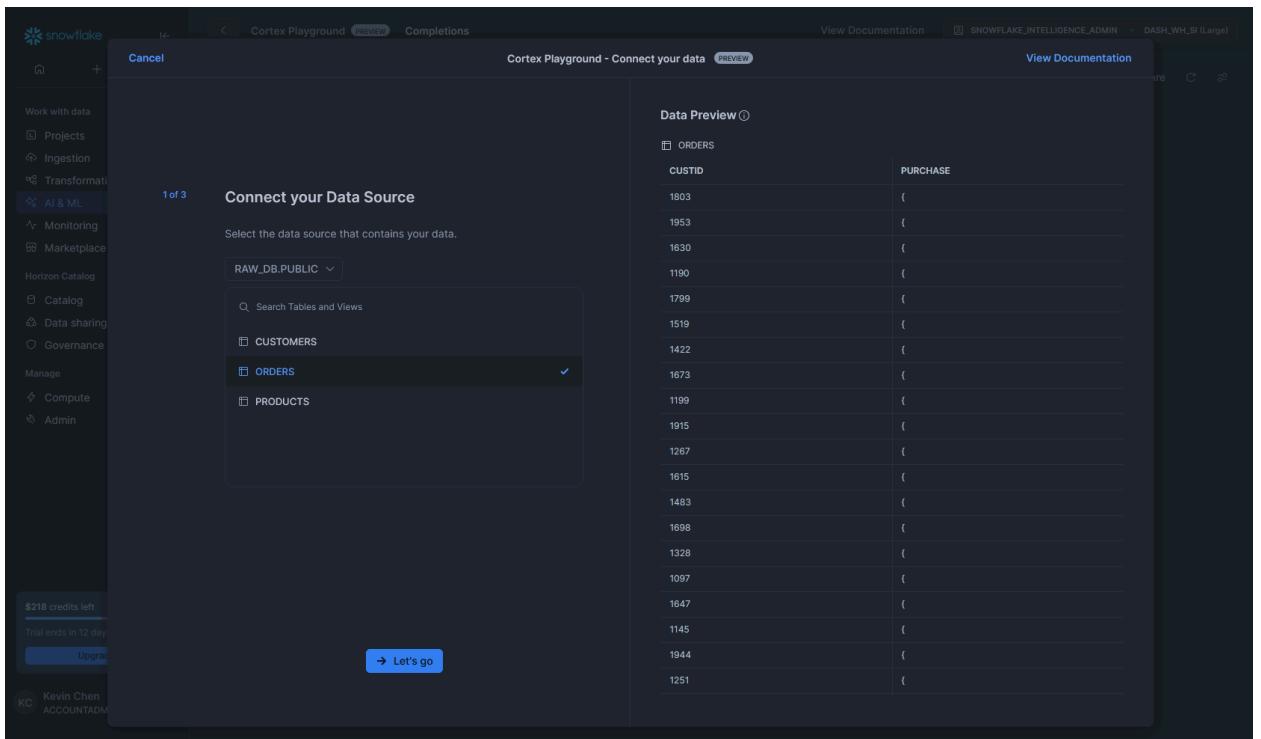
4. Select "Try" for Cortex Playground. Note that we will be testing using the Cortex Playground, which does not directly support Dynamic Tables. To use Dynamic Tables, you would need to leverage Cortex Search and create a custom Agent, which attendees can explore further if they're interested.



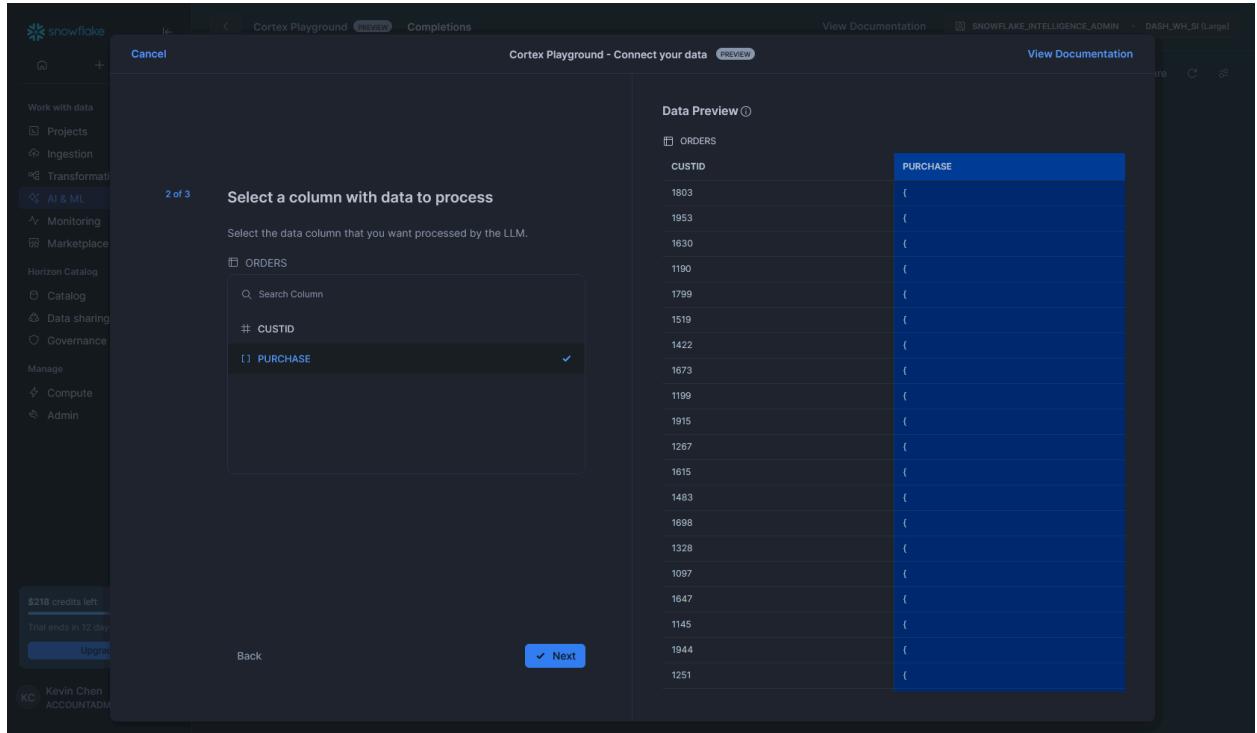
5. Select "Connect your data"



6. Select RAW_DB.PUBLIC.ORDERS for your Data Source, then click "Let's go"

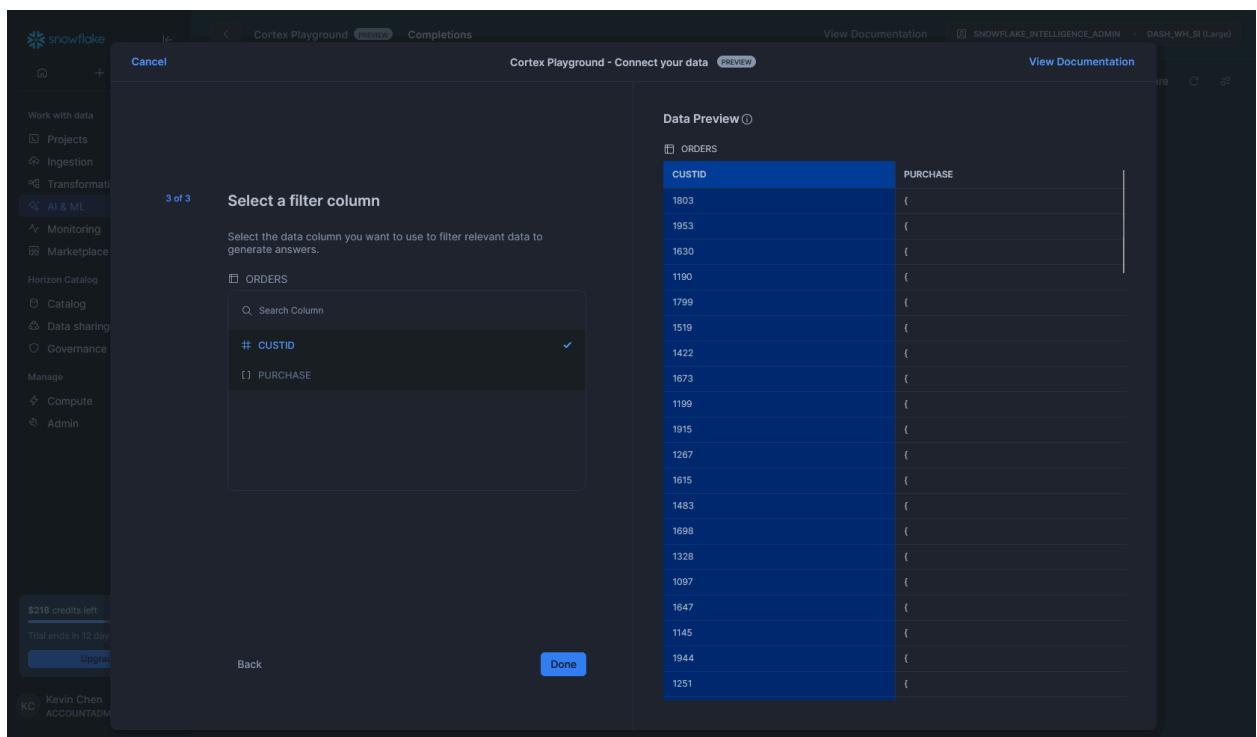


7. Select “PURCHASE” then click “Next”



The screenshot shows the Cortex Playground interface. On the left, there's a sidebar with options like Projects, Ingestion, Transformation, AI & ML, Monitoring, and Marketplace. The main area is titled "Select a column with data to process". It shows a table with columns "ORDERS" and "PURCHASE". The "PURCHASE" column is highlighted with a blue background. The right side has a "Data Preview" section showing a list of values for the "PURCHASE" column, such as 1803, 1953, 1630, etc.

8. Select “CUSTID” then click “Done”



The screenshot shows the Cortex Playground interface. The sidebar and main area are identical to the previous step. The main area is titled "Select a filter column". It shows a table with columns "ORDERS" and "PURCHASE". The "CUSTID" column is highlighted with a blue background. The right side has a "Data Preview" section showing a list of values for the "PURCHASE" column, such as 1803, 1953, 1630, etc.

9. Now select which Custom's Order do you want the AI to have access to and be able to answer questions about. Pick any "CUSTID"

System Prompt

Enter additional context for processing the user prompt

ORDERS
X
▼
▶

ORDERS
X
...
X
▶

10. Ask it a question like "What is this purchase about?"

☰ claude-3-5-son... ▾

ORDERS · Row 9

System
What is this purchase about?

User
{
"prodid": 149,
"purchase_amount": 7.35080000000000e+02,...

Show all ▾

claude-3-5-sonnet
This appears to be a purchase record showing:

- Product ID: 149
- Total purchase amount: \$735.08
- Purchase date: October 21, 2025
- Quantity purchased: 2 units

The customer bought 2 units of product #149 for a total of \$735.08 (which means each unit cost \$367.54) with the purchase scheduled for October 21, 2025.

View Code

System Prompt

Enter additional context for processing the user prompt

ORDERS
X
▼
▶

ORDERS
X
...
X
▶

Module 6: Snowflake Badge

Setup Autograder

1. Follow the instructions in the Autograder setup repo at
<https://mlh.link/snowflake-autograder>

Run Autograding Scripts

1. Run the `snowflake-intelligence.sql` script at
<https://mlh.link/snowflake-autograding-scripts> step by step after you set up the Autograder