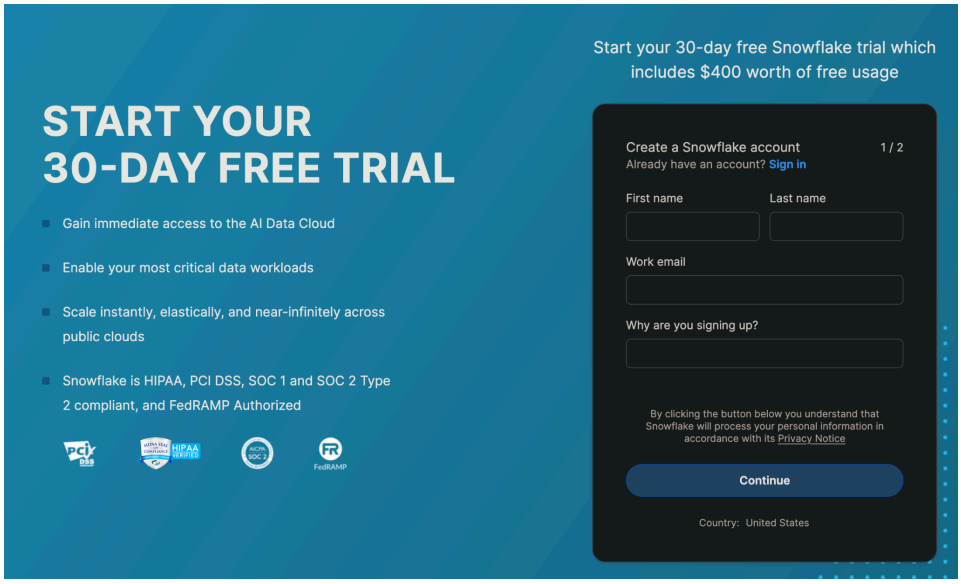
Table of Contents

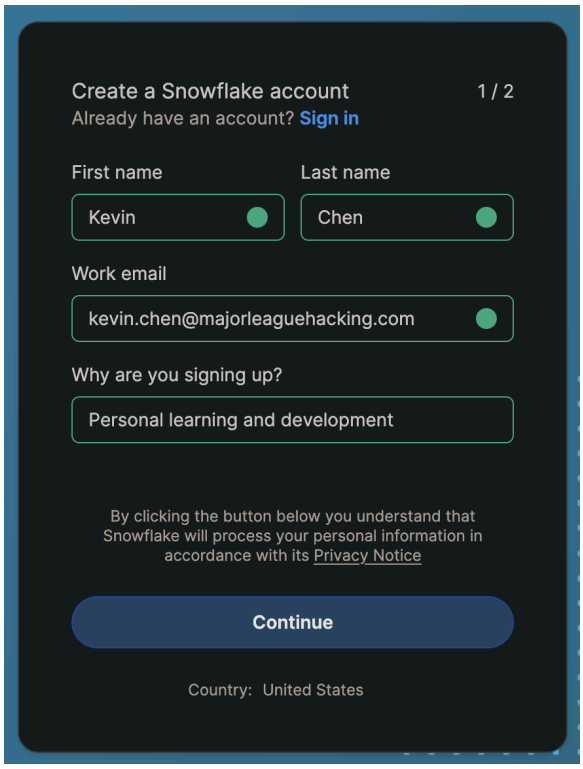
Module 1: Setting Up

Signing-up for Snowflake Trail Account

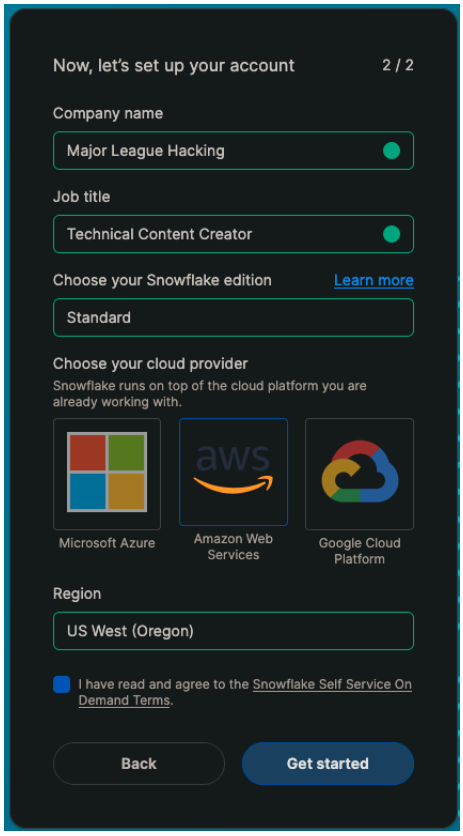
1. Visit mlh.link/snowflake-ai-trial



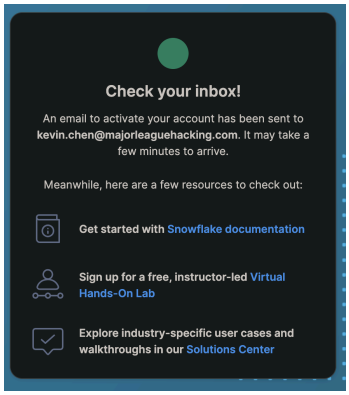
2. Fill out all the information and select “Personal learning and development”

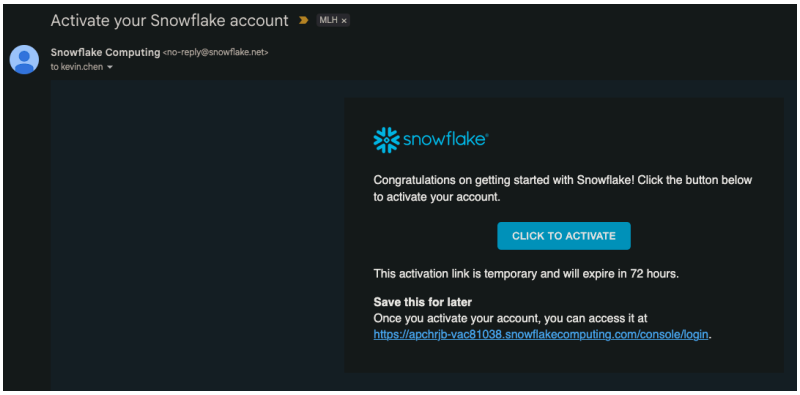


3. On the next page, enter your personal details (company name and role can be fictional). Choose the Standard edition, as it includes all AI/ML features at $2/credit, so your trial credits will stretch further, but any edition will work for this workshop. You should see your cloud provider and region 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.

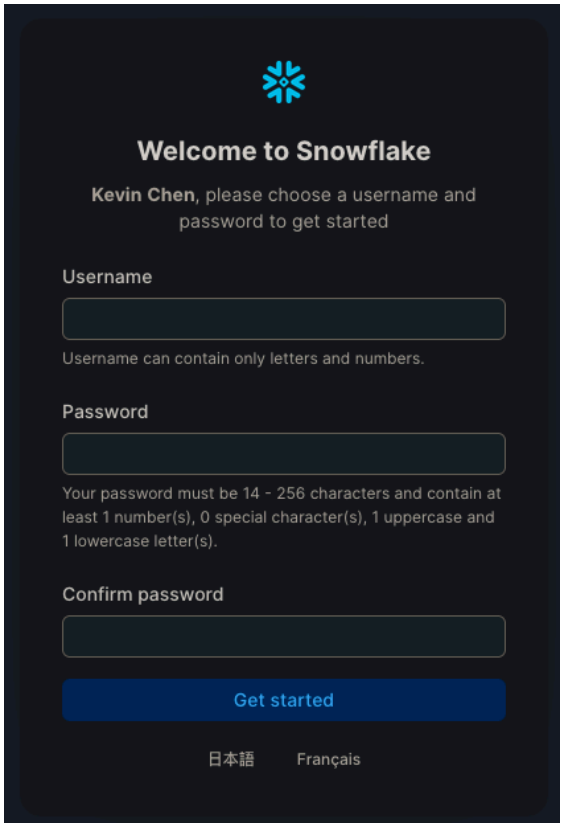


4. Answer the next two optional pages if desired
5. Check for an email and activate your account!

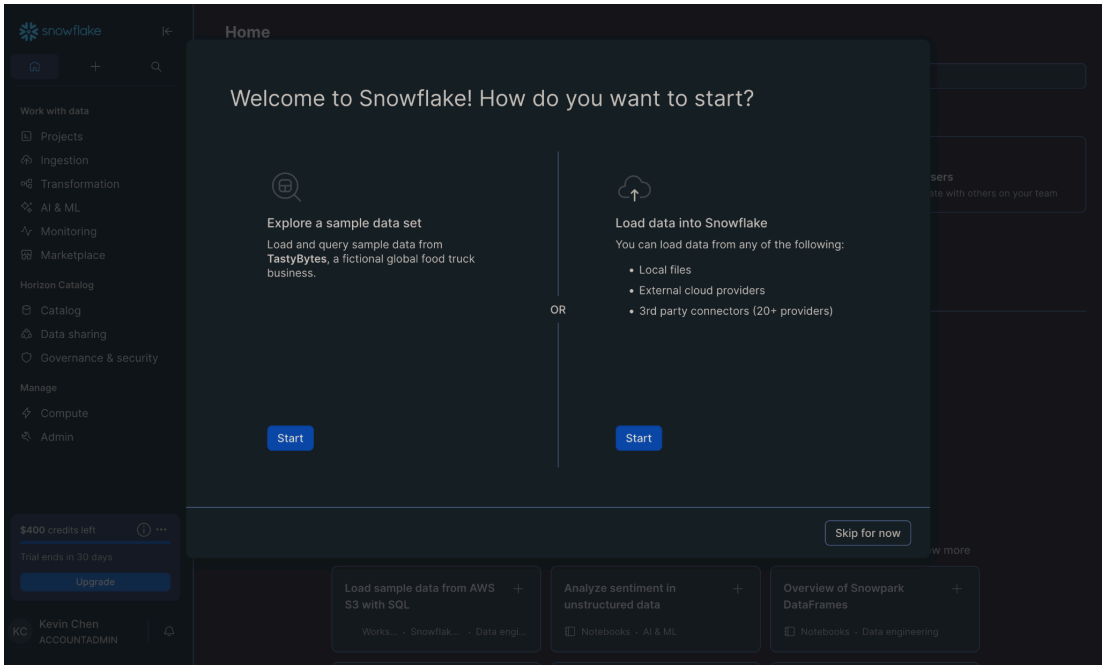




6. Set up your user account for your new Snowflake account.



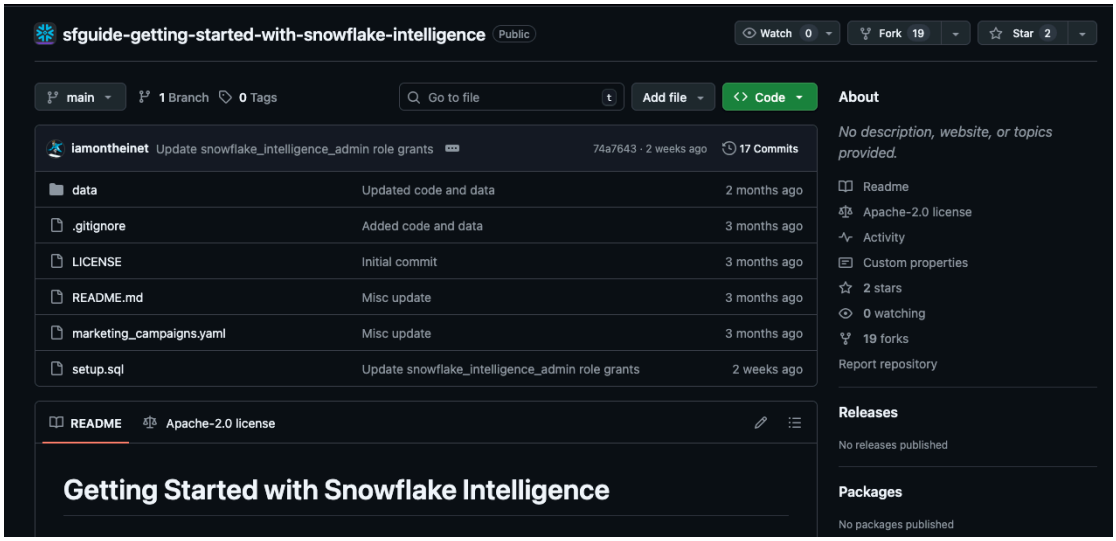
7. Perfect, leave that page up, and you are all setup to continue back to the slides!



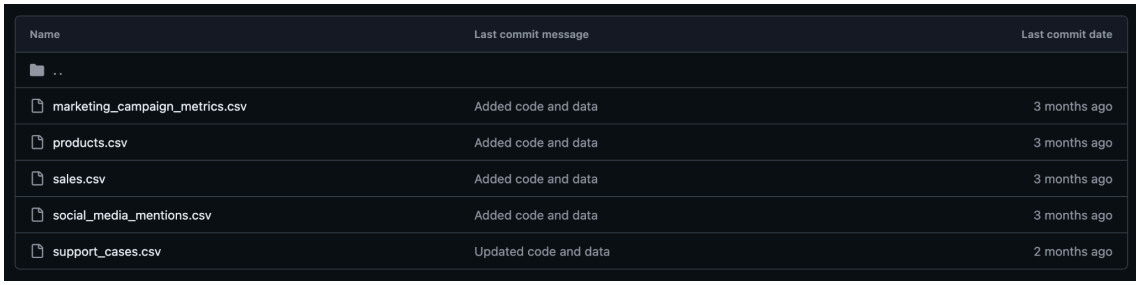
Loading in Data

Showcase the Datasets

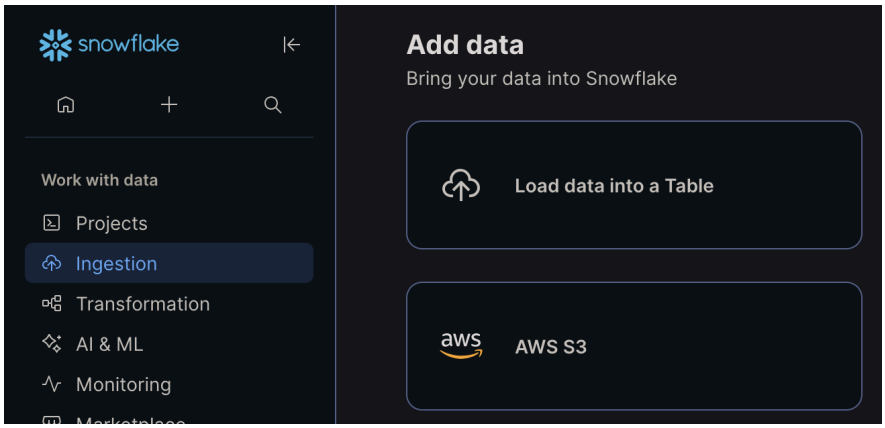
1. Visit mlh.link/snowflake-data



2. Look through the data folder and explore some files. Explain that you could directly ingress these files via the Snowflake UI.

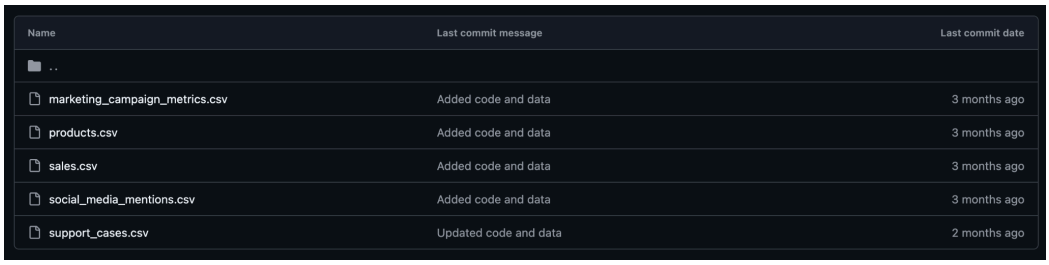


3. Show them “Ingestion > Load data into Table” being the location where they would do so

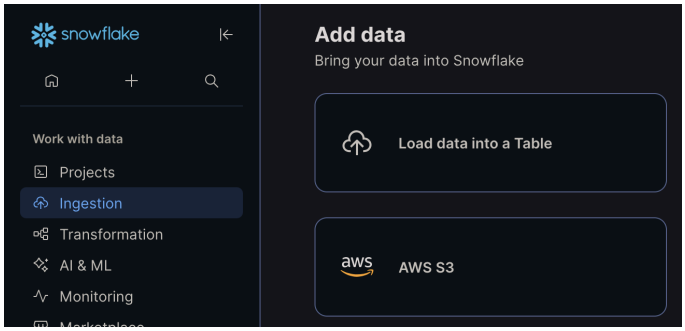


Optional: Ingest Data via Local File

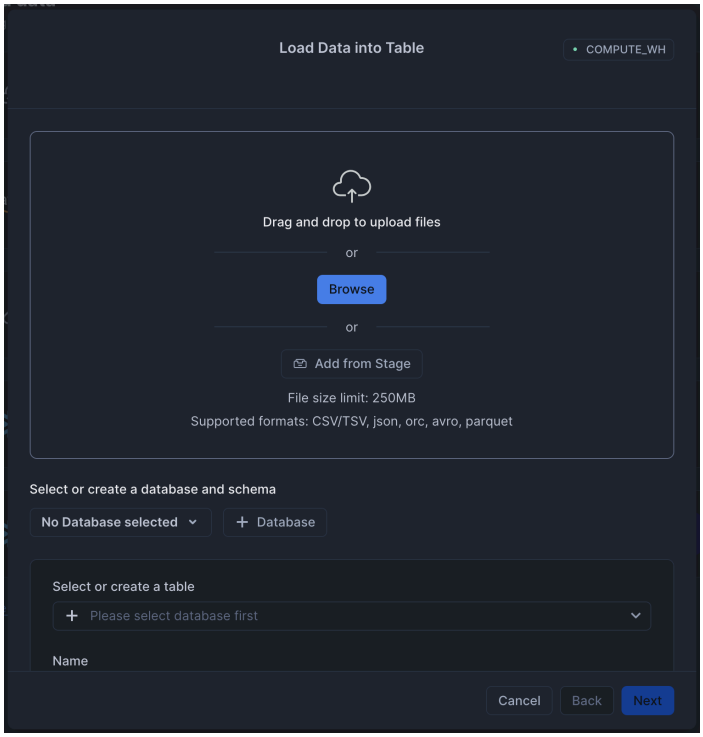
1. Download any one of the .csv files

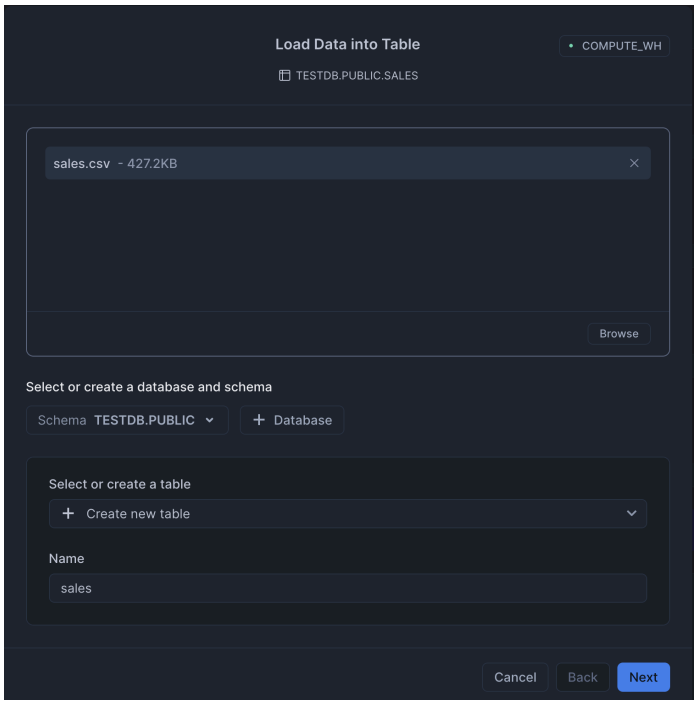


2. Go to “Ingestion > Load data into Table”

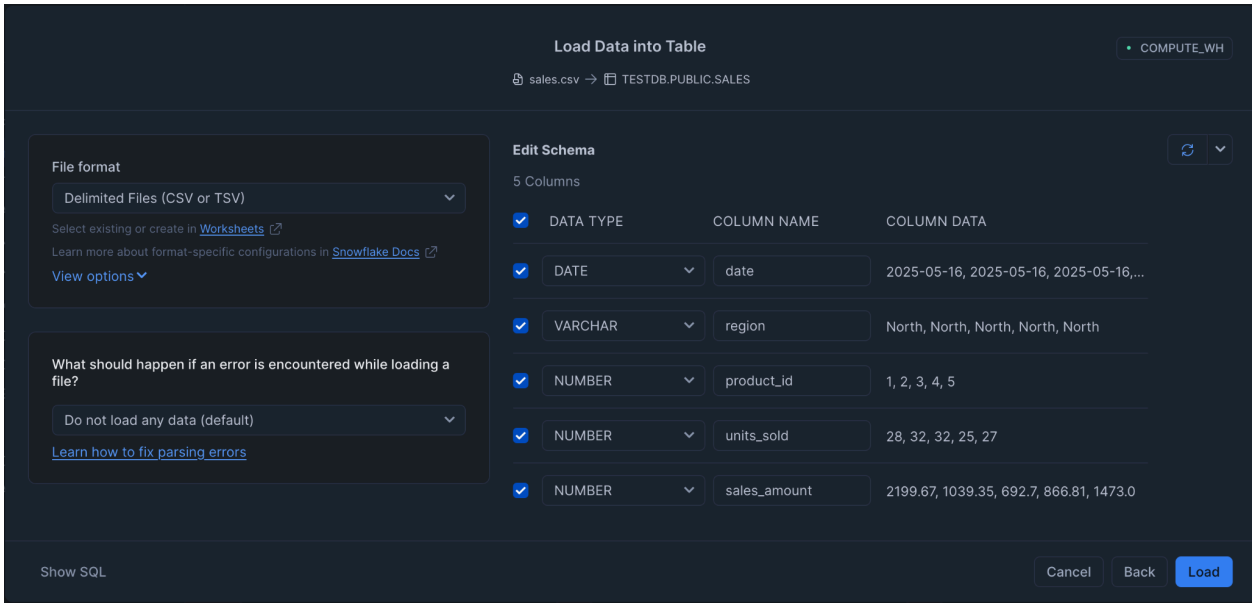


3. Click Browse and upload the file you downloaded. Click + Database, name it TestDB, and confirm. Finally, enter your desired name for the new table before loading.

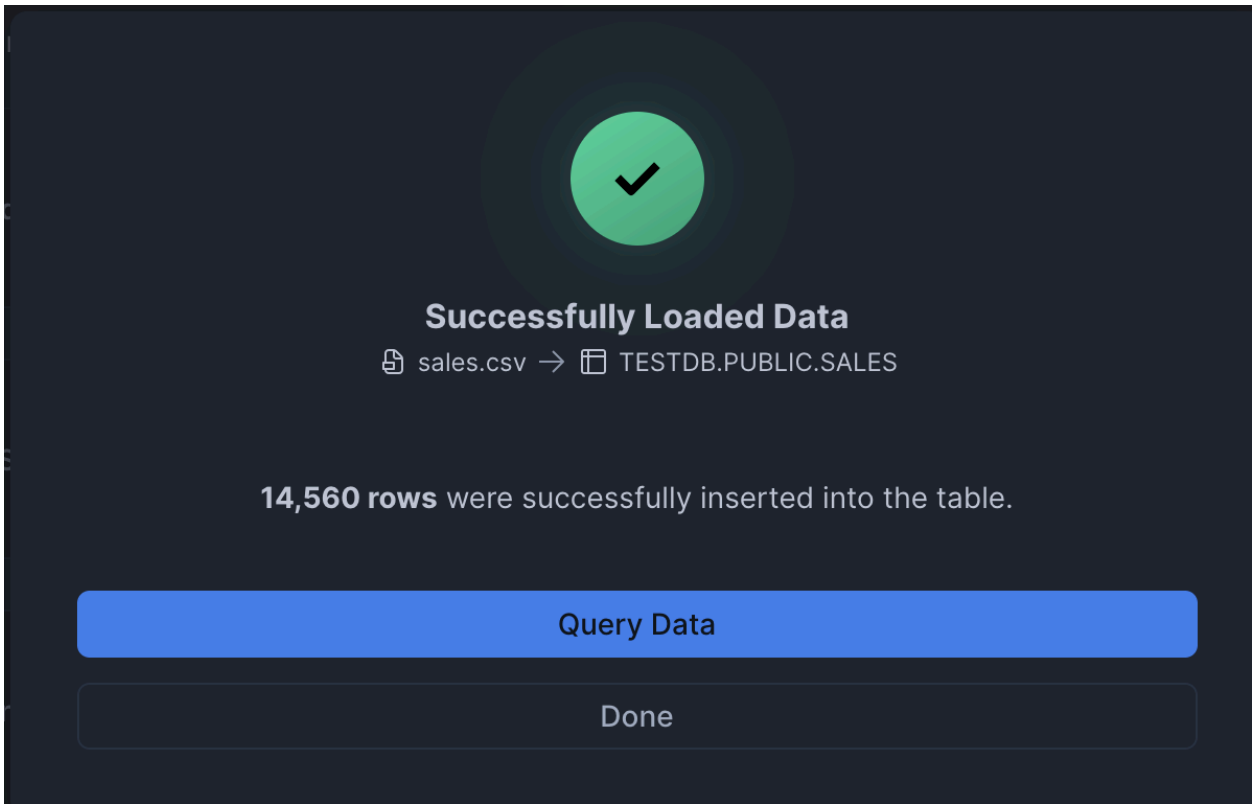


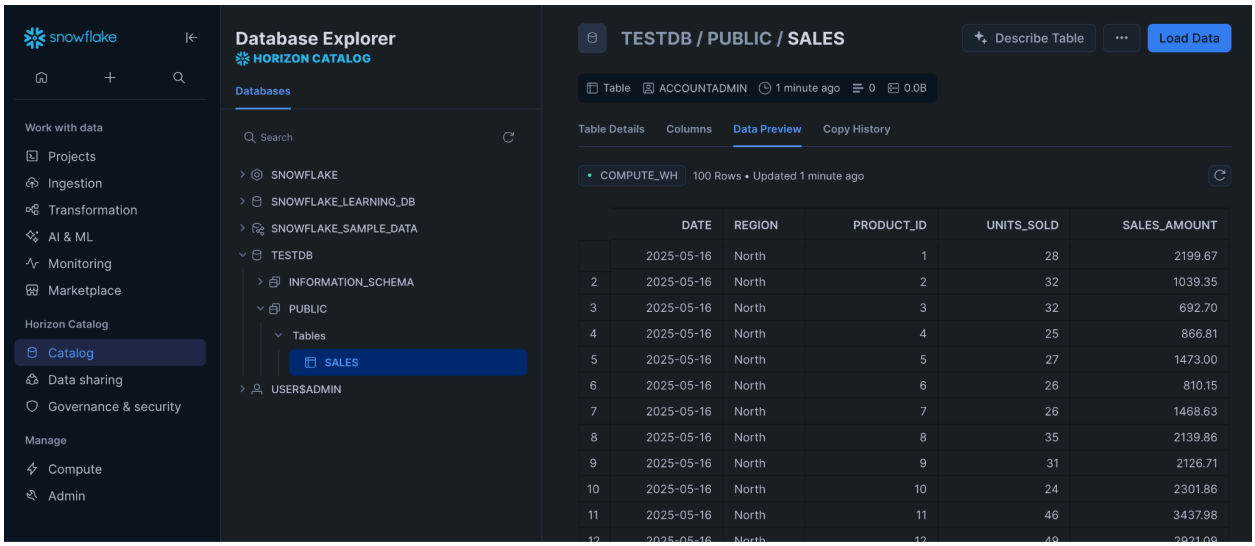


4. Click “Next” to open the schema configuration screen for your new table. Verify the column definitions and data types are correct, then click “Load”.



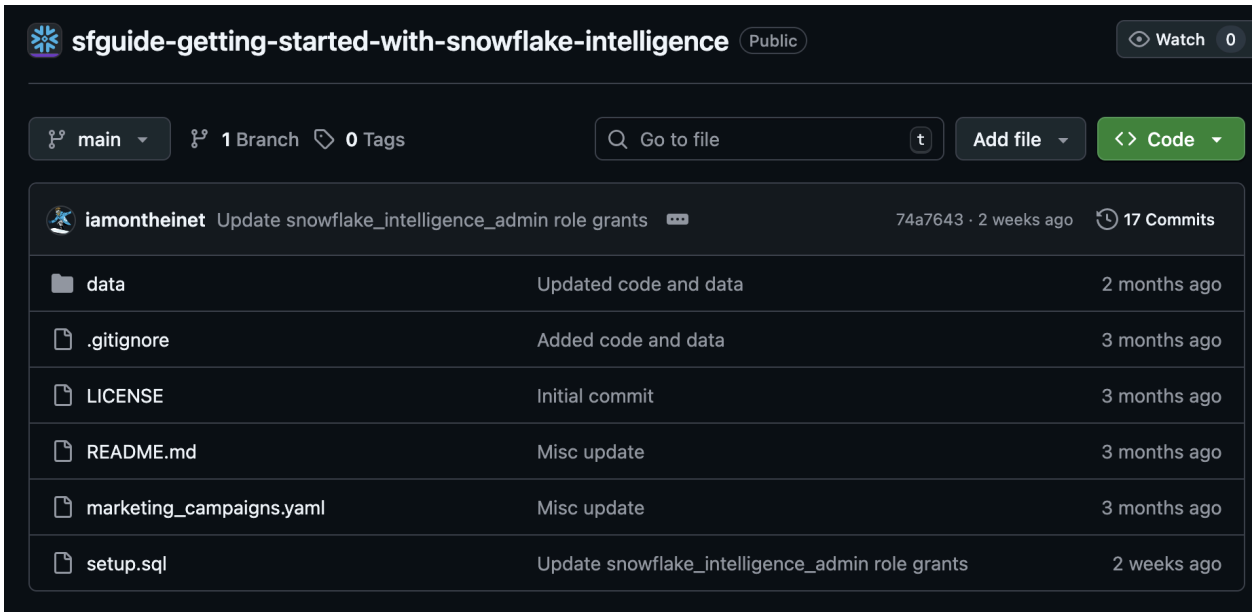
5. After loading completes, and you see “Successfully Loaded Data,” click “Done”. Then go to “Catalog > TestDB/Public/Tables/<TABLENAME>” to view the imported records.

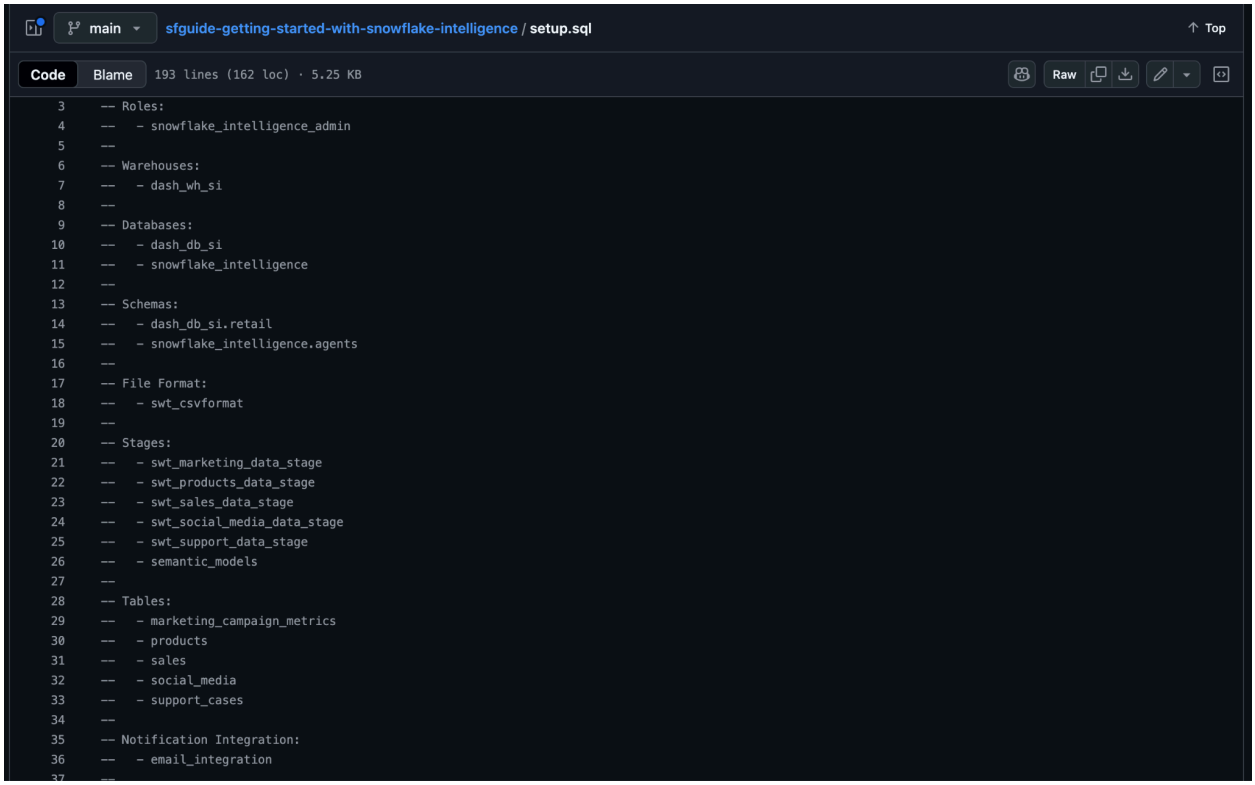




Showcase `setup.sql`

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





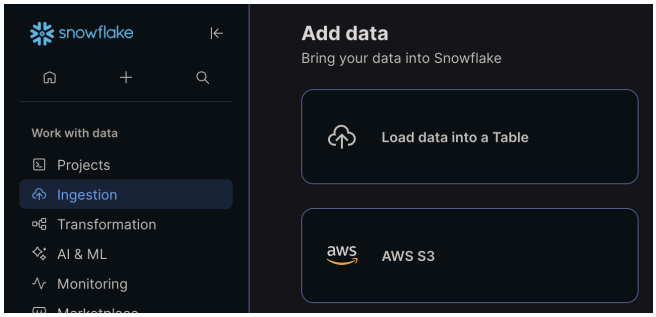
2. Scroll through and explain what the script is doing. Note: We won't be using lines 151 - 189 in this Workshop.
    a. 42: Switch to the ACCOUNTADMIN role.
    b. 44–47: Create the snowflake_intelligence_admin role and grant it account-level privileges (warehouse, database, integration creation).
    c. 49–52: Assign snowflake_intelligence_admin to the current user and set it as the default role and warehouse.
    d. 54: Activate the snowflake_intelligence_admin role.
    e. 55–62: Provision core environments
        i. dash_db_si database with retail schema and dash_wh_si warehouse
        ii. snowflake_intelligence database with agents schema and grant agent creation
    f. 64–66: Set session context to dash_db_si.retail and dash_wh_si.
    g. 68–71: Define the swt_csvformat file format for CSV ingestion.
    h. 74–149: For each dataset (marketing, products, sales, social_media, support):
        i. Create an external stage pointing to the corresponding S3 path
        ii. Create the target table with appropriate columns
        iii. Load data into the table via COPY INTO
    i. 151: Create the semantic_models stage with server-side encryption.
    j. 153–189: Configure email_integration and define the send_email Python stored procedure to wrap SYSTEM$SEND_EMAIL.
    k. 191: Enable cross-region Cortex features via the CORTEX_ENABLED_CROSS_REGION account parameter.
    l. 193: Display a setup completion confirmation message.

Optional: Ingest Data via S3 Buckets

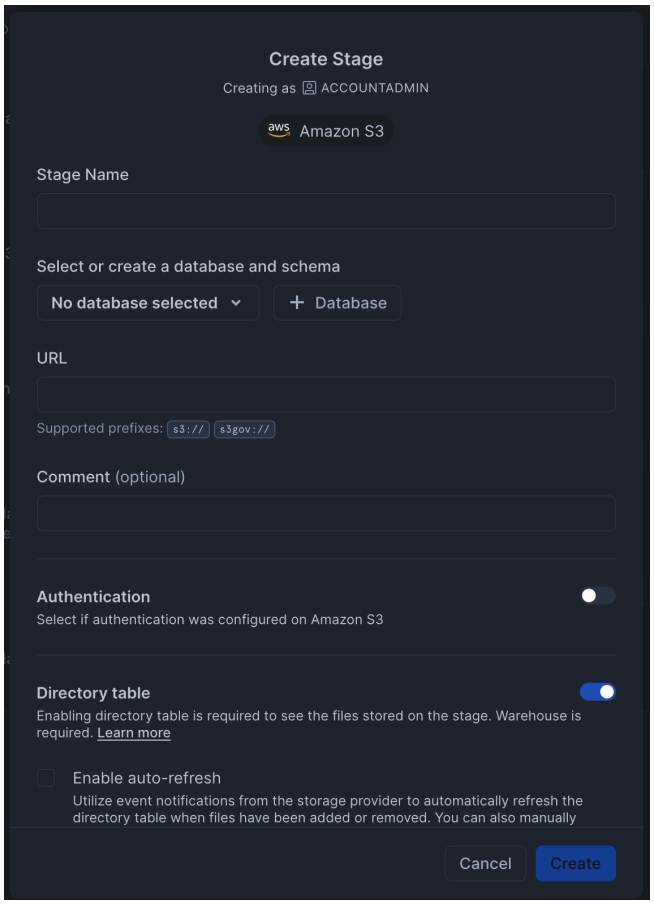
1. Grab one of the S3:// endpoints from the `setup.sql` script

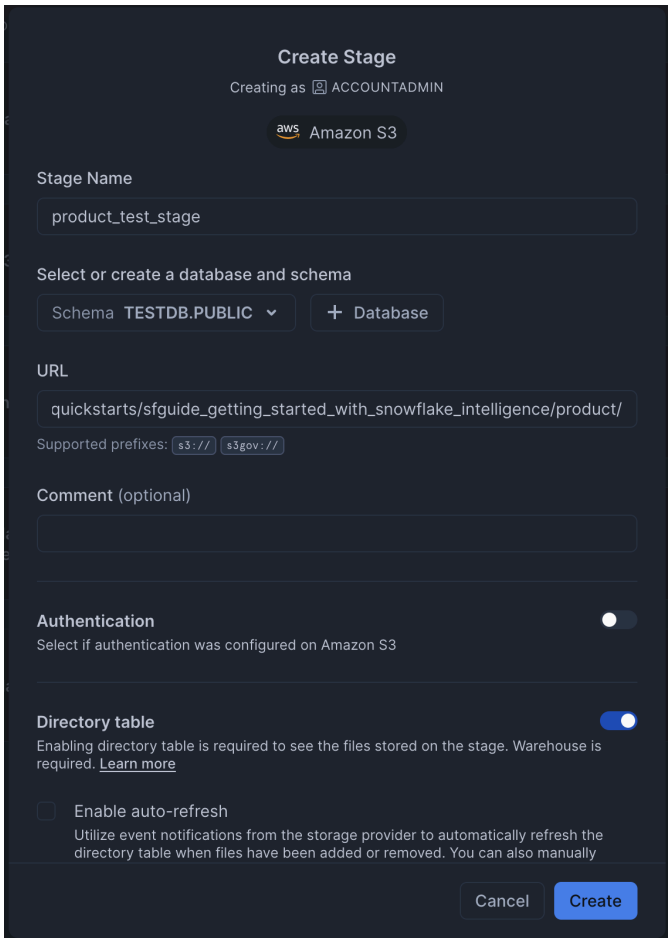
```
89    -- create table products and load data from s3 bucket
90    create or replace stage swt_products_data_stage
91      file_format = swt_csvformat
92      url = 's3://sfquickstarts/sfguide_getting_started_with_snowflake_intelligence/product/';
```

2. Go to “Ingestion > AWS S3”

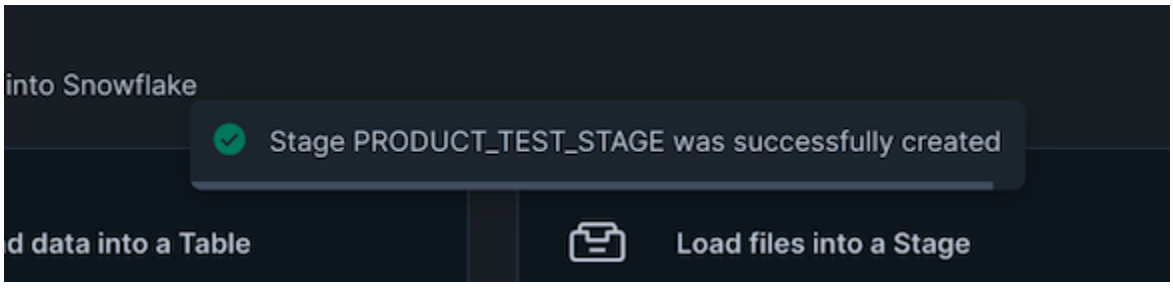


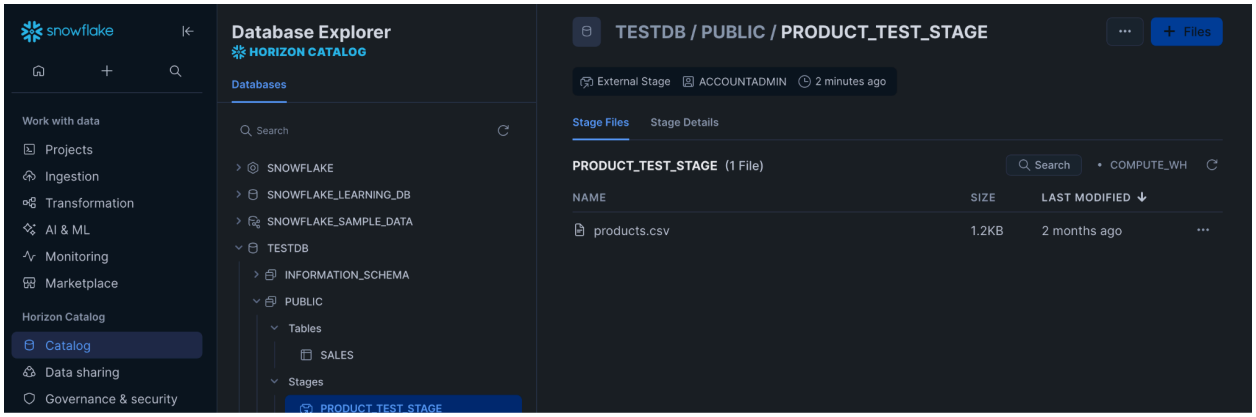
3. Enter a name for your stage and paste in the S3 URL from the `setup.sql` script, then click “Create”.



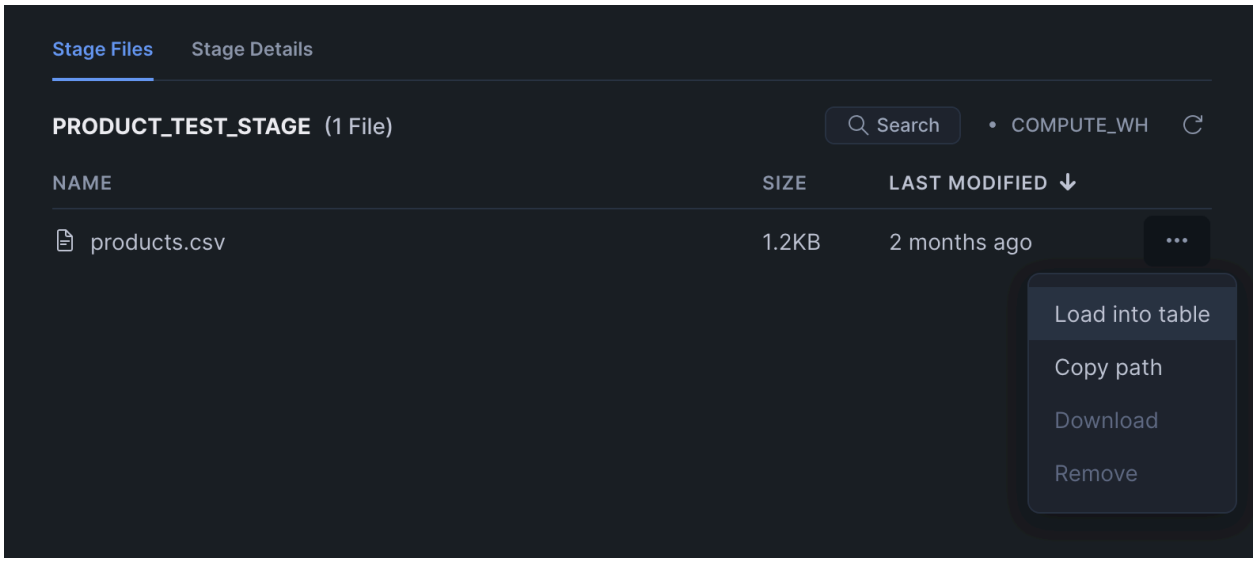


4. Once created, go to “Catalog > TestDB/Public/Stages/<STATGENAME>” to view the files available for loading.





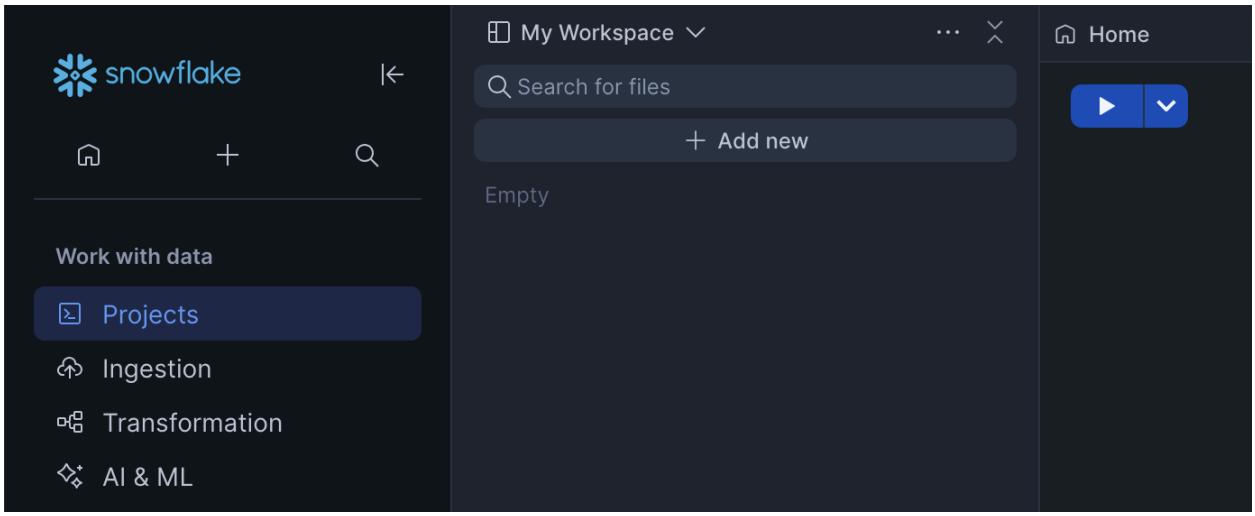
5. Click the three dots next to the CSV file and choose “Load into table”.



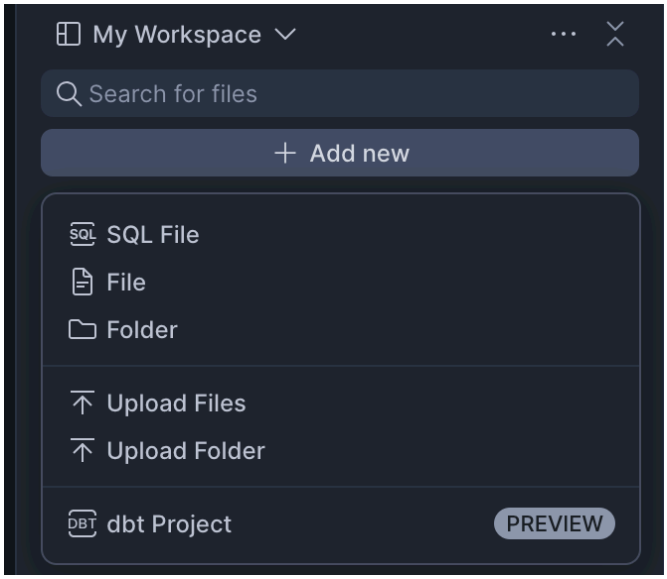
6. Follow the same prompts as in Step 3 for Ingest Data via Local File, creating a new table in TESTDB and defining the table schema.

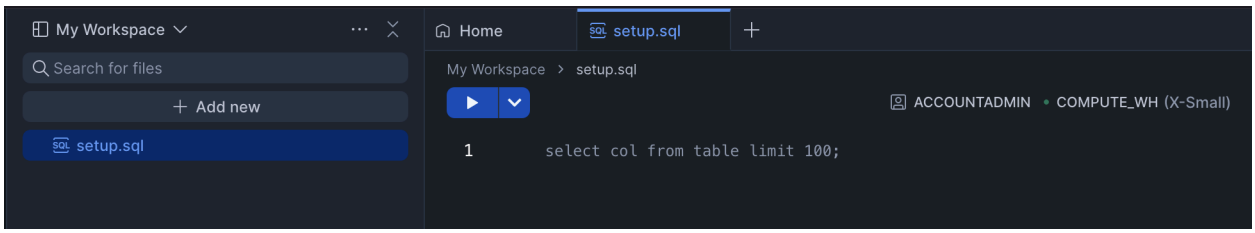
Setting Up and Executing a SQL File in Snowflake Workspace

1. Go to “Projects” which should take you to “My Workspace”

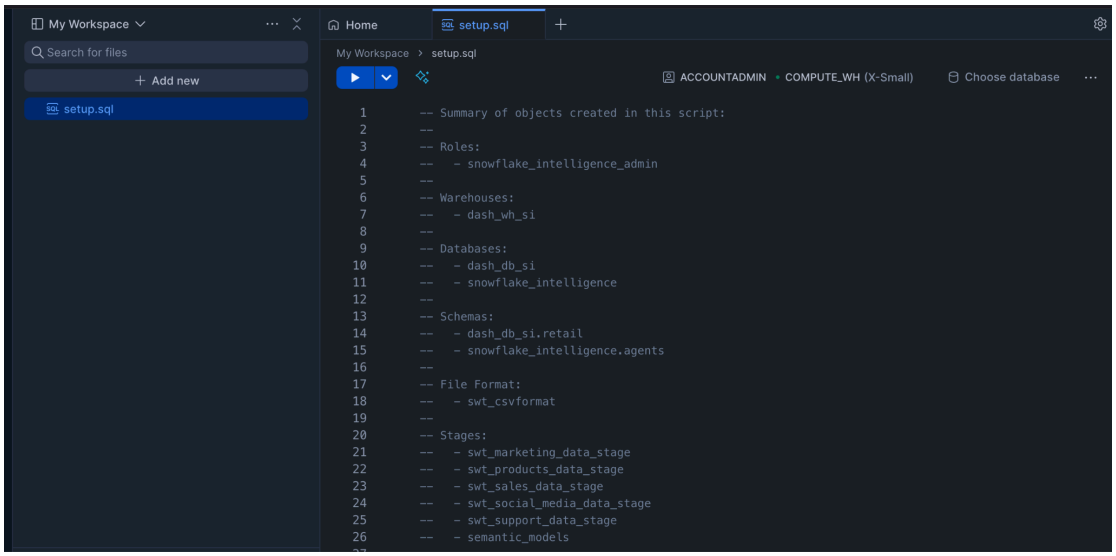


2. Click on “+ Add new” and select “SQL File” and name it `setup.sql`

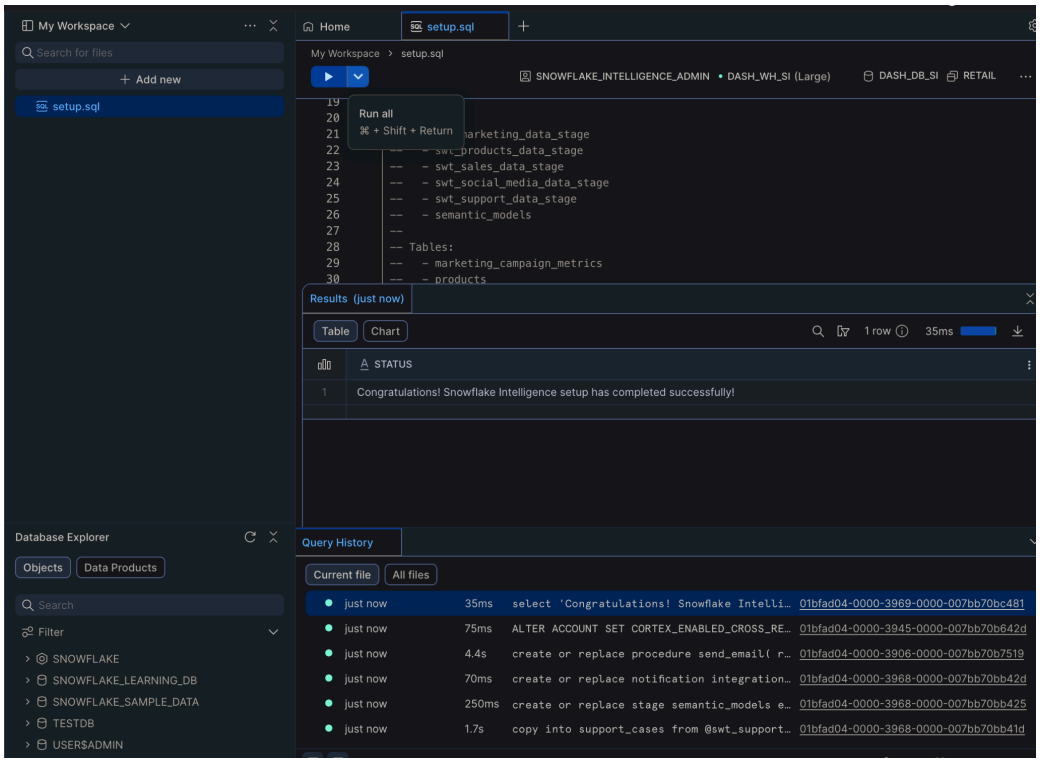




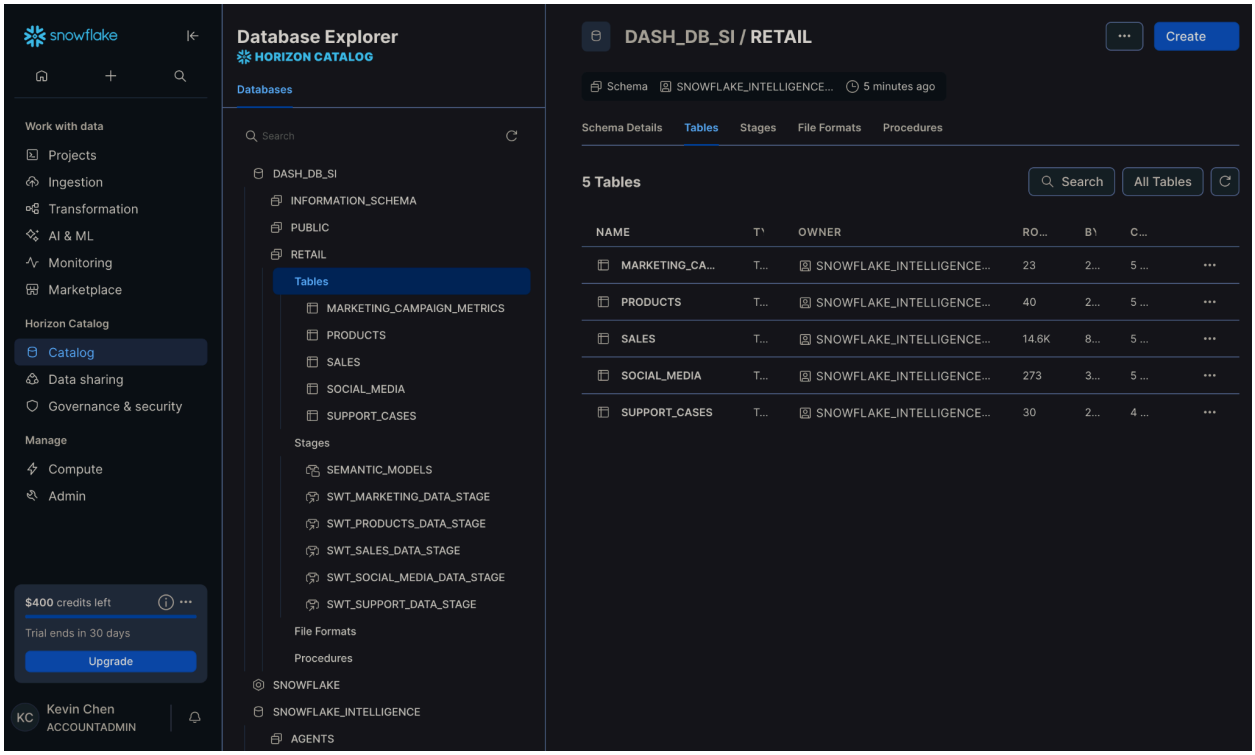
3. Navigate back to the GitHub repo, open `setup.sql`, copy its contents, and paste them into the SQL file you created in your workspace.



4. If you haven't yet, briefly explain the purpose of `setup.sql`: it automates role, warehouse, database, schema, stage, and table creation, then loads all datasets from S3. Click "Run All" to execute the entire script. You should see: "Congratulations! Snowflake Intelligence setup has completed successfully!"



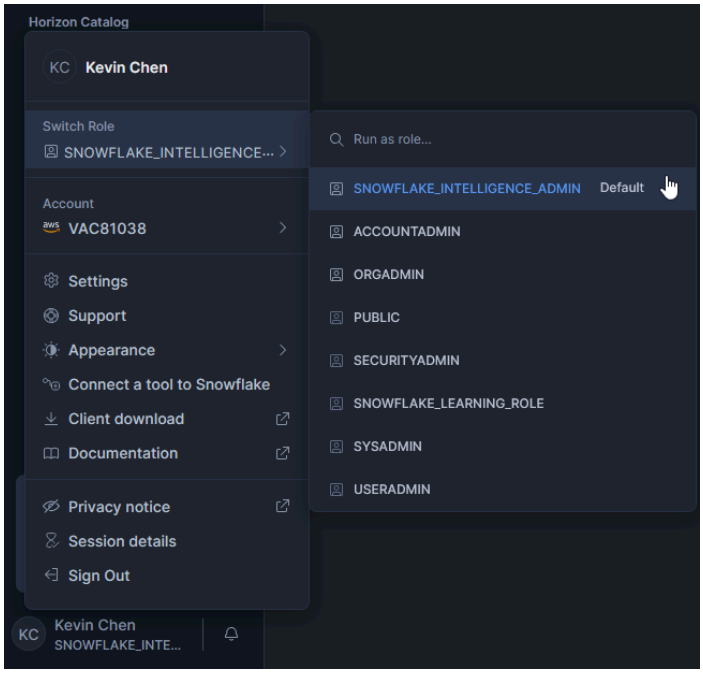
5. Finally, verify the script ran successfully by going to “Catalog” and checking for “DASH_DB_SI/RETAIL” DB which should list five tables populated from S3. Also check to make sure “SNOWFLAKE_INTELLIGENCE” DB should exist and be ready for creating AI agents.



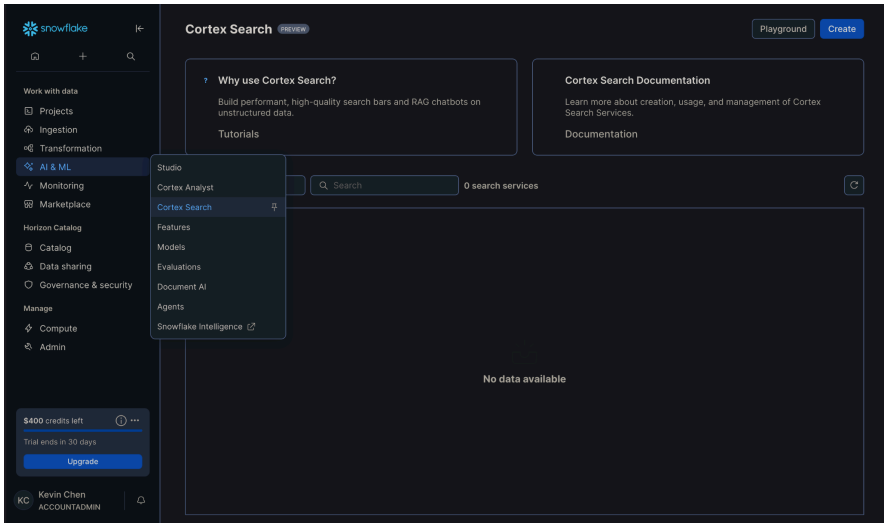
Module 2: Cortex Search

Setting Up Cortex Search

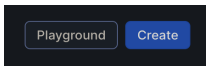
1. In the bottom-left corner, select your account, then switch your role to SNOWFLAKE_INTELLIGENCE_ADMIN.



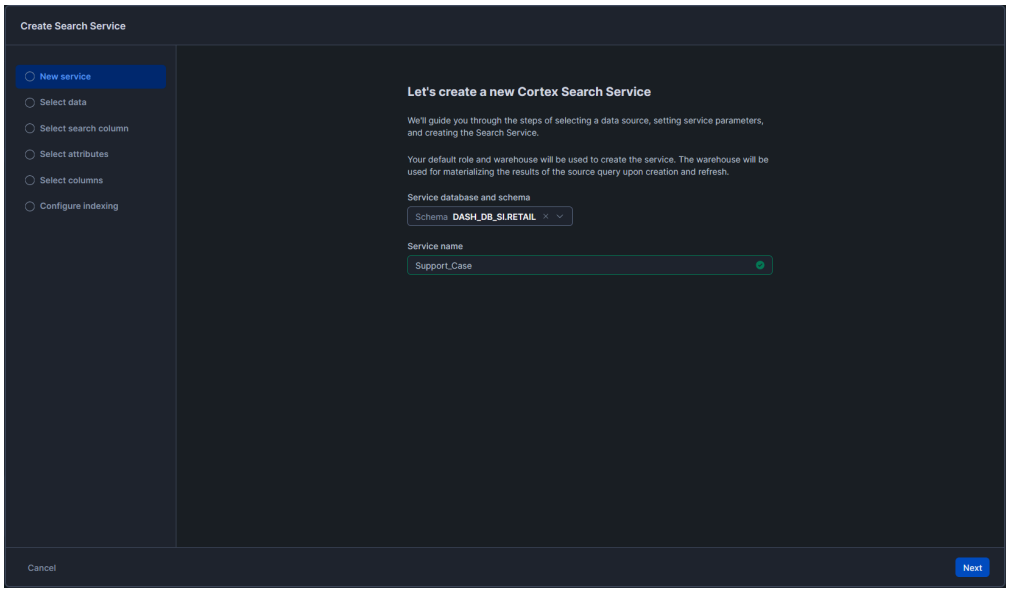
2. Navigate to “AI & ML > Cortex Search”



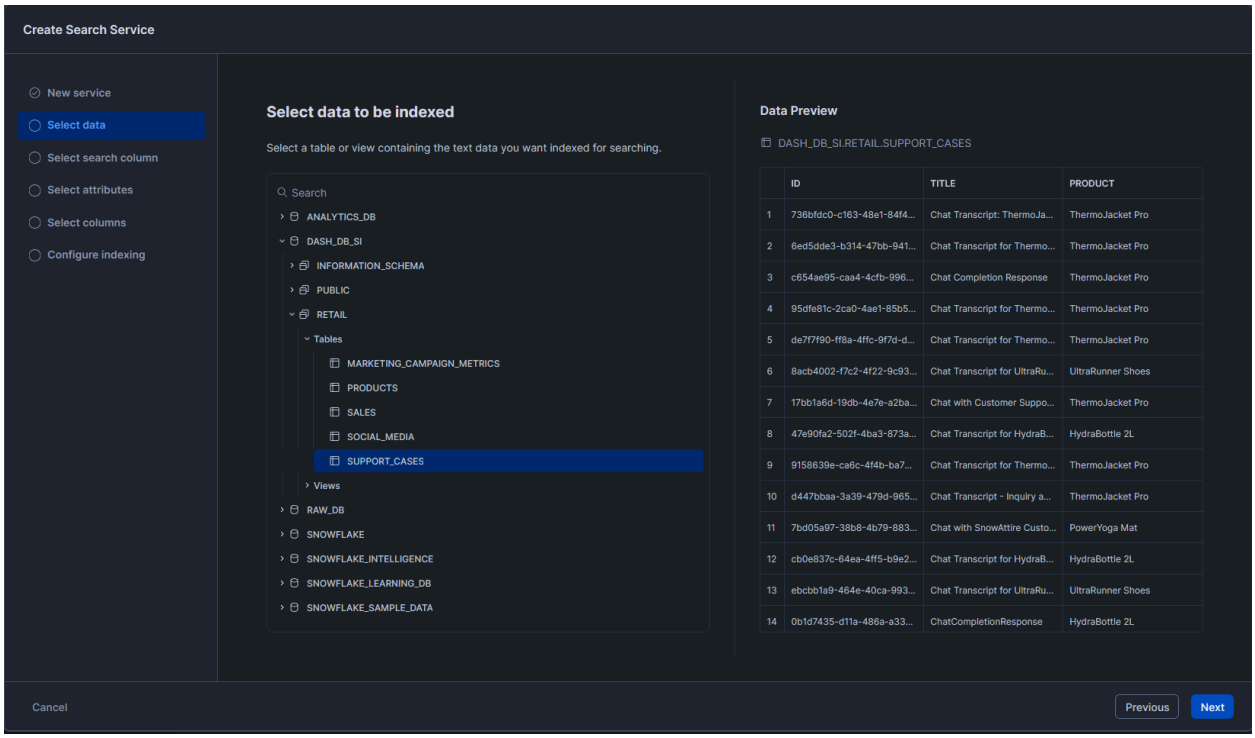
3. Click “Create” in the top right corner



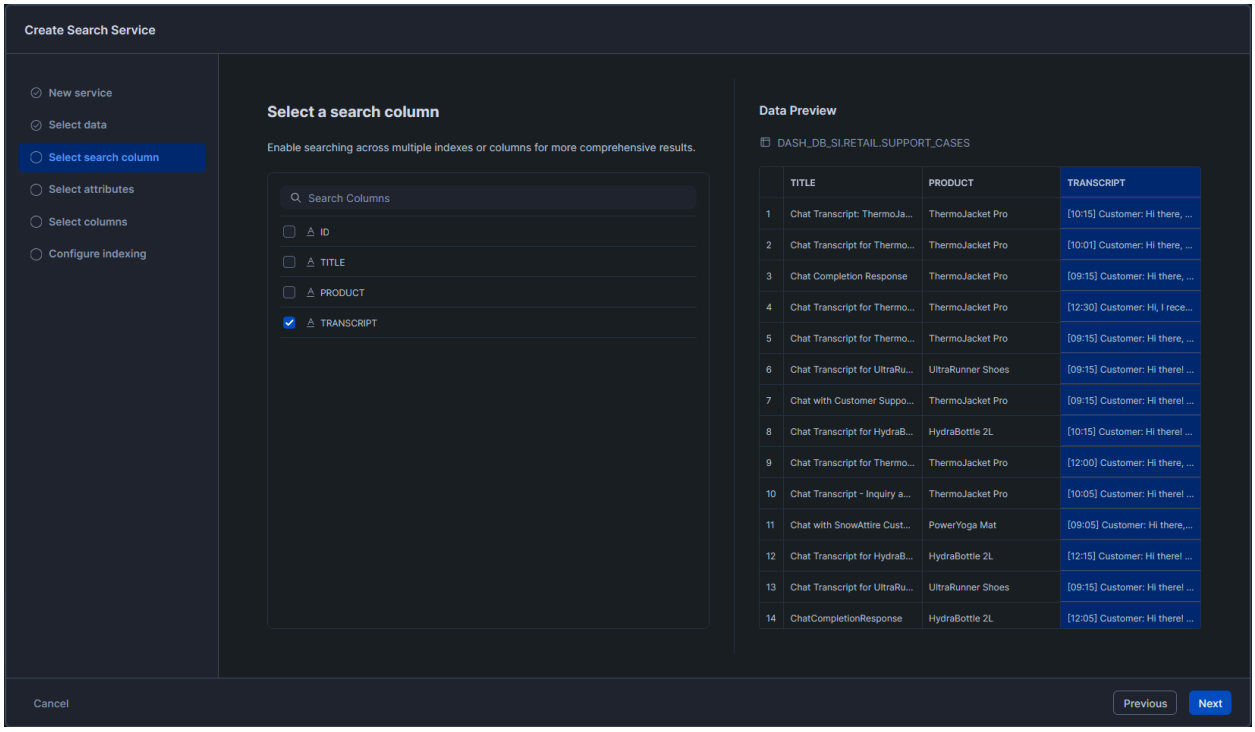
4. Select “DASH_DB_SI.RETAIL” for Database and Schema, and “Support_Cases” for Name. Then select “Next: Select data”



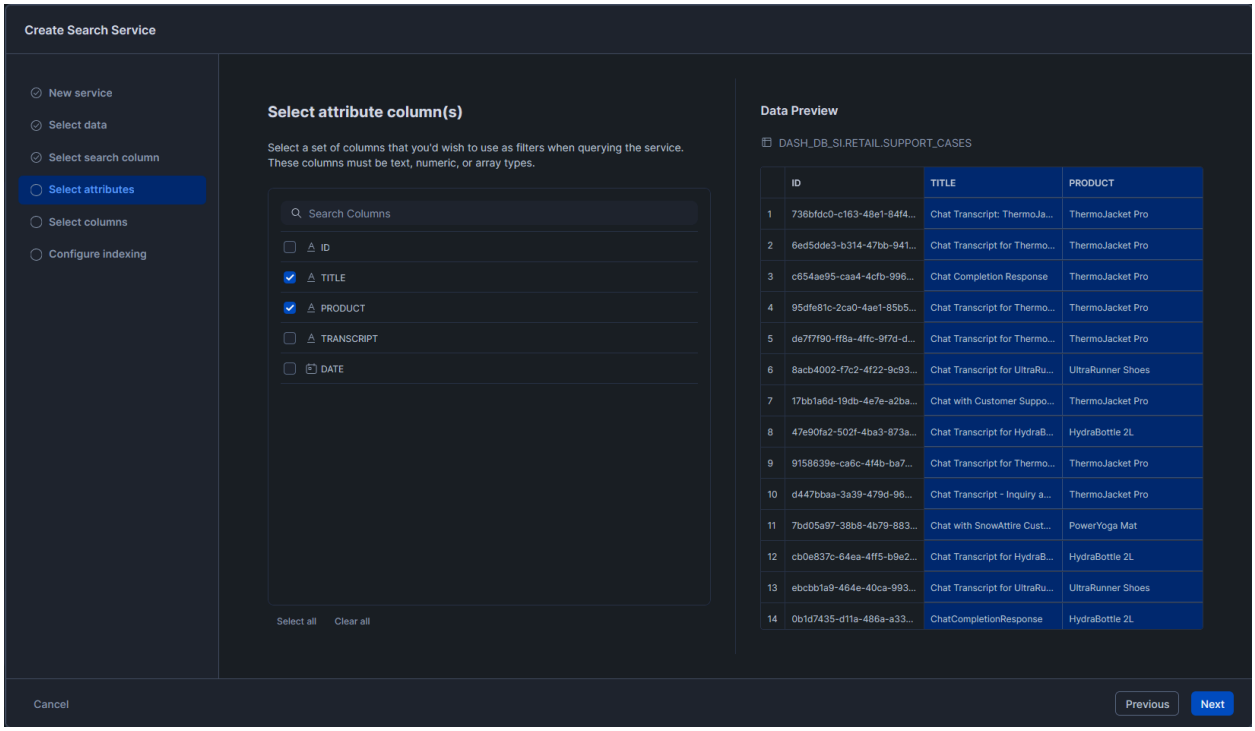
5. Select the “SUPPORT_CASES” table as the data to be indexed. Then select “Next: Select search column”



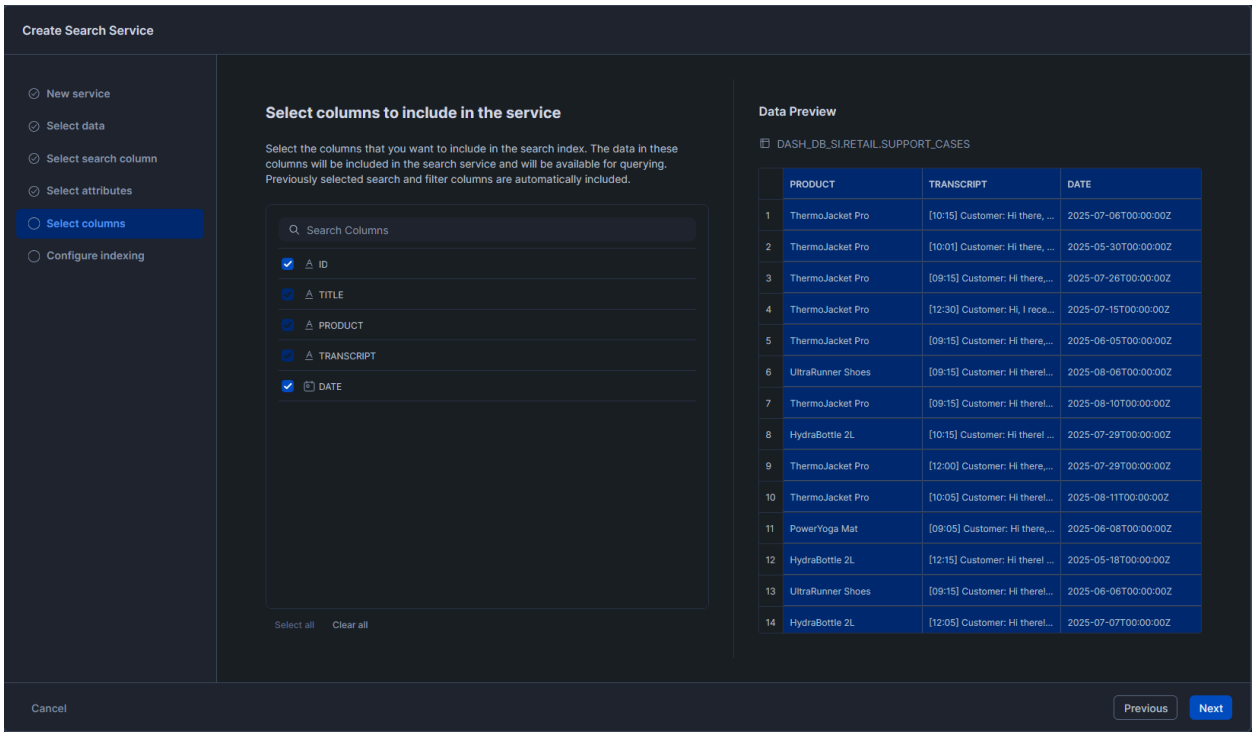
6. Select “TRANSCRIPT” as the search column. Then select “Next: Select attributes”



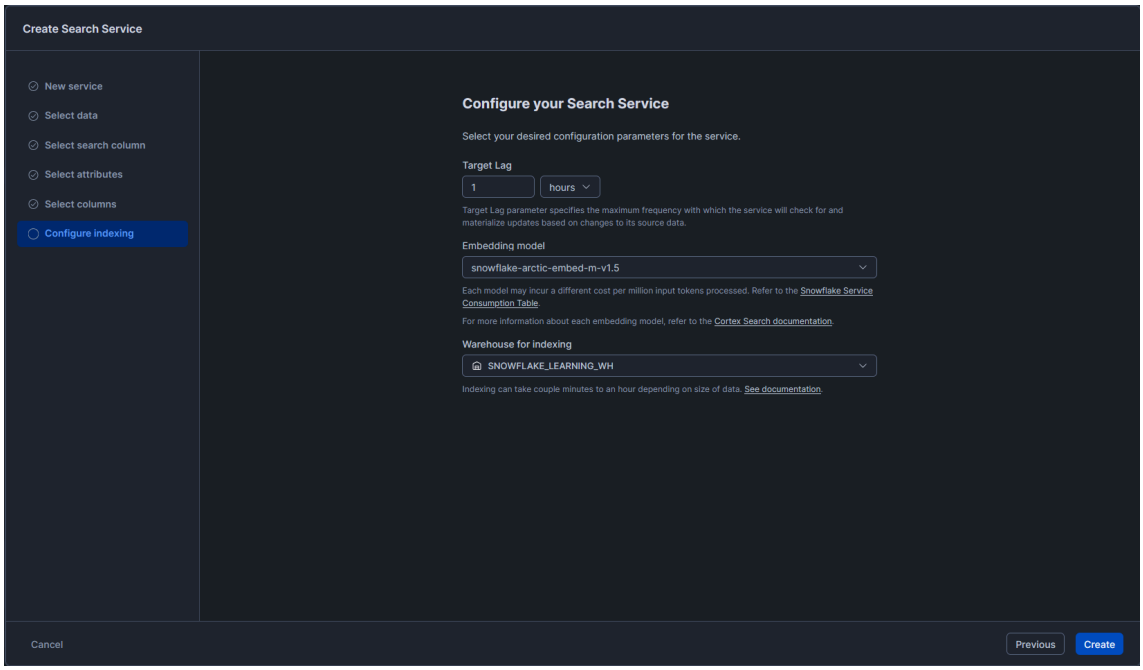
7. Select “TITLE, PRODUCT” as the attribute column(s). Then select “Next: Select search column”



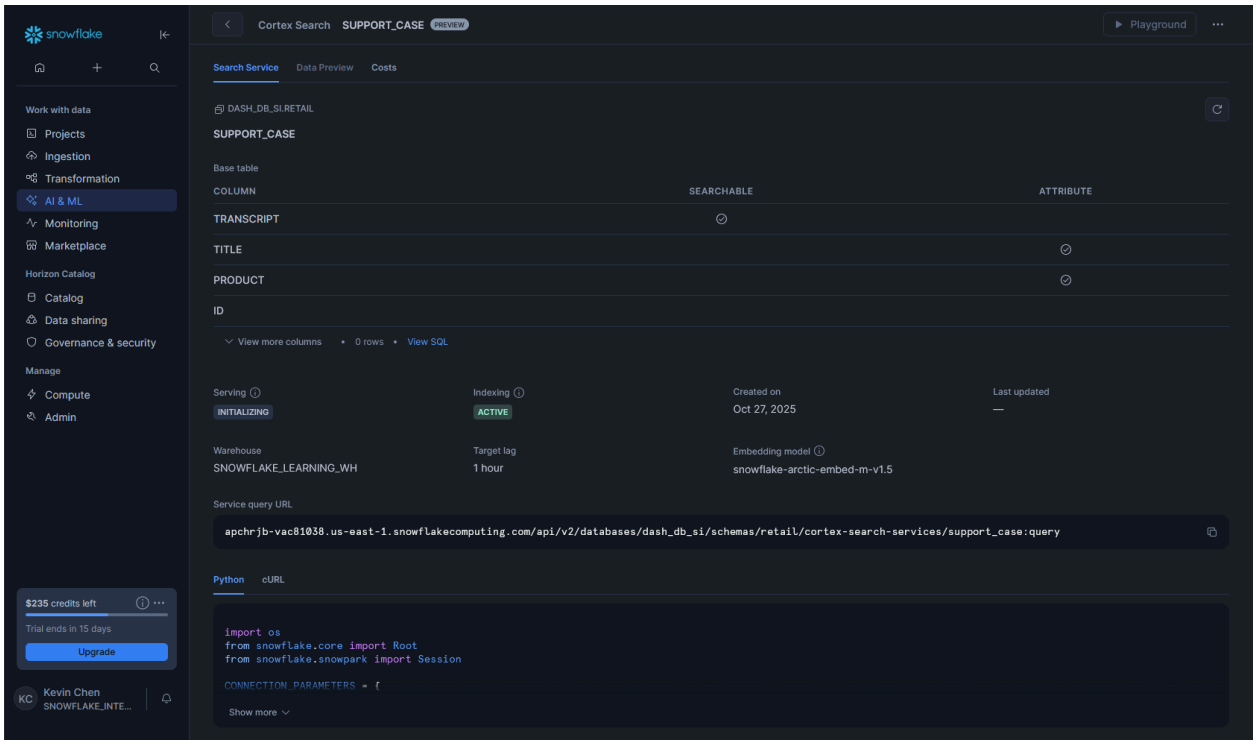
8. Select “Select all” in the bottom left as column to include in the service. Then select “Next: Configure indexing”



9. Select “SNOWFLAKE_LEARNING_WH” as the Warehouse for indexing. Then select “Create Search Service”

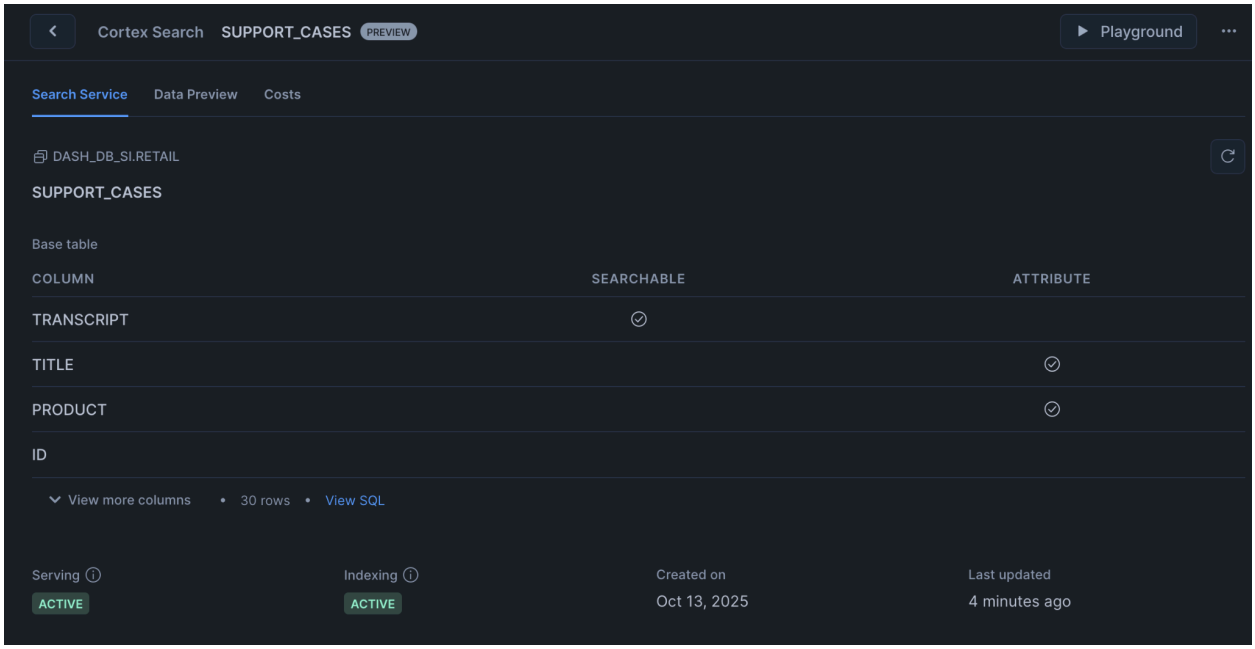


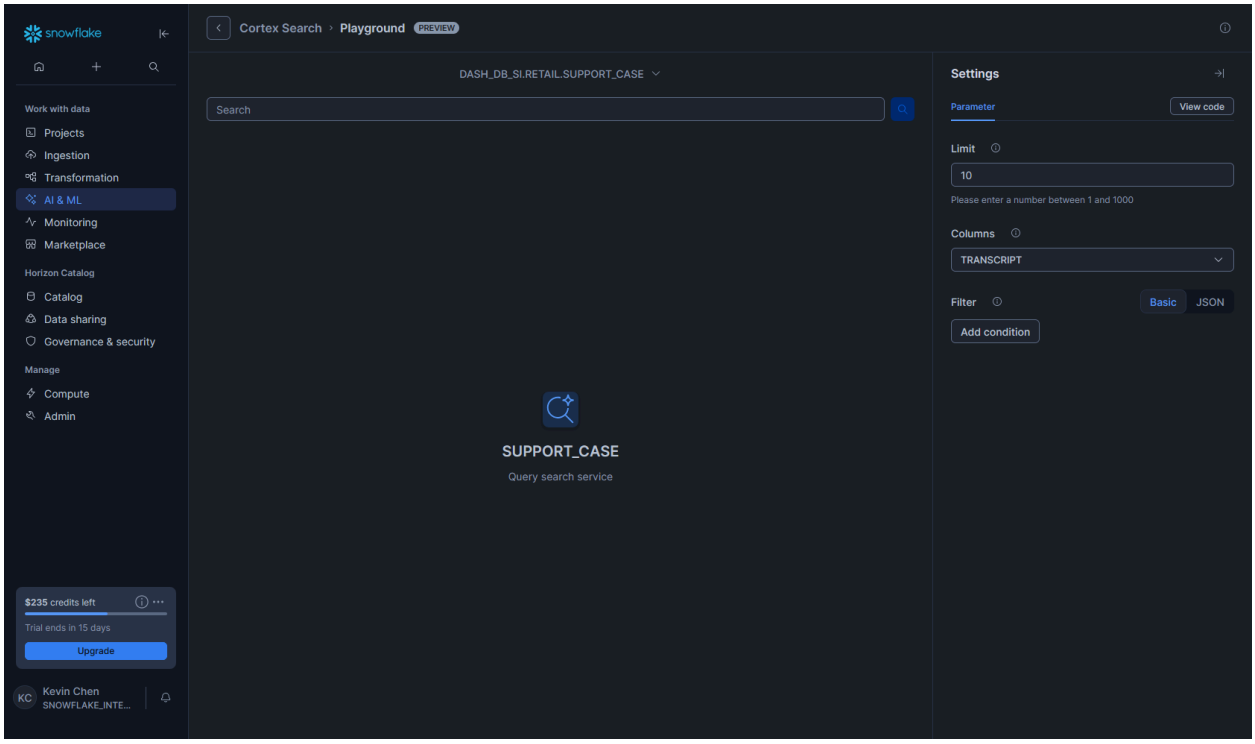
10. There we go, now you have a Cortex Search Service. Make sure to show how you can also use the different methods to display to interact with Cortex Search.



Using Cortex Search

1. Make sure everyone is at this step before proceeding. Once the “Serving” and “Indexing” states are both “Active” we can select the “Playground” button in the top right corner. Note: You may need to refresh the page, either by using your web browser’s refresh button or by selecting the refresh icon located in the top-right corner under Search Services.





2. Now you can perform any search, here are some sample search terms:
    a. Clothes — Shows you all tickets related to clothing products
    b. Shoes — Shows you all tickets related to shoes, including activities you would need shows for
    c. Replacement — Shows you all tickets related to issuing a replacement

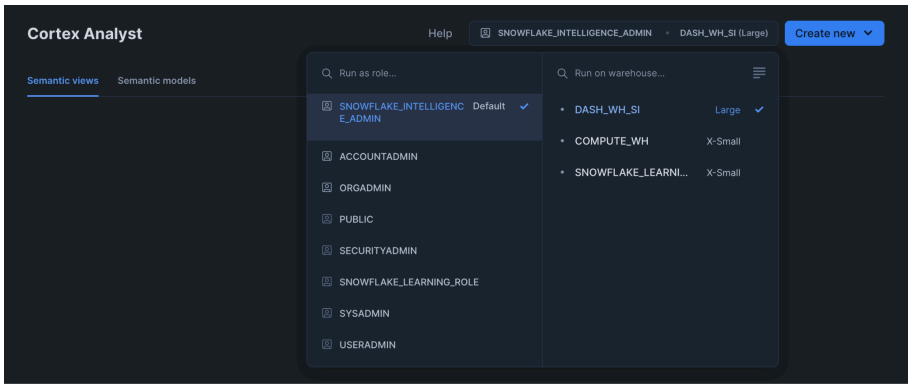
   d. Damage — Shows you all tickets that take about any damage, defect, etc
   e. Feel free to mix and match them, be creative and come up with additional ones on your own
3. Lastly, ask it a question like “What are the top theme's of all support tickets?”, you should see that it doesn’t really provide an answer to those type of question as Cortex Search works as a RAG for unstructured data. This is your transition to introduce our next Module on Cortex Analyst

Module 3: Cortex Analyst

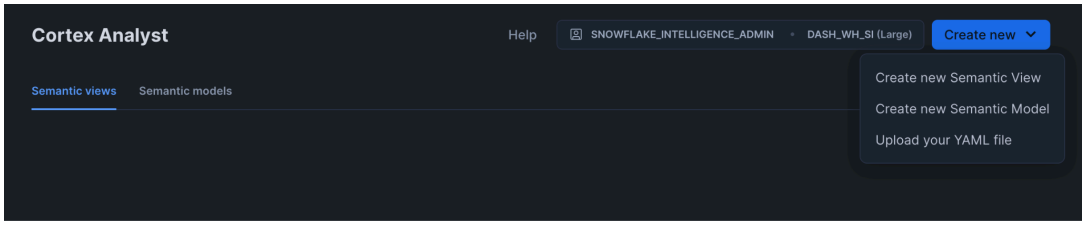
Setting Up Cortex Analyst

1. Navigate to mlh.link/snowflake-data
2. Go through the marketing_campaigns.yaml file
3. Make sure to explain the following:
    a. 1–2: name: Defines the analyst configuration name (Sales_And_Marketing_Data).
    b. 3–95: tables:
        i. Each table block (MARKETING_CAMPAIGN_METRICS [4–53], PRODUCTS [54–82], SALES [83–112], SOCIAL_MEDIA [113–143]) specifies:
            1. Which Snowflake database, schema, and table to use
            2. Primary key columns
            3. Dimensions (business attributes and synonyms; include example values and SQL expressions)
            4. Facts (quantitative measures; include synonyms and example values)
            5. Time dimensions (date fields; include synonyms and example values)
    c. 144–157: relationships:
        i. Defines how tables are joined (e.g., SALES to PRODUCTS, SOCIAL_MEDIA to MARKETING_CAMPAIGN_METRICS)
        ii. Specifies join columns, relationship type (many_to_one), and join type (inner).
    d. 158–175: verified_queries:
        i. Example of a validated query (sales)
            1. Written as natural language
            2. Shows corresponding SQL
            3. Indicates who verified and when
            4. Used for onboarding or demonstration
4. Download the marketing_campaigns.yaml
5. Navigate to “AI & ML > Cortex Analyst”.

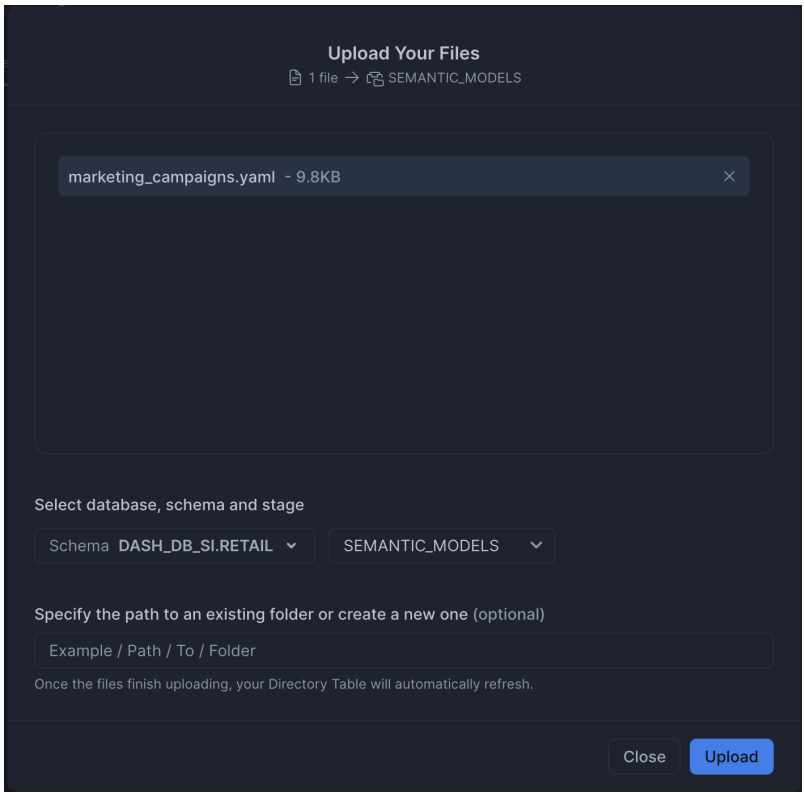
6. Update your Role and Warehouse in the top right corner to "SNOWFLAKE_INTELLIGENCE_ADMIN | DASH_WH_SI"



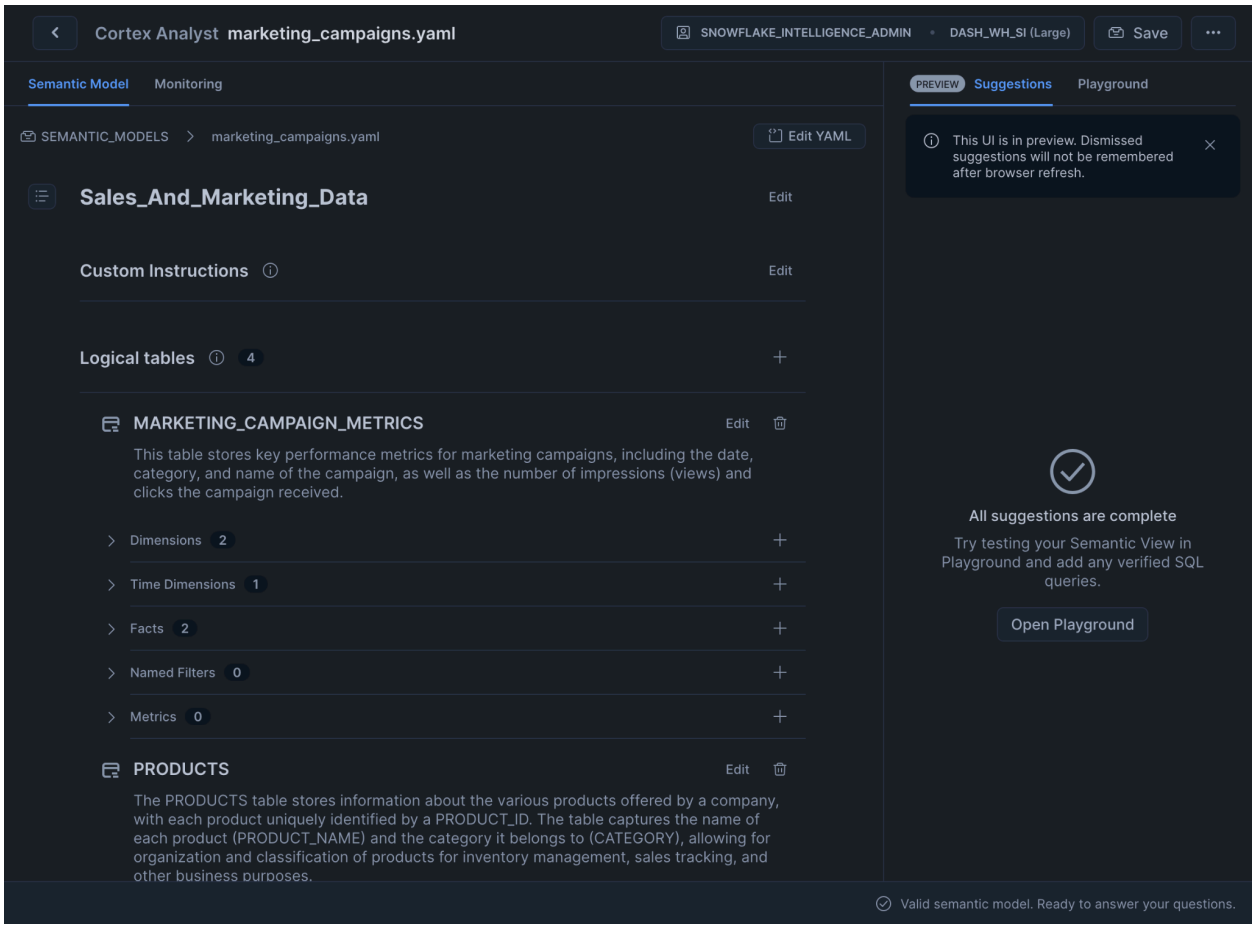
7. Click on "Create new" in the top right corner and select "Upload your YAML file"



8. Select "DASH_DB_SI.RETAIL > SEMANTIC_MODELS" for database, schema, and stage. Then select "Upload".

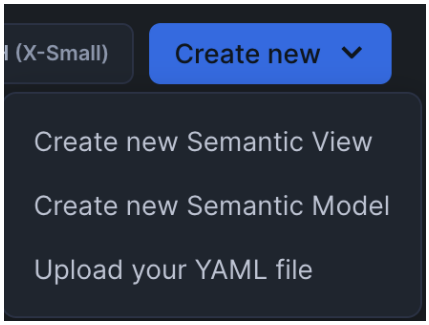


9. Scroll through and showcase the different components that the YAML file is generating



Optional: Showcase and Explain Semantic Views & Models

1. Navigate back to “AI & ML > Cortex Analyst”.

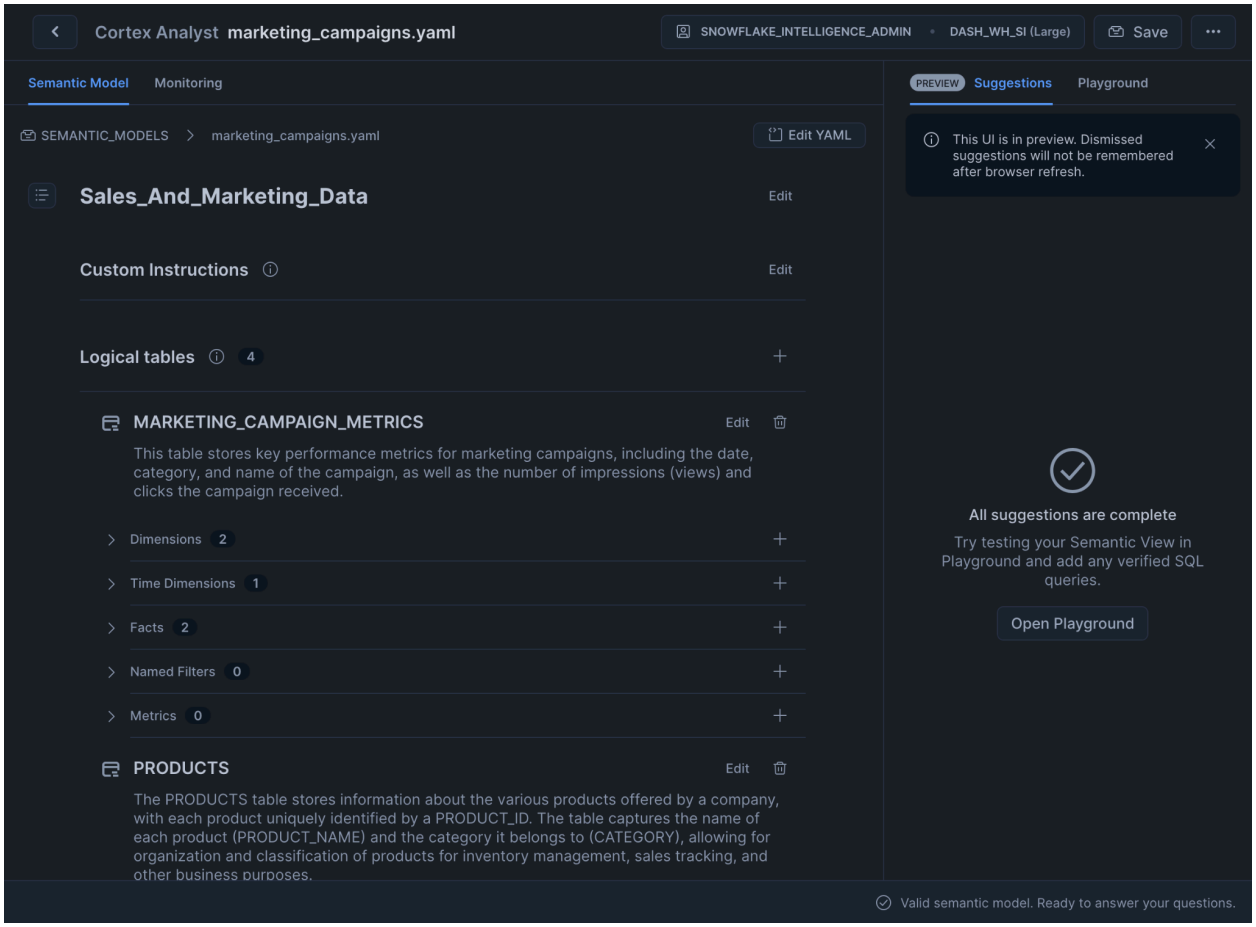


2. Showcase how you can create each one and make sure to explain:
    a. Semantic View: SQL-defined view that encapsulates business logic such joins and calculations over one or more base tables.

b. Semantic Model: Defines entities, attributes, hierarchies, and measures.
c. YAML File: Declarative file that tells Cortex Analyst how to interpret your data model for natural-language queries. Specifies tables, columns (dimensions, facts, time dimensions), synonyms, descriptions, relationships, and verified example queries. This file basically declares a Semantic Model.

Using Cortex Analyst

1. Make sure everyone is at this step before proceeding. Once the "Valid semantic model. Ready to answer your question is displayed". You are now ready to click "Open Playground" and start asking questions.



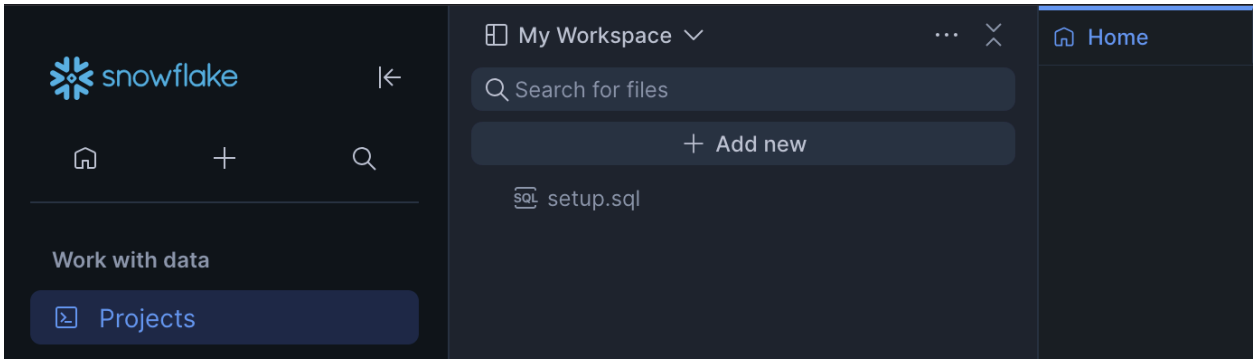
2. Now you can ask any question, here are some sample questions:
   a. Explain the dataset
   b. Show me the trend of sales by product category between June 2025 and August 2025

Module 4: User Defined Functions

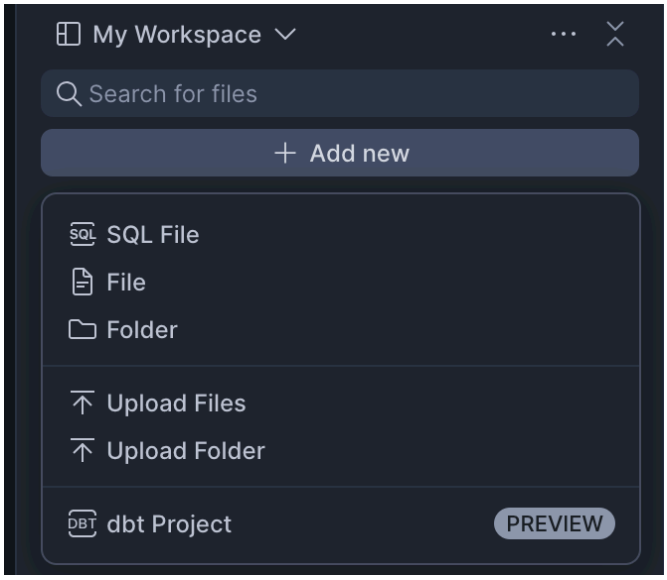
Still need to design example UDF based on scenario
https://github.com/MLH/build25-udf-udtf-examples

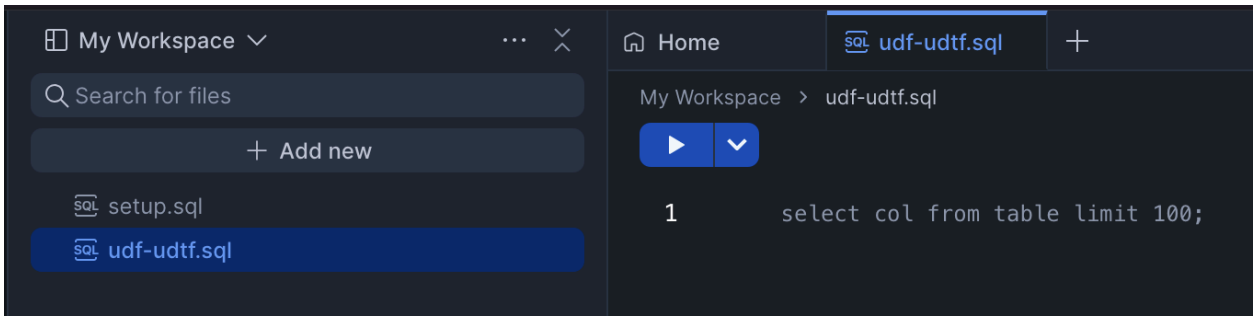
Setting Up UDF/UDTF via SQL File

1. Go to “Projects” which should take you to “My Workspace”

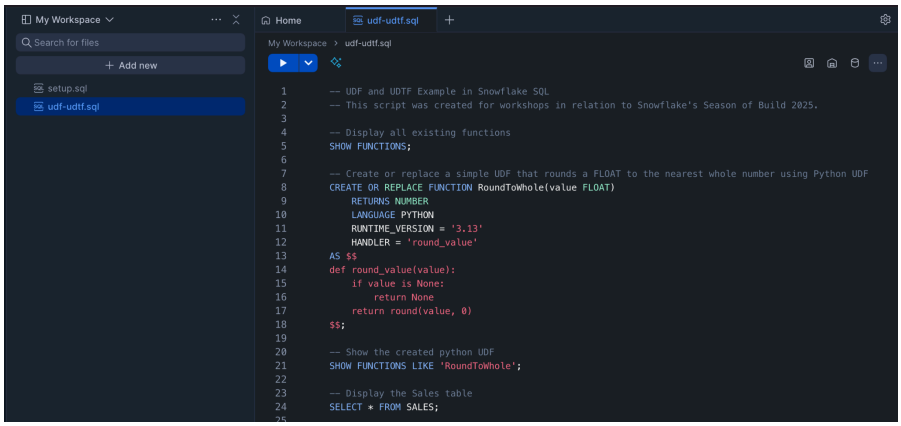


2. Click on "+ Add new" and select "SQL File" and name it `udf-udtf.sql`



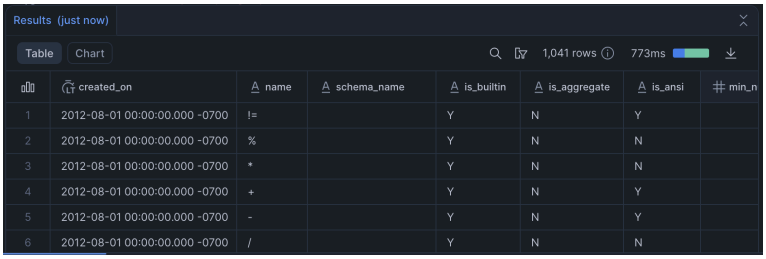


3. Navigate back to the GitHub repo, open `udf-udtf.sql`, copy its contents, and paste them into the SQL file you created in your workspace.



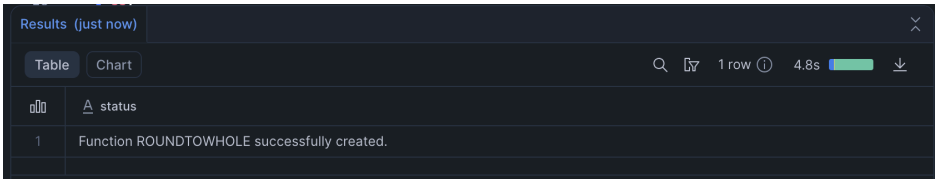
Executing and Using UDF

1. Instead of executing the whole .sql file like last time, slowly go step by step and click the play button after explaining what each of the lines is doing.
    a. 5: Displays all existing functions, which if you scroll through all of them should have `is_builtin` set to Y

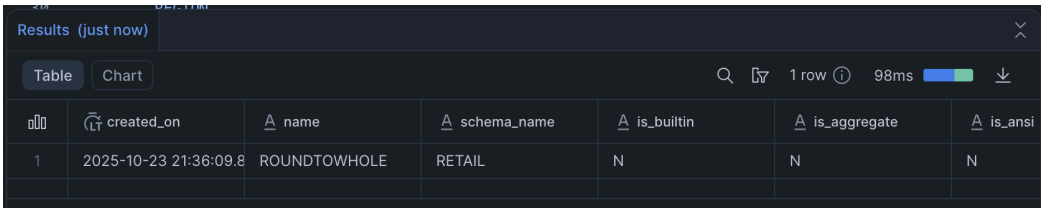


	created_on	name	schema_name	is_builtin	is_aggregate	is_ansi	min_n
1	2012-08-01 00:00:00.000 -0700	!=		Y	N	Y	
2	2012-08-01 00:00:00.000 -0700	%		Y	N	N	
3	2012-08-01 00:00:00.000 -0700	*		Y	N	N	
4	2012-08-01 00:00:00.000 -0700	+		Y	N	Y	
5	2012-08-01 00:00:00.000 -0700	-		Y	N	Y	
6	2012-08-01 00:00:00.000 -0700	/		Y	N	N	

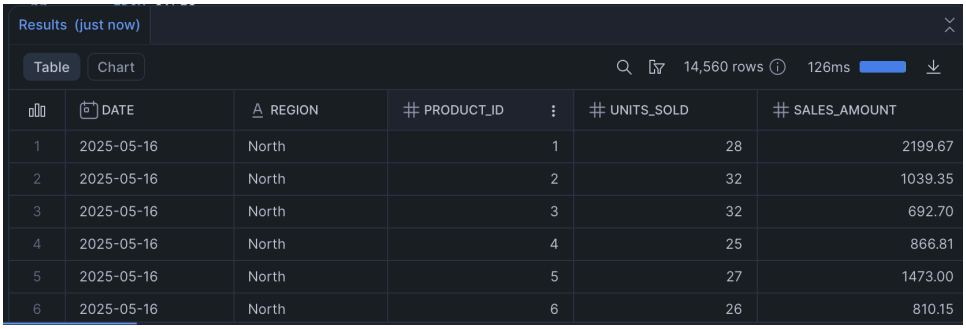
    b. 8–18: Creates a simple UDF that rounds a FLOAT to the nearest whole number using Python UDF. Call out that it took seconds and the format:
        i. Return: What should this UDF return
        ii. Language: What coding/query language is being used
        iii. Runtime Version: Which Python Runtime should Snowflake use when the Python for this function is run
        iv. Handler: What is the function's name that you will define in the UDF
        v. As: The Python code that is going to be this UDF



    c. 21: Checks to make sure in the Functions list there does exist a `RoundToWhole` UDF. Note: The `is_builtin` is set to N, scroll to the right and call out the `is_table_function` is set to N since this is a UDF, and show the language is set to `PYTHON`.



d. 24: Displays all columns from the Sales table

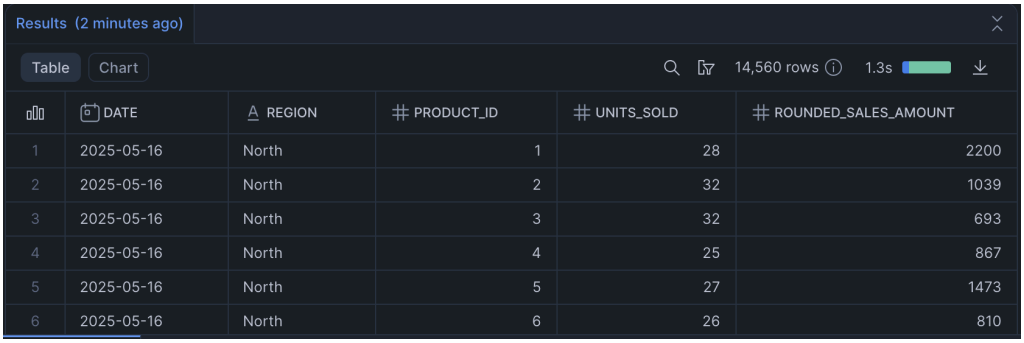


Results (just now)

Table | Chart | 14,560 rows | 126ms

	DATE	REGION	PRODUCT_ID	UNITS_SOLD	SALES_AMOUNT
1	2025-05-16	North	1	28	2199.67
2	2025-05-16	North	2	32	1039.35
3	2025-05-16	North	3	32	692.70
4	2025-05-16	North	4	25	866.81
5	2025-05-16	North	5	27	1473.00
6	2025-05-16	North	6	26	810.15

e. 27–38: Using the new UDF that we create, display the Sales table with the UDF applied to the `Sales_Amount`. Note: You should see all the results rounded.

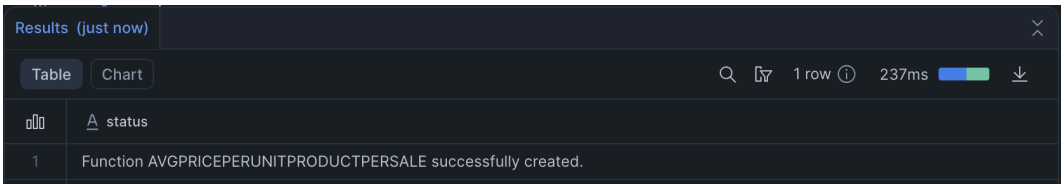


Results (2 minutes ago)

Table | Chart | 14,560 rows | 1.3s

	DATE	REGION	PRODUCT_ID	UNITS_SOLD	ROUNDED_SALES_AMOUNT
1	2025-05-16	North	1	28	2200
2	2025-05-16	North	2	32	1039
3	2025-05-16	North	3	32	693
4	2025-05-16	North	4	25	867
5	2025-05-16	North	5	27	1473
6	2025-05-16	North	6	26	810

f. 42–62: Create a simple UDTF that will create a new column that stores the average price per unit of product for each sale record within the Sales table. Note: Call out how the SQL UDTF only took milliseconds compared to the seconds for the Python UDF.

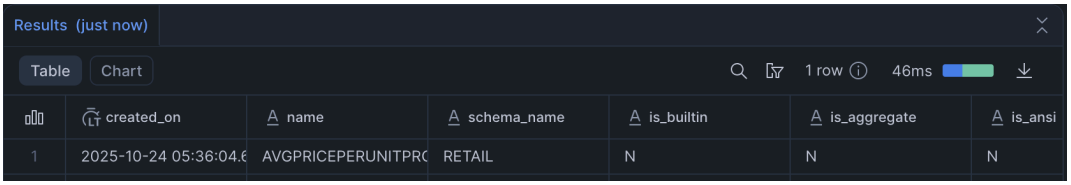


Results (just now)

Table | Chart | 1 row | 237ms

	status
1	Function AVGPRICEPERUNITPRODUCTPERSALE successfully created.

g. 65: Checks to make sure in the Functions list there does exist a `AvgPricePerUnitProductPerSale` UDTF. Note: The `is_builtin` is set to N, scroll to the right `is_table_function` is set to Y since this is a UDTF, and show the language is set to SQL.



Results (just now)

Table | Chart | 1 row | 46ms

	created_on	name	schema_name	is_builtin	is_aggregate	is_ansi
1	2025-10-24 05:36:04.6	AVGPRICEPERUNITPRC	RETAIL	N	N	N

h. 68: Show the simple UDTF in action by producing a table with avg price per unit of a product for each sale record. What do you notice that is strange about those numbers? (Hint: Do prices for products ever stay the same?)

Results (just now)

Table | Chart — 14,560 rows — 311ms

	DATE	REGION	PRODUCT_ID	UNITS_SOLD	SALES_AMOUNT	AVG_PRICE_PER_UNIT
1	2025-06-03	East	1	26	1184.81	45.56961538
2	2025-07-16	East	1	31	1478.08	47.68000000
3	2025-05-21	East	1	25	1037.07	41.48280000
4	2025-08-02	East	1	33	900.52	27.28848485
5	2025-07-15	East	1	31	1167.76	37.66967742
6	2025-07-29	East	1	36	1561.95	43.38750000

i. 71–76: Using both the UDF and UDTF, create a view that leverages the `AvgPricePerUnitProductPerSale` UDTF to generate a table. Then, apply the `RoundToWhole` UDF to the `avg_price_per_unit` column from that table, creating a `rounded_avg_price_per_unit`.

Results (just now)

Table | Chart — 1 row — 132ms

	status
1	View AVG_PRICE_PER_UNIT_PRODUCT_PER_SALE successfully created.

j. 78–85: Creates a new table `PRODUCTS_WITH_AVG_PRICE` that joins the `PRODUCTS` table with the average price, which is the average of the rounded average prices of each product per sale for all regions from the VIEW that we created.

Results (just now)

Table | Chart — 1 row — 2.0s

	status
1	Table PRODUCTS_WITH_AVG_PRICE successfully created.

k. 88: Sets the status to show that UDF/UDTF creation and usage was successful

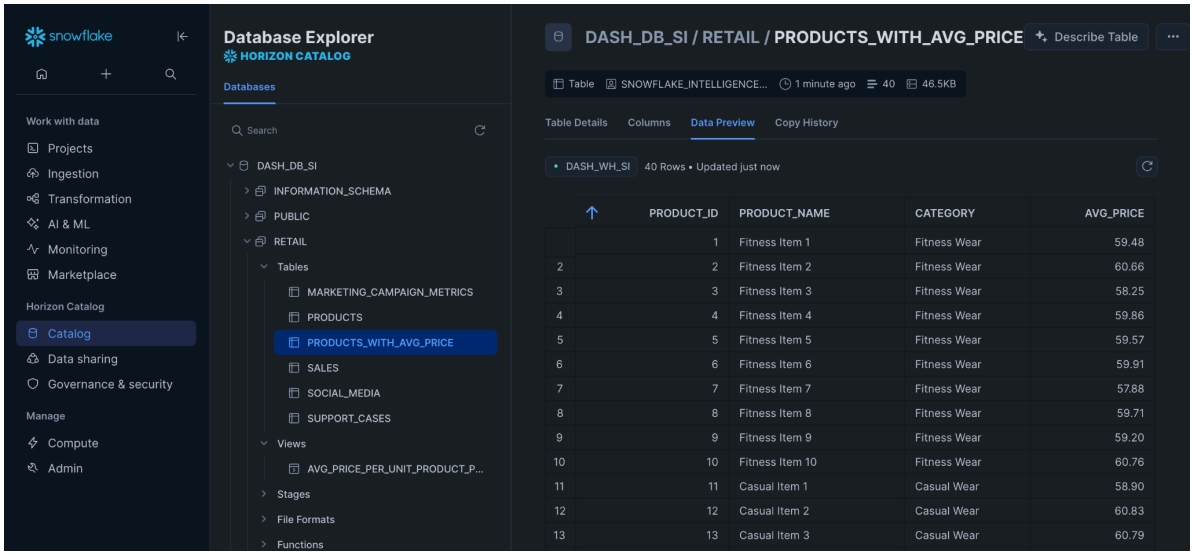
Results (1 minute ago)

Table | Chart — 1 row — 43ms

	STATUS
1	UDF and UDTF creation and usage completed successfully!

Verifying the Results of Running the UDF/UDTF

1. Now, let's go verify the UDF/UDTF script ran successfully by going to “Catalog” and checking for “DASH_DB_SI/RETAIL” DB. This should list a new table called `PRODUCTS_WITH_AVG_PRICE` with 40 rows showing all the products with the average price from all sales and a new view called `AVG_PRICE_PER_UNIT_PRODUCT_PER_SALE`.



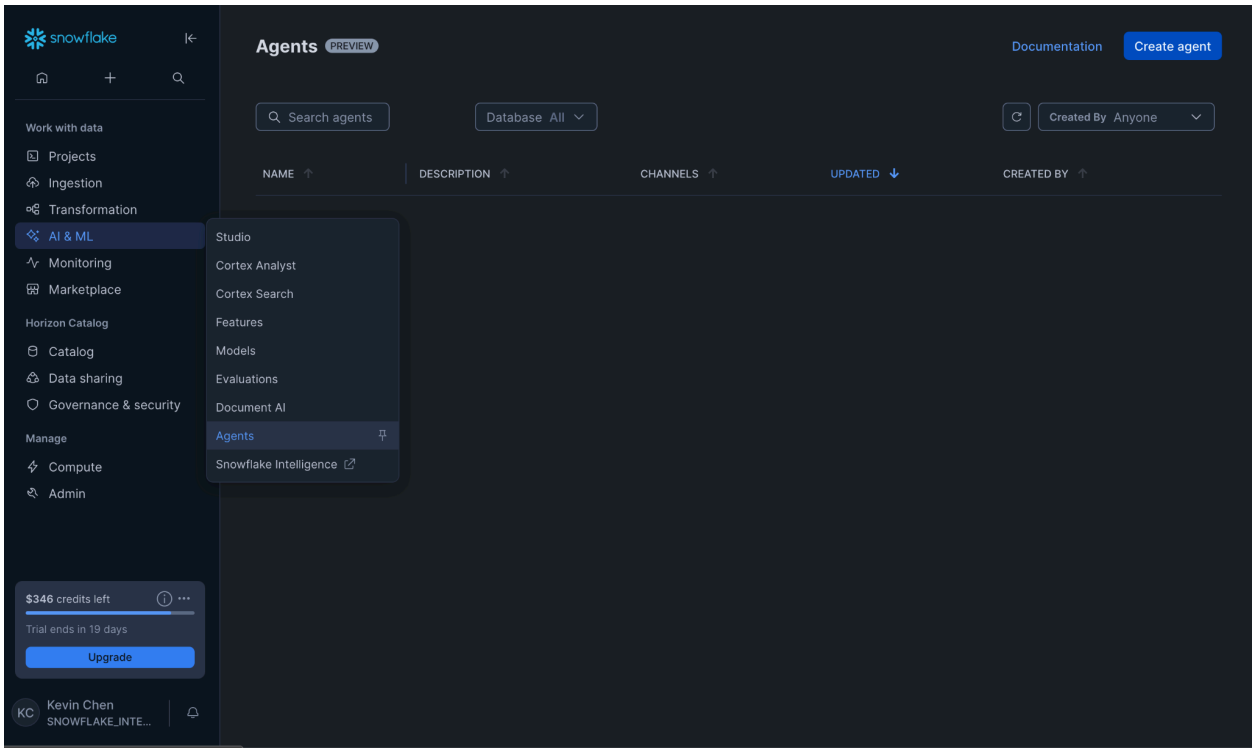
2. The results indicate that the average price across all products is relatively consistent.

Module 5: Snowflake Intelligence

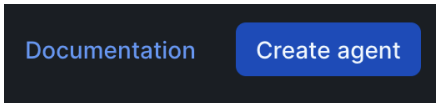
Setting Up Custom Snowflake Intelligence

Creating an Agent

1. Navigate to “AI & ML > Cortex Agent”

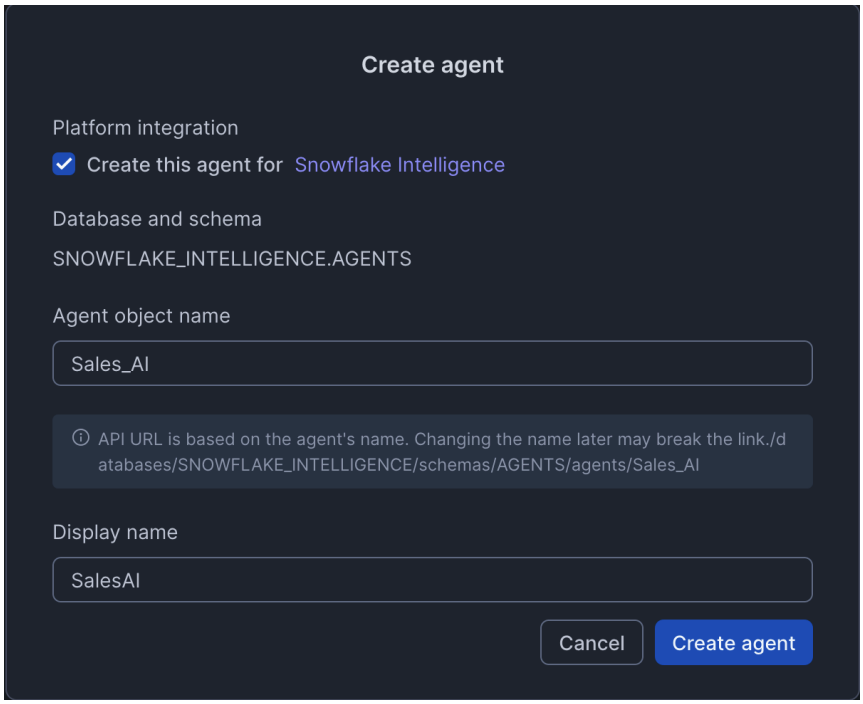


2. Click “Create agent” in the top right corner

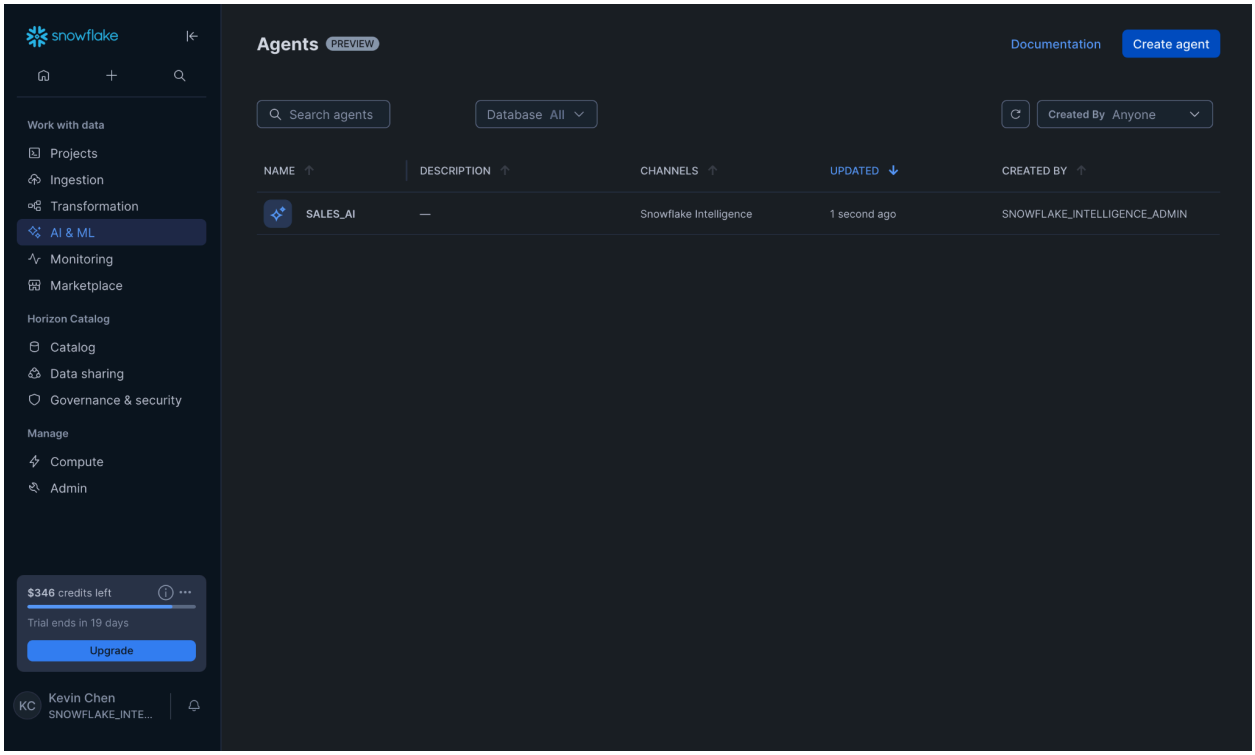


3. Check the box for "Create this agent for Snowflake Intelligence". The default Database and schema should be set to `SNOWFLAKE_INTELLIGENCE.AGENTS`. For Agent object name, enter `Sales_AI`, this is the internal identifier Snowflake uses to organize your agent's metadata. For Display name, choose any user-facing name

you prefer; we'll use `SalesAI` as the name shown to users.

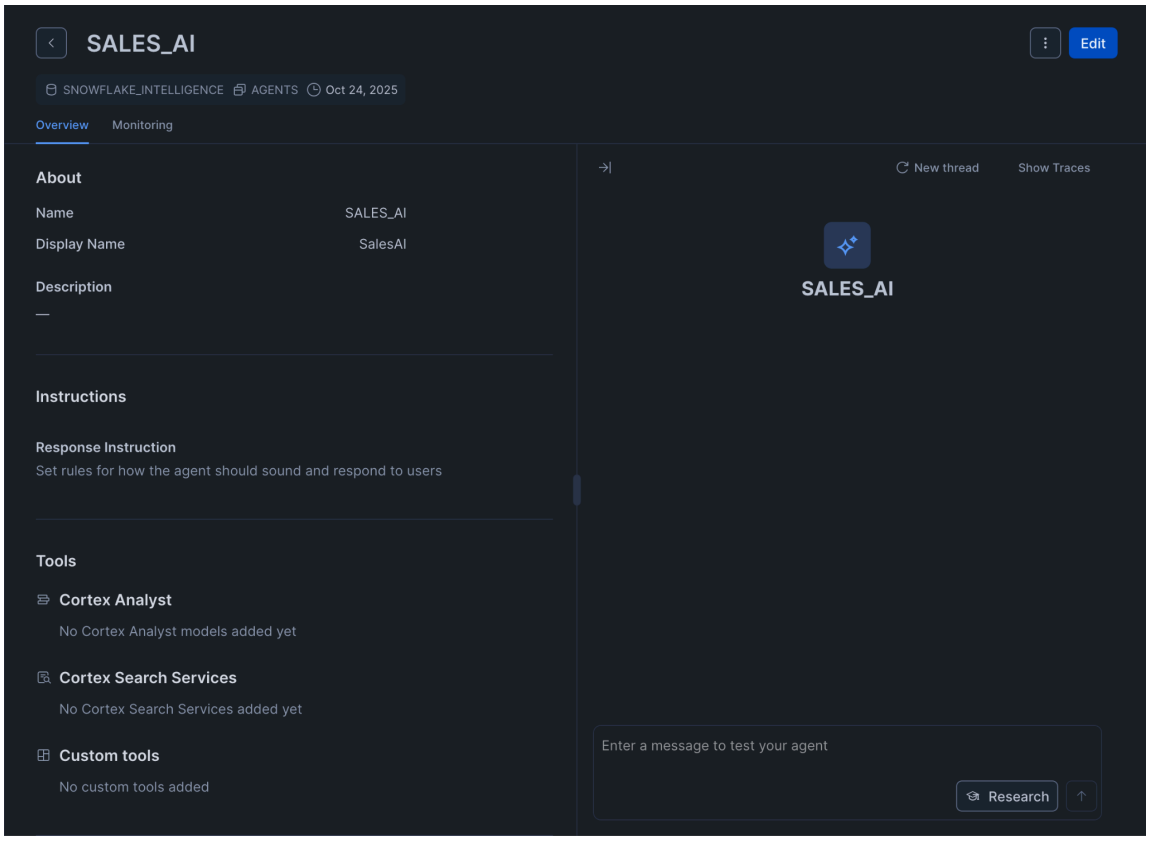


4. Select the Sales_AI Agent from the list to open and configure the Agent.



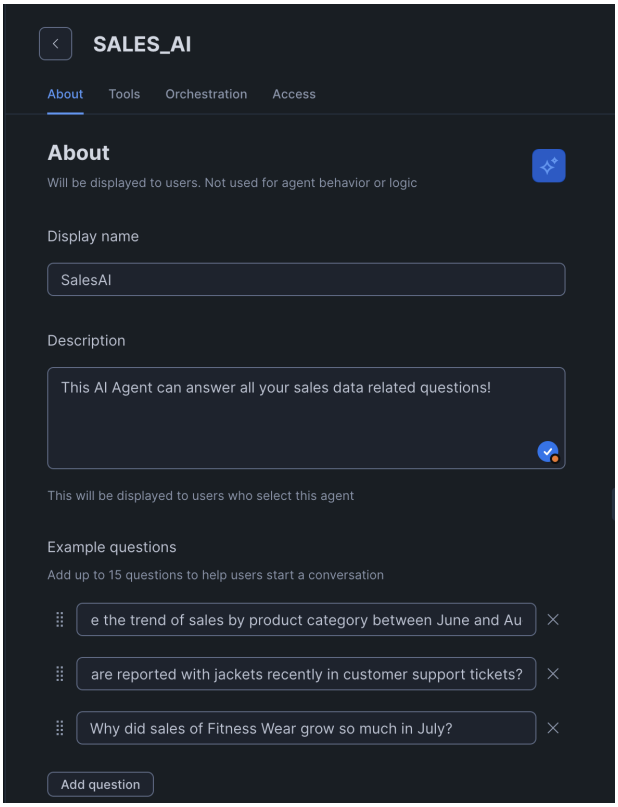
5. You can now view the current `Sales_AI` configuration, which displays general information, instructions, tools, and a preview test chat interface. To customize your

Agent, click the “Edit” button in the top right corner.

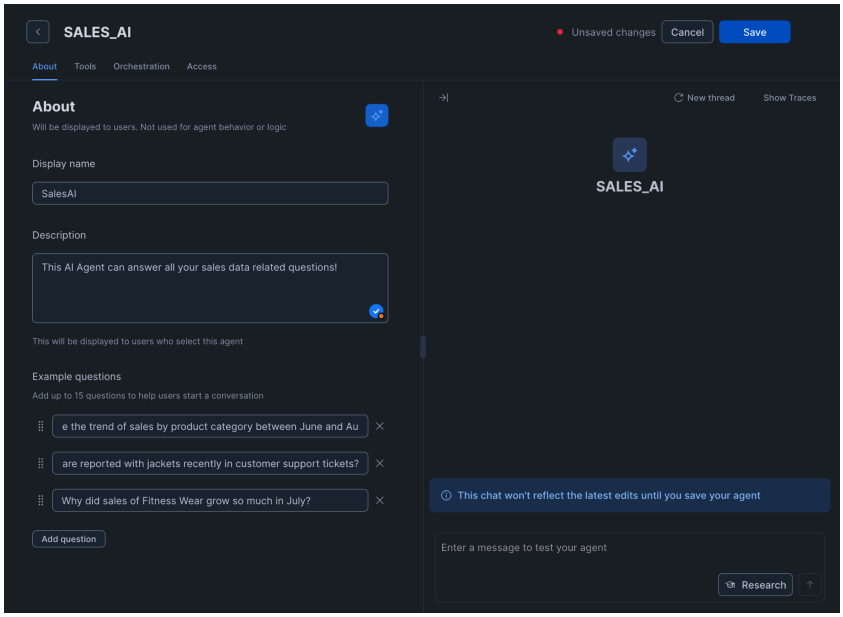


6. In this step, configure the essential details for your Agent. Optionally, add a description to clarify its purpose or scope. To guide users and spark ideas, enter a few sample questions that demonstrate the types of queries the Agent can answer. Examples:
    a. Show me the trend of sales by product category between June and August
    b. What issues are reported with jackets recently in customer support tickets?

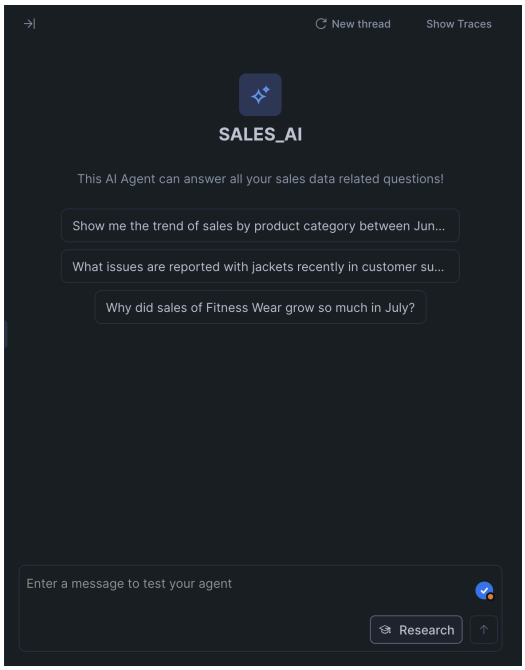
c. Why did sales of Fitness Wear grow so much in July?



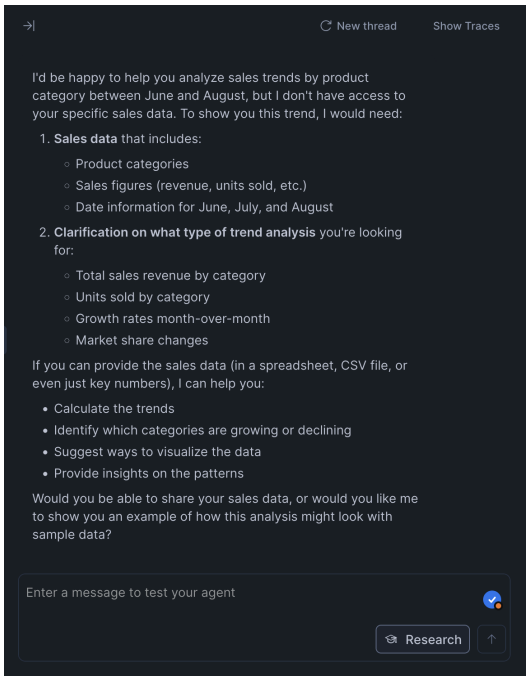
7. After completing all fields, the changes won't appear until you save them. Click the "Save" button in the top right corner to persist your Agent configuration.



8. Once you click “Save”, the preview test chat interface will reflect all the configuration details you've set for the Agent.

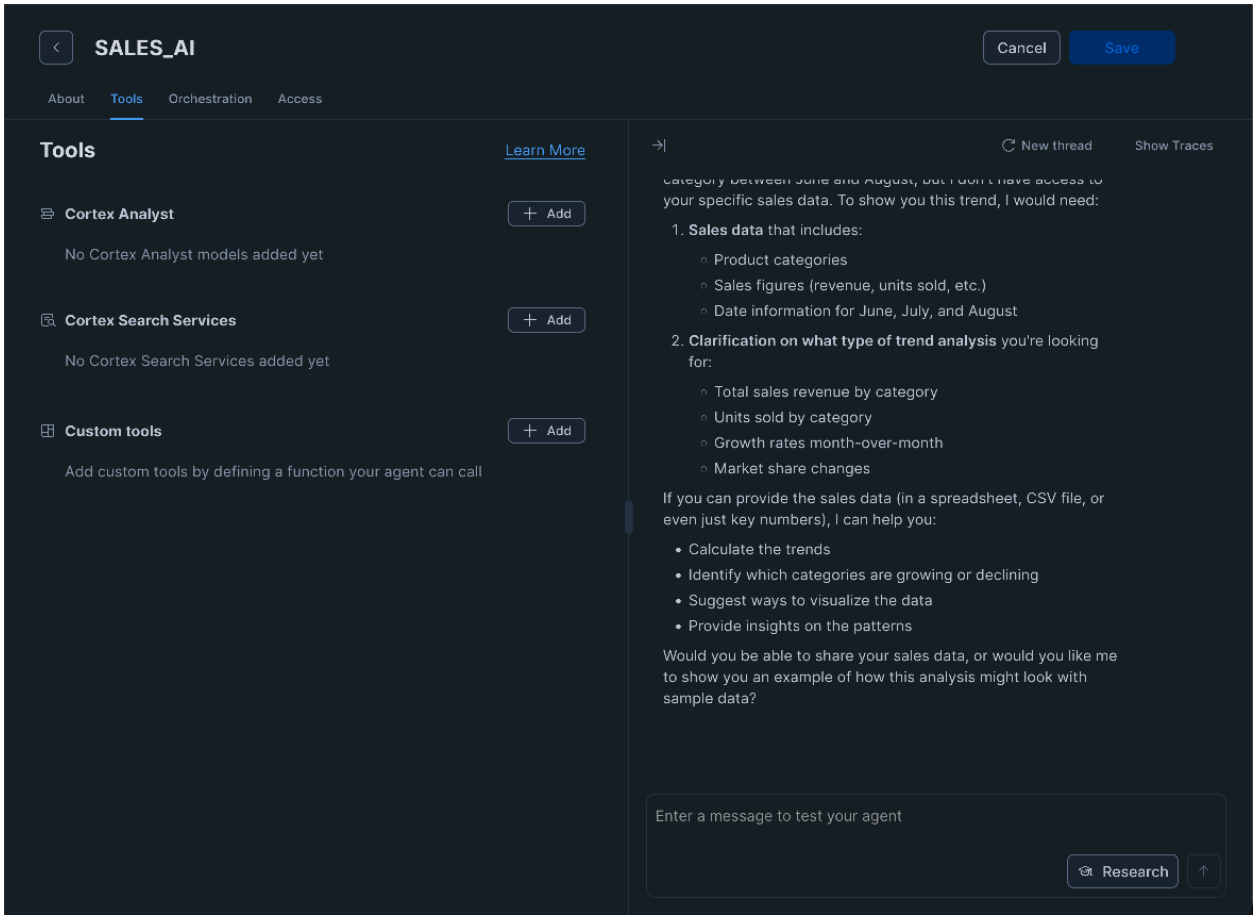


9. Select one of the sample questions to test the Agent. You'll notice it responds that it lacks sufficient data or information to fulfill the request. Consider why this might occur? (Hint: Think about components that enable Snowflake Intelligence to function)

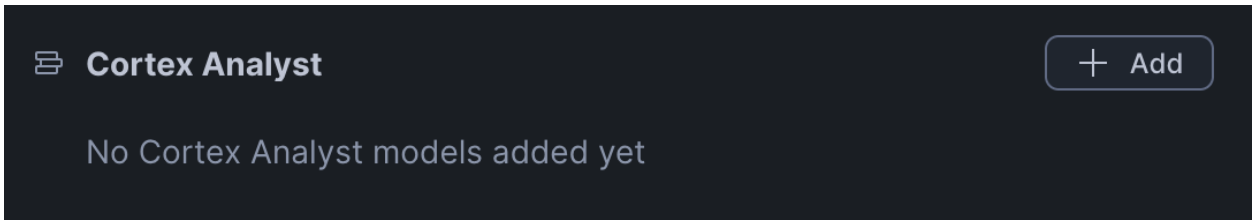


Setting up the Tools

1. Click “Tools” in the top left corner to configure the capabilities available to your AI Agent. Here you can integrate tools including Cortex Analyst for data analysis, Cortex Search for information retrieval, and Custom Tools such as UDFs for specialized functionality.

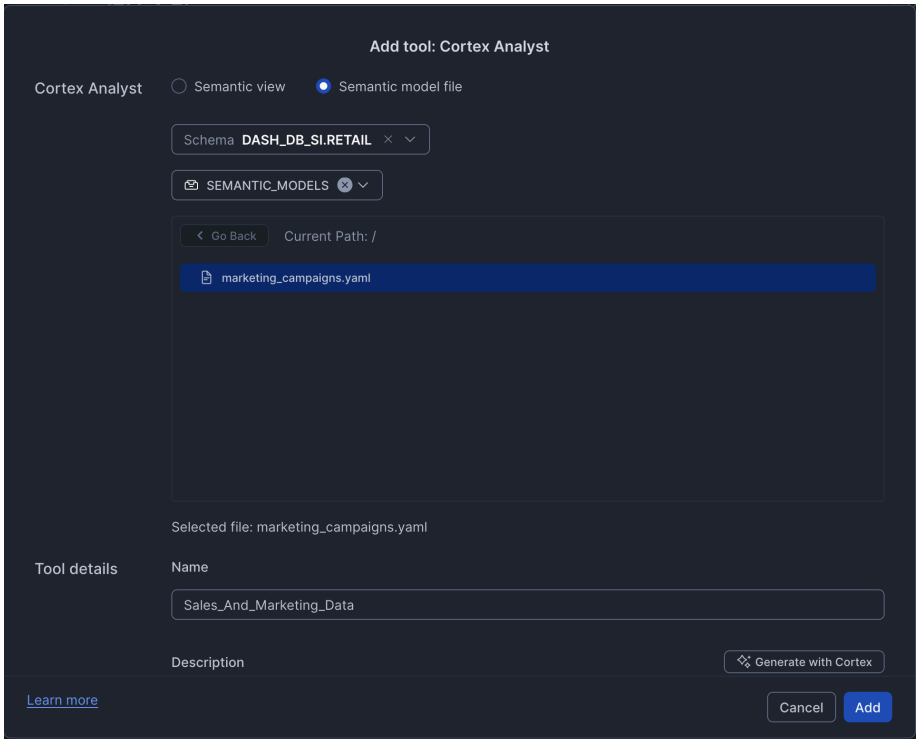


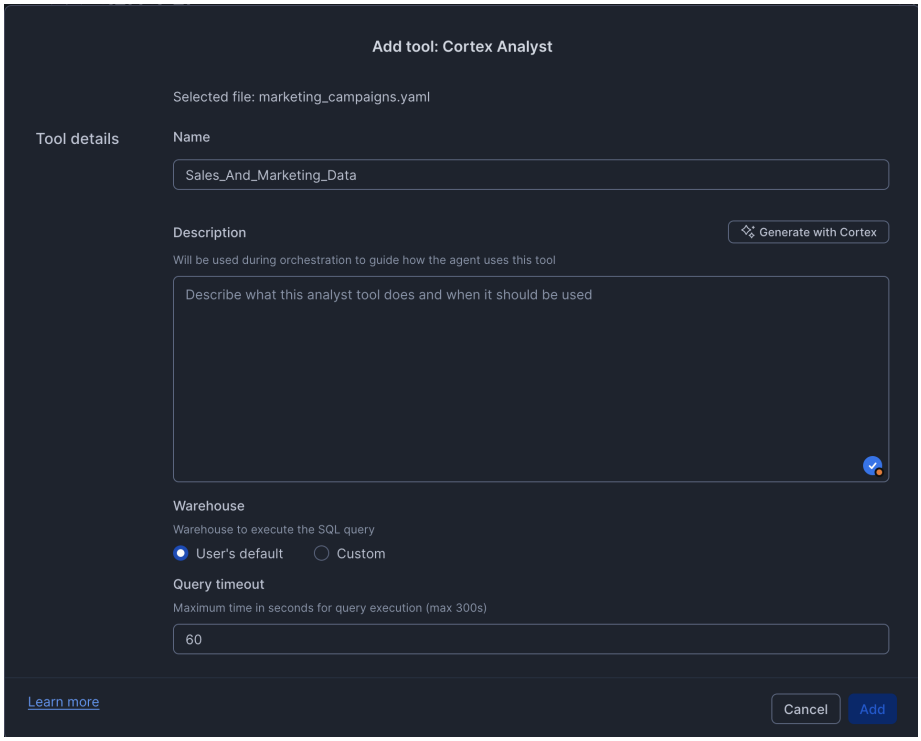
2. Begin by adding the Cortex Analyst configured in Module 3 to your Agent's available tools. Click the “+ Add” button.



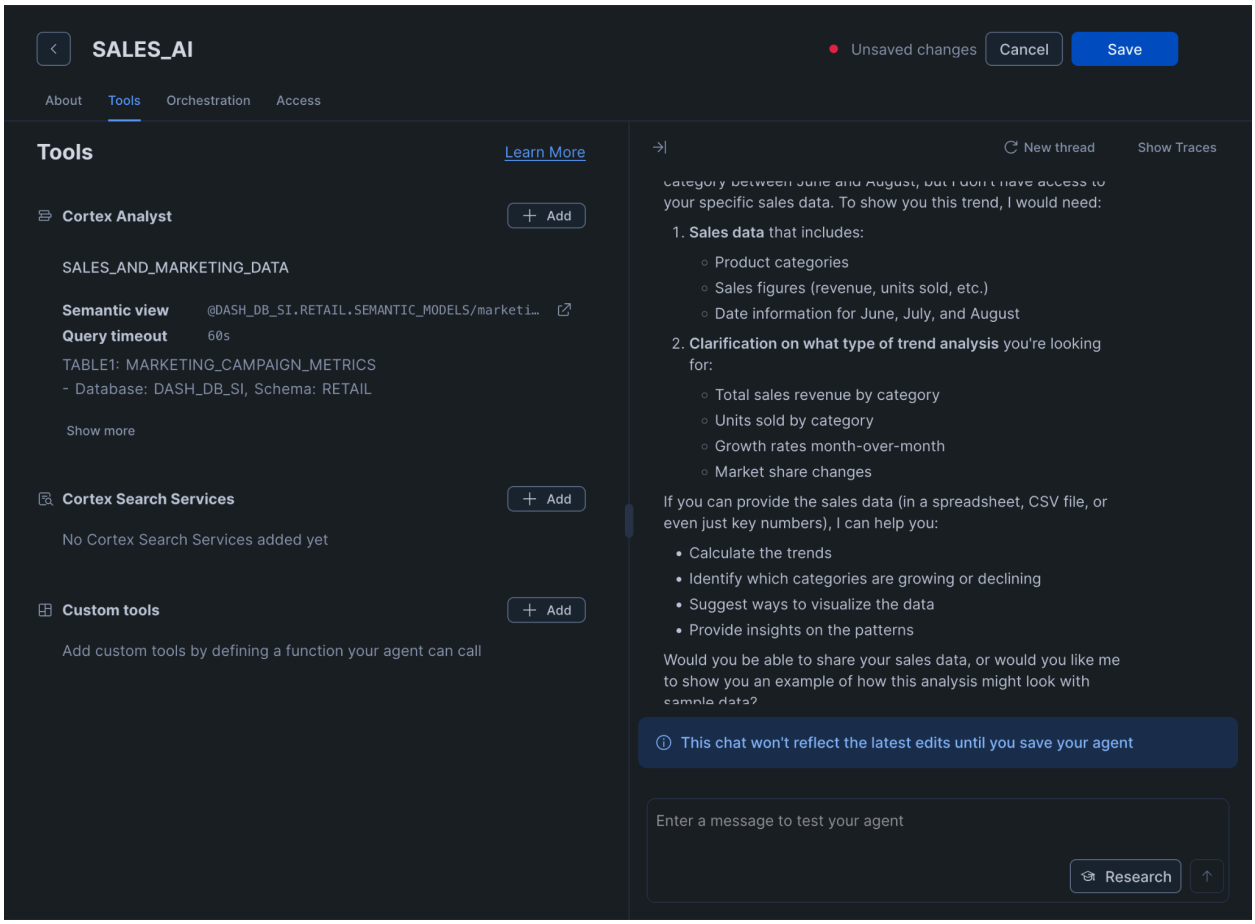
3. Select the “Semantic model file radio” button. Navigate to and select `DASH_DB_SI.RETAIL.SEMANTIC_MODELS`, then choose the `marketing_campaigns.yml` file. Enter `Sales_And_Marketing_Data` as the tool

name. For the description, click “Generate with Cortex” to automatically create a description that will guide the Agent during orchestration. Set the “Query timeout” to 60 seconds, then click “Add” in the bottom right corner.

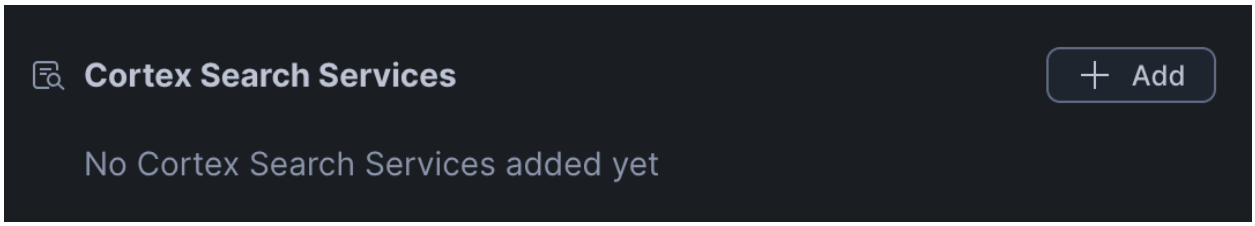




4. You should now see that under Cortex Analyst you have the instance configure for the Agent to use

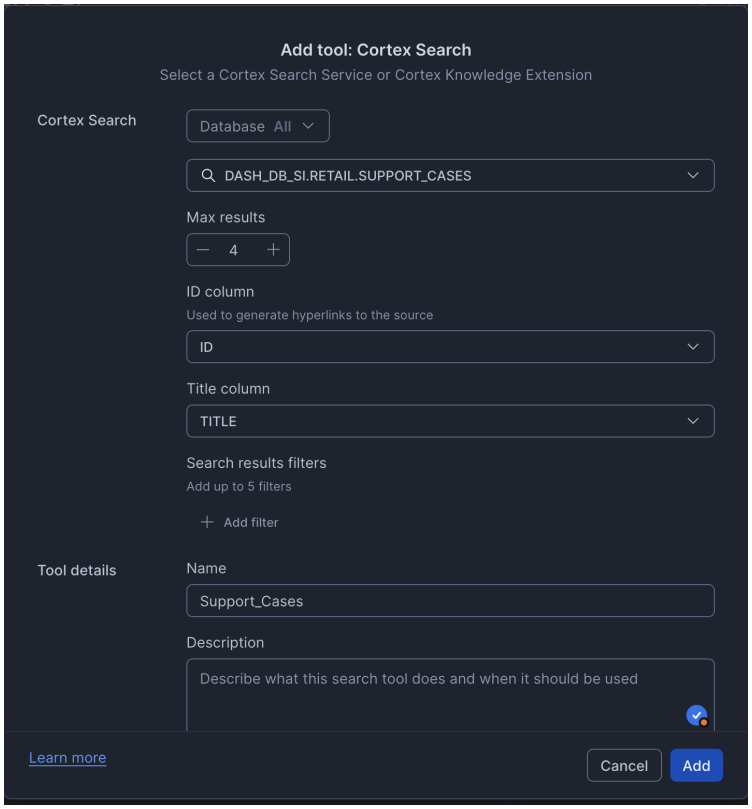


5. Next, we are going to begin adding the Cortex Search configured in Module 2 to your Agent's available tools. Click the "+ Add" button.

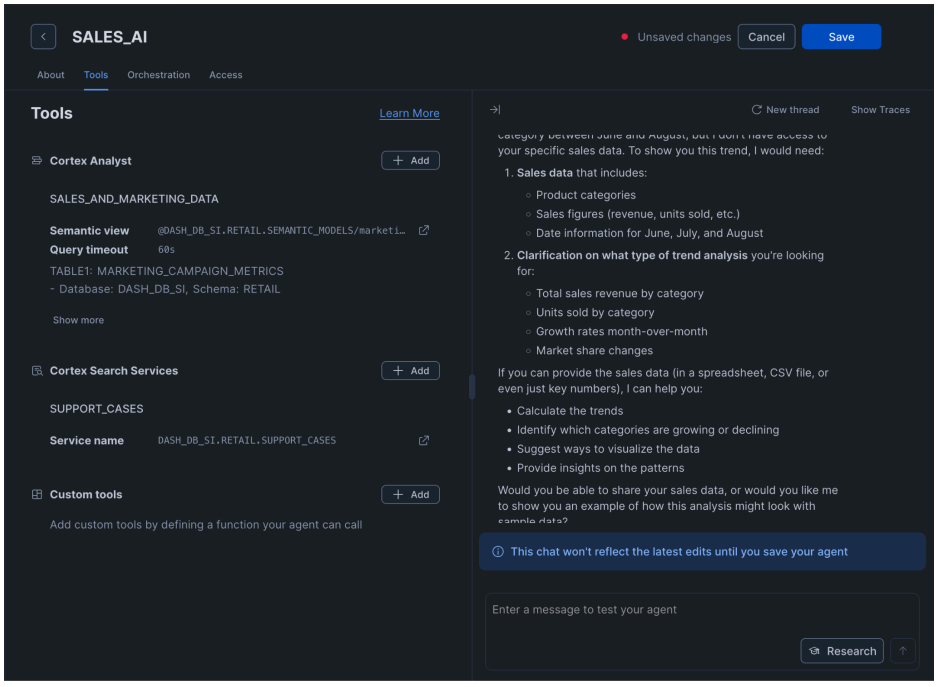


6. Select `DASH_DB_SI.RETAIL.SUPPORT_CASES`. Update the "Name" to `Support_Cases`, the "ID column" to `ID`, and the "Title column" to `TITLE`. Leave all

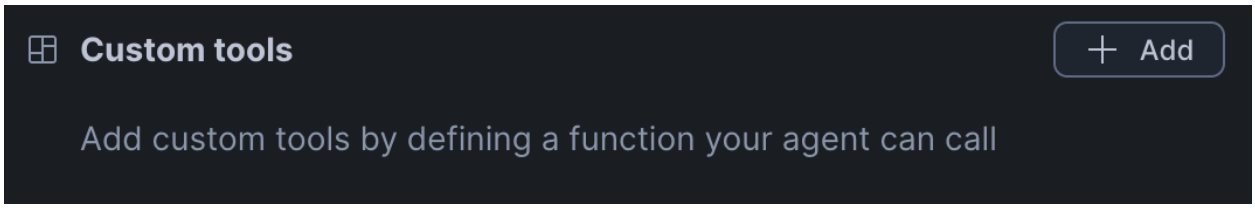
other settings unchanged and click Add in the bottom right corner.



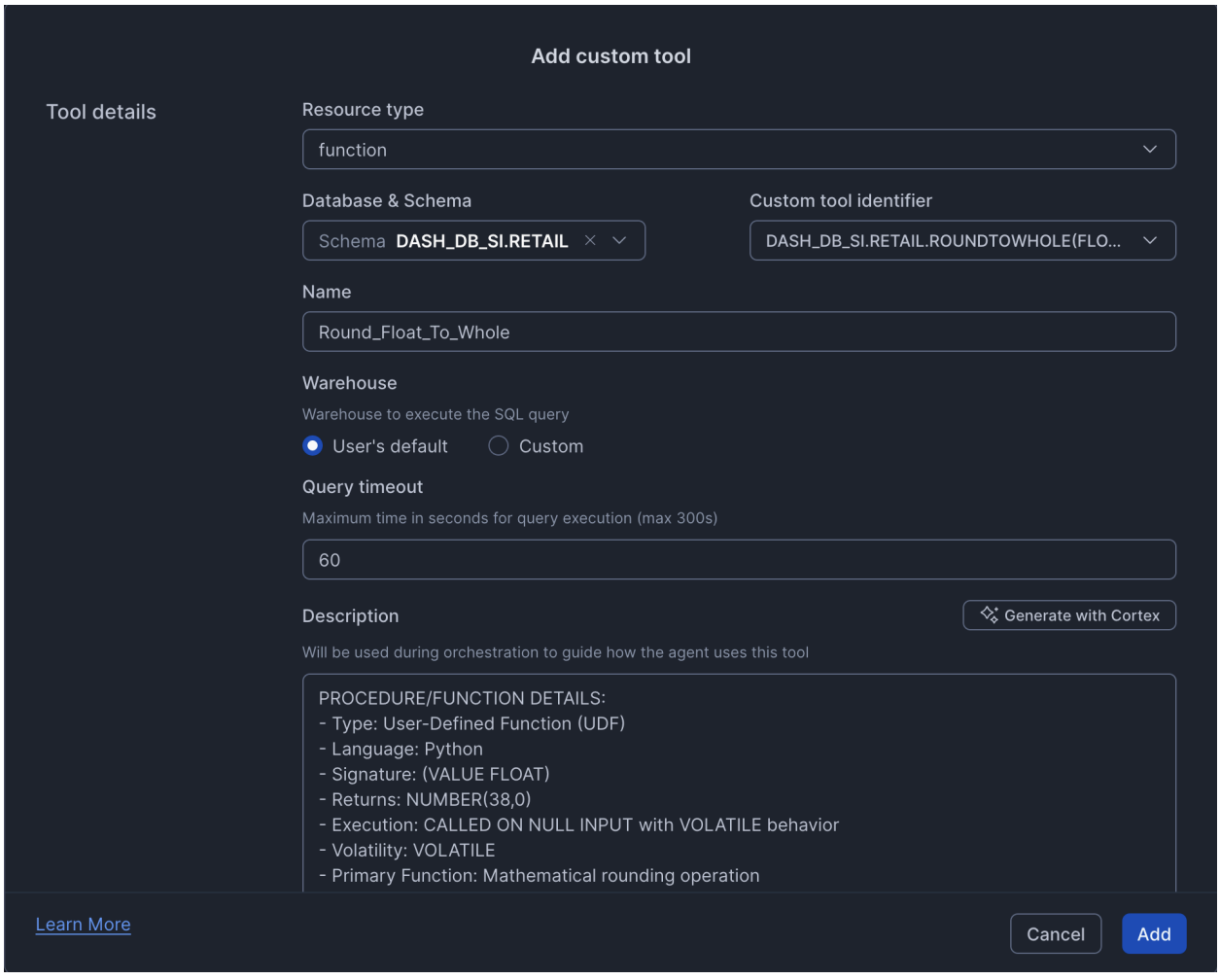
7. You should now see that under Cortex Search you have the instance configure for the Agent to use



8. Lastly, we are going to add the UDF and UDTF that we created in Module 4 as Custom tools for the AI Agent to use. Click the "+ Add" button.

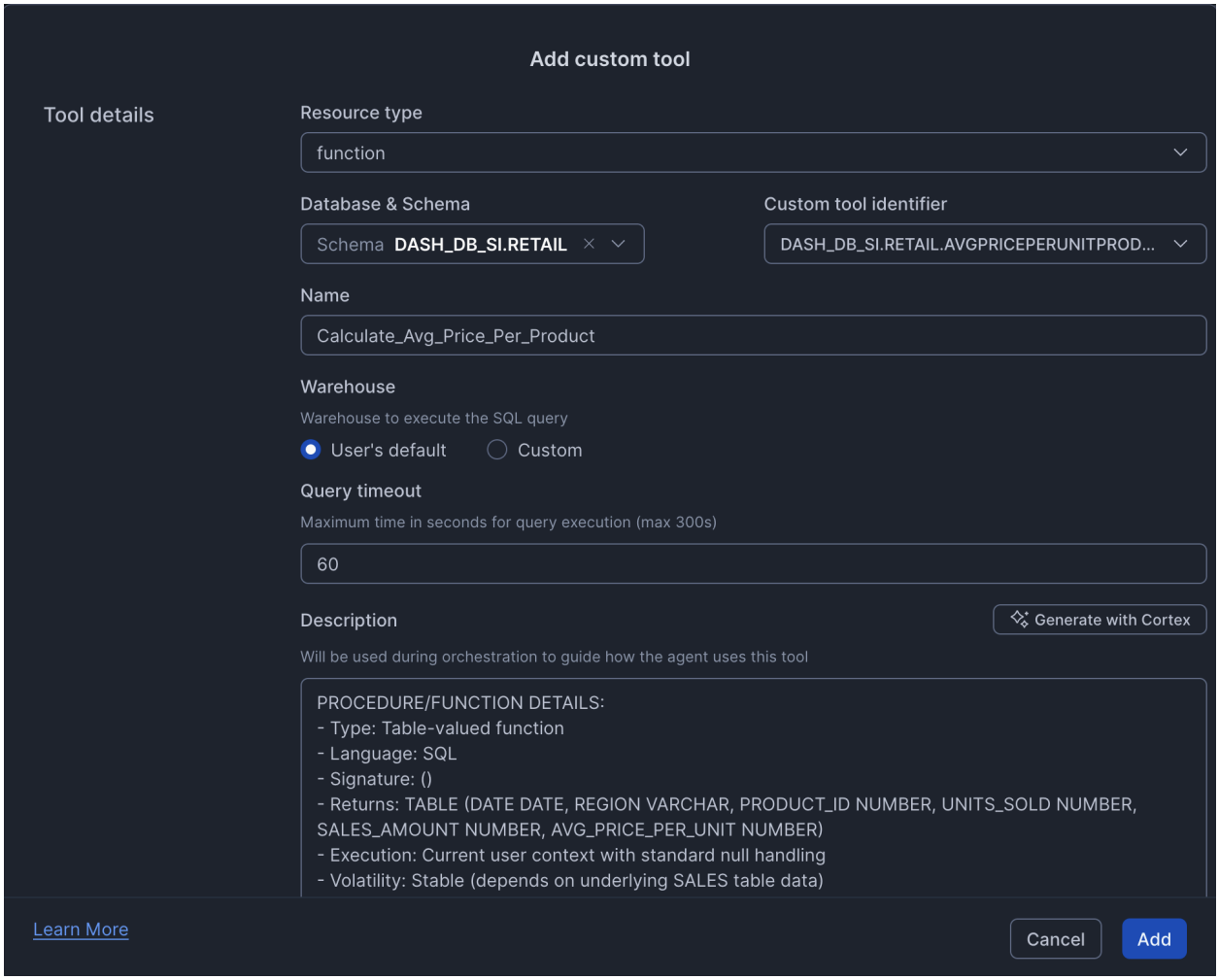


9. Select `function` as the "Resource type". Choose `DASH_DB_SI.RETAIL` as the "Database & Schema", then select the `RoundToWhole` UDF from Module 4. Name the tool `Round_Float_To_Whole` and set the "Query timeout" to 60 seconds. Click "Generate with Cortex" to automatically create the "Description", then click Add in the bottom right corner.

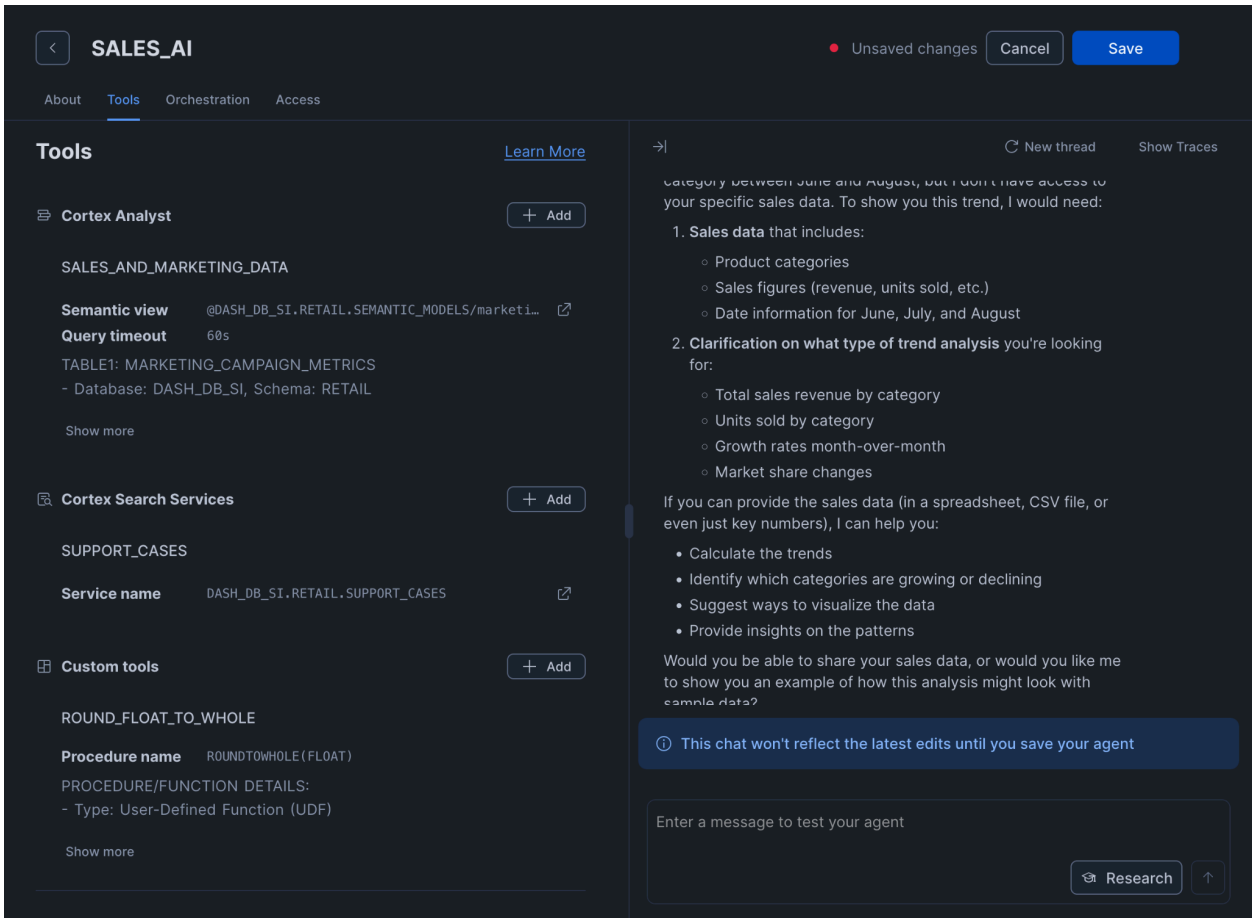


10. Let's now add other UDTF as a tool as well, so click the "+ Add" button again. Select `function` as the "Resource type". Choose `DASH_DB_SI.RETAIL` as the "Database &

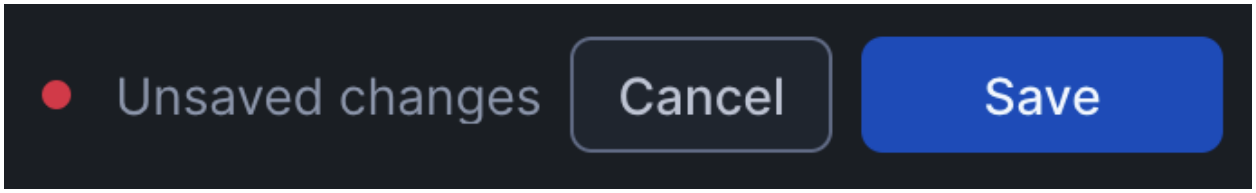
Schema", then select the `AvgPricePerUnitProductPerSale` UDTF from Module 4. Name the tool `Calculate_Avg_Price_Per_Product` and set the "Query timeout" to 60 seconds. Click "Generate with Cortex" to automatically create the "Description", then click Add in the bottom right corner.

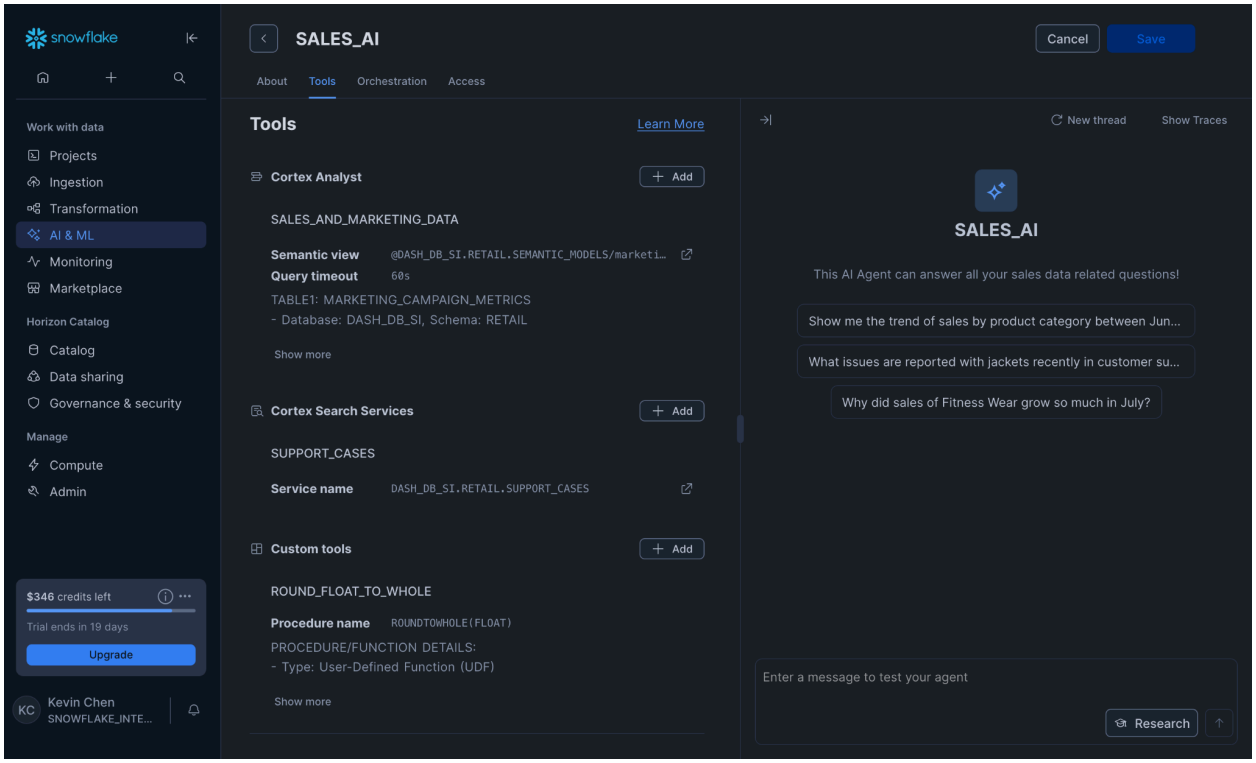


11. You should now see that under “Custom tools” you have both instance of the UDF/UDTF configure for the Agent to use



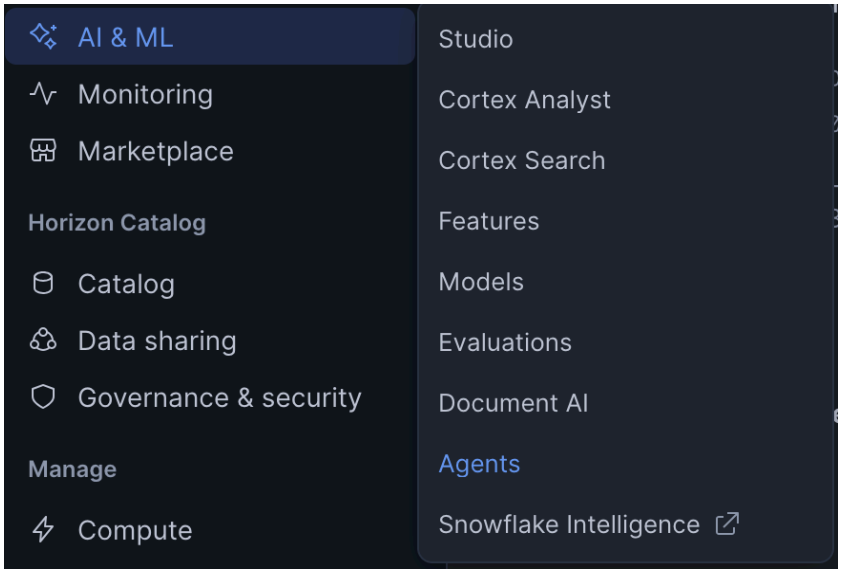
12. Click the Save button to persist all the tool configurations you've just added.



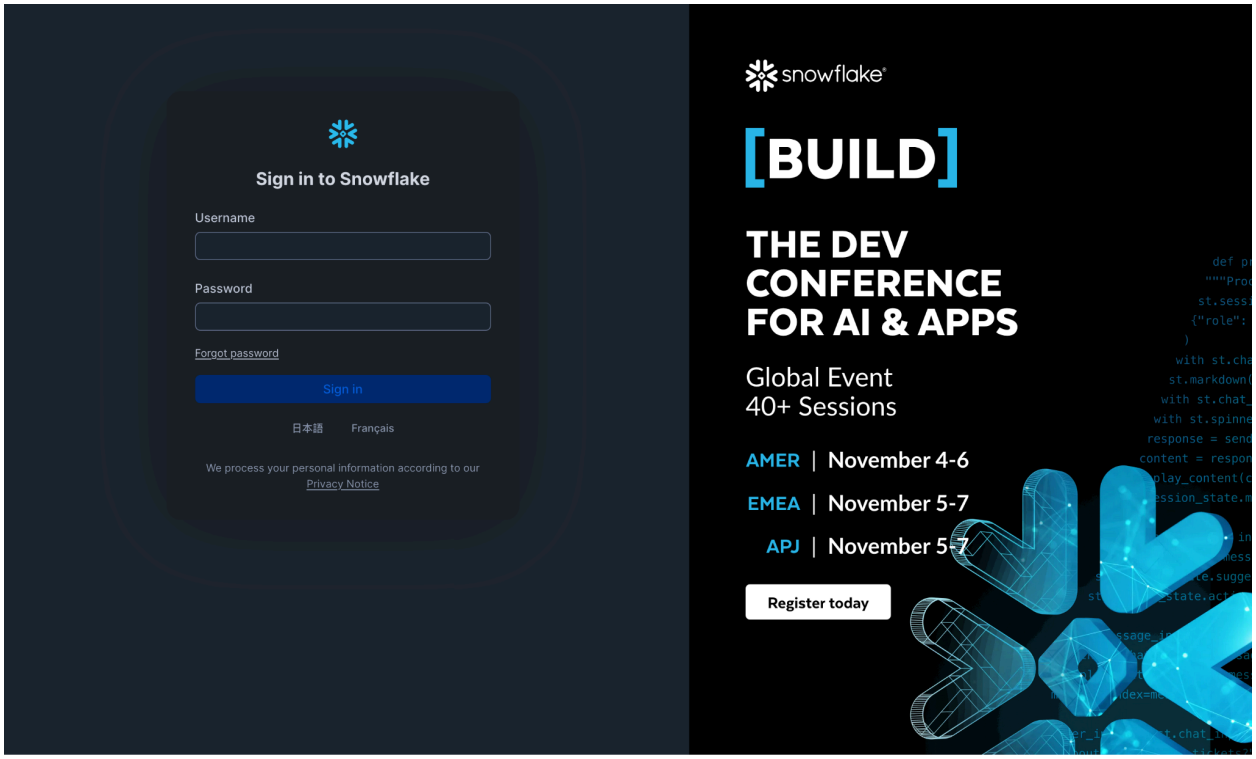


Using Custom Snowflake Intelligence Agent

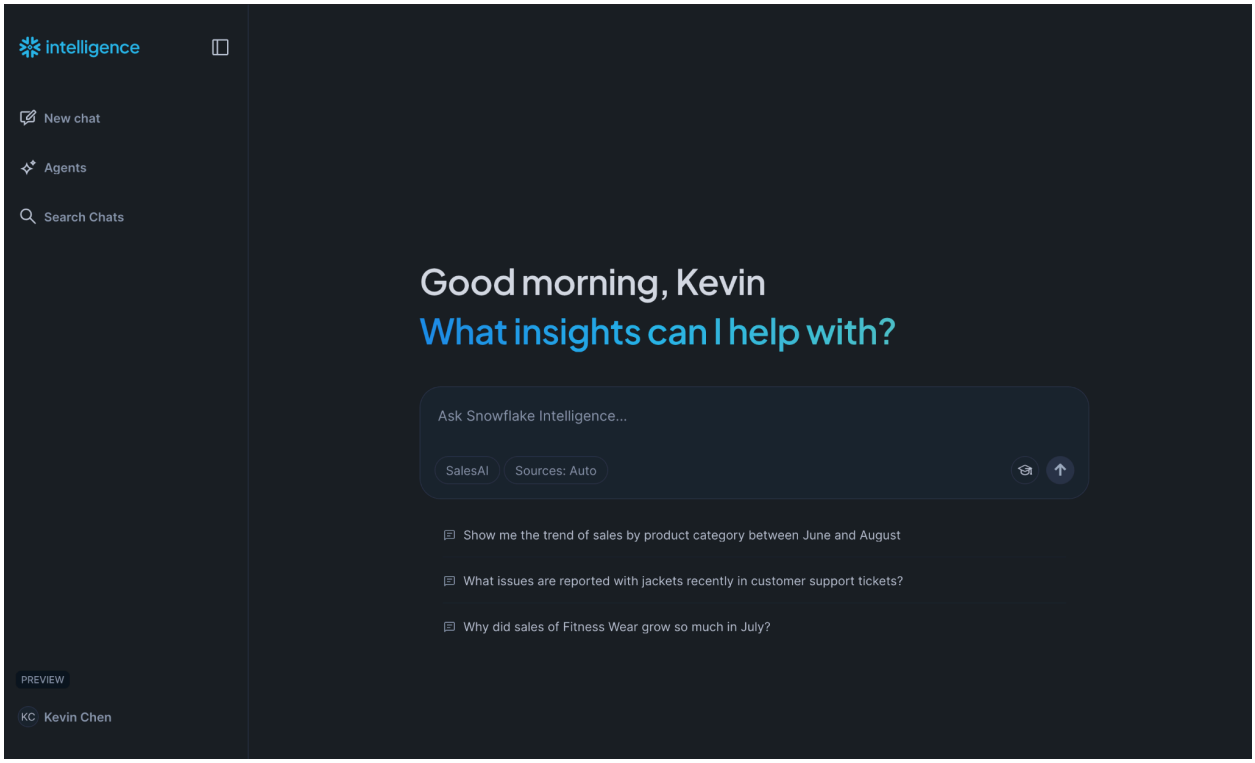
1. You're ready to test and utilize the Snowflake Intelligence Agent. Navigate to “AI & ML > Snowflake Intelligence”.



2. This navigates you to the Snowflake Intelligence interface for your account. Sign in using your existing user credentials.



3. After logging in, you'll see a familiar chat-like interface similar to standard AI Agent tools. Here you can test the three sample questions you configured and observe the Agent in action.



4. Below are some additional questions that you can use to test the different tools that the AI Agent has access to:
    a. (Use UDF/UDTF) What is the avg price for all products?
    b. (Use UDF/UDTF) When does each product sell for the lowest price?
    c. Which product categories perform best on social media?
    d. What's the relationship between social media mentions and sales?
    e. How do different regions respond to marketing campaigns?

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