

Snowflake Intelligence HOL

Snowflake Intelligence, a new application, empowers users to create and interact with data agents. These agents can access data from platforms like Snowflake and Salesforce to generate visualizations and simple applications and facilitate action on insights. Powered by Cortex LLM functions, Analyst, and Search, Snowflake Intelligence offers a curated, enterprise-ready application experience.

The ideal customer profile for a Snowflake Intelligence user is a business user, such as a sales rep, executive, or marketing manager, who is looking for insights and to take action based on data in their organization. Traditionally, they would have relied on BI applications or manually navigated file search (like SharePoint and Google Drive) and taken manual actions that can now be enhanced and automated with data agents.

Snowflake Intelligence is currently under development, and while some workflows in this lab may evolve as we approach production and refine our strategy, the foundational elements like semantic models, agents, and search services will persist. This lab will be updated periodically to reflect the product's progress. It represents the state of Snowflake Intelligence leading up to Summit 2025, as we aimed to provide this training beforehand.

What you'll do:

It is recommended that you **download** this document from Compass and run it from your GDrive. That will make copying and pasting a lot easier, as sometimes formatting can go wrong when copying and pasting directly from Compass.

This lab explores accessing festival data (ticket sales, customer details, events) and a parsed result of multiple bands' contract information. We'll join these datasets and build a semantic model using synonyms and defining relationships among the tables. This model will then be used in a prompt to retrieve data from Snowflake for end users via a simple chat-like interface. Consequently, users can obtain answers and data visualizations without needing SQL knowledge, understanding unstructured data queries, or data modeling concepts.

Below is an outline of the lab:

- Ingest data into a database
- Create a simple agent
- Create a semantic layer
- Create a smarter agent using the semantic layer
- Create a search service
- Update the smart agent to also use the search service
- Utilize Snowflake Intelligence

Why it matters:

- **Self-serve data exploration:** Create charts and get instant answers using natural language. With an intuitive interface powered by agentic AI, enable teams to discover trends and analyze data without technical expertise or waiting for custom dashboards.
- **Democratize intelligence:** Access and analyze thousands of data sources simultaneously, going beyond basic AI tools that only handle single documents.
- **Comprehensive integration:** Seamlessly analyze structured and unstructured data together. Connect insights from spreadsheets, documents, images, and databases simultaneously.
- **Automatic security:** Existing Snowflake security controls, including role-based access and data masking, automatically apply to all AI interactions and conversations.

Step 1: Snowflake Intelligence Setup

You will need to enable these features on your demo account. Do this by logging into the deployment (snowflake.okta.com) and choosing the correct deployment for your account, and then run this command. At this time, 5/15/25, this is the case, but this process will change in the future.

SQL

```
alter account <<account locator>>
set UI_ENABLE_AI_ML_FEATURE_19 = 'ENABLED'
    , ENABLE_DATA_TO_ANSWER=true
    , COPILOT_ORCHESTRATOR_PARAM_10='true'
    , COPILOT_ORCHESTRATOR_PARAM_13='true'
    , UI_ENABLE_AI_ML_FEATURE_29 = 'ENABLED'
parameter_comment = 'Enable Snowflake Intelligence - SE Demo';
```

This will allow you to see the needed navigation in Snowsight.

The screenshot shows the Snowflake AI & ML Studio interface. On the left, a sidebar menu lists various categories: Home, Search, Projects, Data, Data Products, AI & ML, and Studio. Under AI & ML, there are sub-options: Studio (which is selected and highlighted in blue), Cortex Search, Features, Models, Evaluations, Document AI, Agents, and Snowflake Intelligence. Two red arrows point from the 'Agents' and 'Snowflake Intelligence' items towards the main content area. The main content area features a large header 'Snowflake AI & ML Studio' and a section titled 'Fast, Easy and Secure'. Below this, there are two cards: 'Cortex Playground' (LLM PREVIEW) and 'Cortex Search' (PREVIEW). The 'Cortex Playground' card includes a description: 'Generate a text response using edge LLMs'. The 'Cortex Search' card includes a description: 'Create and manage search'.

A few things to note:

- If you go into Snowflake Intelligence at this point, it will just **spin forever** because we have not set up the needed parts. This will change in the future, but at this time, this is the case.
- At this time, Snowflake Intelligence (SI) will utilize your **default role**. You need to make sure this is set to the role we will create in the lab (directions below).
- Snowflake Intelligence will also utilize your default warehouse, you must set this up for SI to work.

Profile

⚠ [REDACTED]

Profile photo ? Upload

Username [REDACTED]

First Name [REDACTED]

Last Name [REDACTED]

Email [REDACTED]

🔒 Contact an account admin to change your first name, last name, or email.

Default role & warehouse SNOWFLAKE_INTELLIGENCE_ADMIN... • SNOWFLAKE_INTELLIGENCE_WAREHOUSE...

Language English

Notifications Notify when queries finish in the background

Users in Snowflake Intelligence will map to Snowflake users. Over time, we will move the UI out of Snowflake Snowsight and into a standalone experience (ai.Snowflake.com), but even there, Snowflake roles are planned to be leveraged for configuration. Users of Snowflake Intelligence need a few layers of permissions:

1. They need permission to SELECT on the `config` table that holds the agent's configuration (this will likely disappear in the future, as we'll store configuration automatically).
2. They need permissions on the `temp` schema, which is utilized for different operations like file upload/parsing (this will also likely disappear in the future).
3. They need permission to the underlying Cortex Search Service and the ability to call Cortex Analyst.
4. They need permission for any underlying data surfaced by an agent.

Important notes:

- **Users for Snowflake Intelligence need to have a default role and a default warehouse.**
- **The database needs to be named `Snowflake_intelligence`.**

Metadata Objects

Please create a new worksheet and run this script to create the roles and database objects needed to store Snowflake Intelligence metadata. This will not be needed in the future.

SQL

```
use role accountadmin;

--ability to run across cloud if claude is not in your region:
ALTER ACCOUNT SET CORTEX_ENABLED_CROSS_REGION = 'ANY_REGION';

-- Create roles
create role Snowflake_intelligence_admin_rl;

--need to add ability to create databases
GRANT CREATE DATABASE ON ACCOUNT TO ROLE Snowflake_intelligence_admin_rl;

-- Warehouse that is going to be used for Cortex Search Service creation
-- as well as query execution.
create warehouse Snowflake_intelligence_wh with warehouse_size =
'X-SMALL';
grant usage on warehouse Snowflake_intelligence_wh to role
Snowflake_intelligence_admin_rl;

-- Create a database. This will hold configuration and other objects to
-- support Snowflake Intelligence.
create database Snowflake_intelligence;
grant ownership on database Snowflake_intelligence to role
Snowflake_intelligence_admin_rl;

-- Dynamically grant role 'Snowflake_intelligence_admin_rl' to the
current user
DECLARE
    sql_command STRING;
```

```

BEGIN
    sql_command := 'GRANT ROLE Snowflake_intelligence_admin_rl TO USER
" || CURRENT_USER() || '"';
    EXECUTE IMMEDIATE sql_command;
    RETURN 'Role Snowflake_intelligence_admin_rl granted successfully to
user ' || CURRENT_USER();
END;

-- Set up stages and tables for configuration.
use role Snowflake_intelligence_admin_rl;
use database Snowflake_intelligence;

-- Set up a temp schema for file upload (only temporary stages will be
created here).
create or replace schema Snowflake_intelligence.temp;
grant usage on schema Snowflake_intelligence.temp to role public;

-- OPTIONAL: Set up stages and tables for configuration - you can have
your semantic models be anywhere else, just make sure that the users have
grants to them
create schema if not exists config;
use schema config;

create stage semantic_models encryption = (type = 'SNOWFLAKE_SSE');

```

Agent Configuration

This script is needed at this time to store all of your agent information. Please add this below your above code, and then run. This creates an Agent Policy that allows you to control which roles can see which agents.

```

SQL
use role Snowflake_intelligence_admin_rl;
create schema if not exists Snowflake_intelligence.agents;

-- Make SI agents in general discoverable to everyone.

```

```

grant usage on schema Snowflake_intelligence.agents to role public;

CREATE OR REPLACE ROW ACCESS POLICY
Snowflake_intelligence.agents.agent_policy
AS (grantee_roles ARRAY) RETURNS BOOLEAN ->
    ARRAY_SIZE(FILTER(grantee_roles::ARRAY(VARCHAR), role ->
is_role_in_session(role))) <> 0;

-- Create an agent config table. Multiple tables can be created to give
granular
-- UPDATE/INSERT permissions to different roles.
create or replace table Snowflake_intelligence.agents.config (
    agent_name varchar not null,
    agent_description varchar,
    grantee_roles array not null,
    tools array,
    tool_resources object,
    tool_choice object,
    response_instruction varchar,
    sample_questions array,
    constraint pk_agent_name primary key (agent_name)
)
with row access policy Snowflake_intelligence.agents.agent_policy on
(grantee_roles);
grant select on table Snowflake_intelligence.agents.config to role
public;

```

Currently, Snowflake Intelligence uses your **DEFAULT ROLE**, so we need to change our **default role** to the role we created from the above script. This role is called:

Snowflake_intelligence_admin_rl. This way, all objects will be owned by the correct role, and we will not run into permission issues. Go ahead and change your default role to the newly created role. You can do this by selecting your name at the bottom and going to switch role, and choosing the Snowflake_intelligence_admin_rl, and when you hover over it, choose set as default, and also make sure this is your selected role:

The screenshot shows the Snowflake UI with a red arrow pointing from the 'Switch Role' button in the top-left corner to a dropdown menu. The dropdown menu lists several roles, with 'SNOWFLAKE_INTELLIGENCE_AD' being highlighted by a red box and the text 'Set as default' next to it. Other roles shown include ACCOUNTADMIN, ALTR_SERVICE_ROLE, and ANALYST.

Role	Status
SNOWFLAKE_INTELLIGENCE_AD	Set as default
ACCOUNTADMIN	Default
ALTR_SERVICE_ROLE	
ANALYST	
ALT_AIPOWEREDCONCEPTEXTRACTION_APP	
ALT_AIPOWEREDCONCEPTEXTRACTION_BI	
ALT_AIPOWEREDCONCEPTEXTRACTION_DATA_ENGINEER	
ALT_AIPOWEREDCONCEPTEXTRACTION_DATA_SCIENTIST	

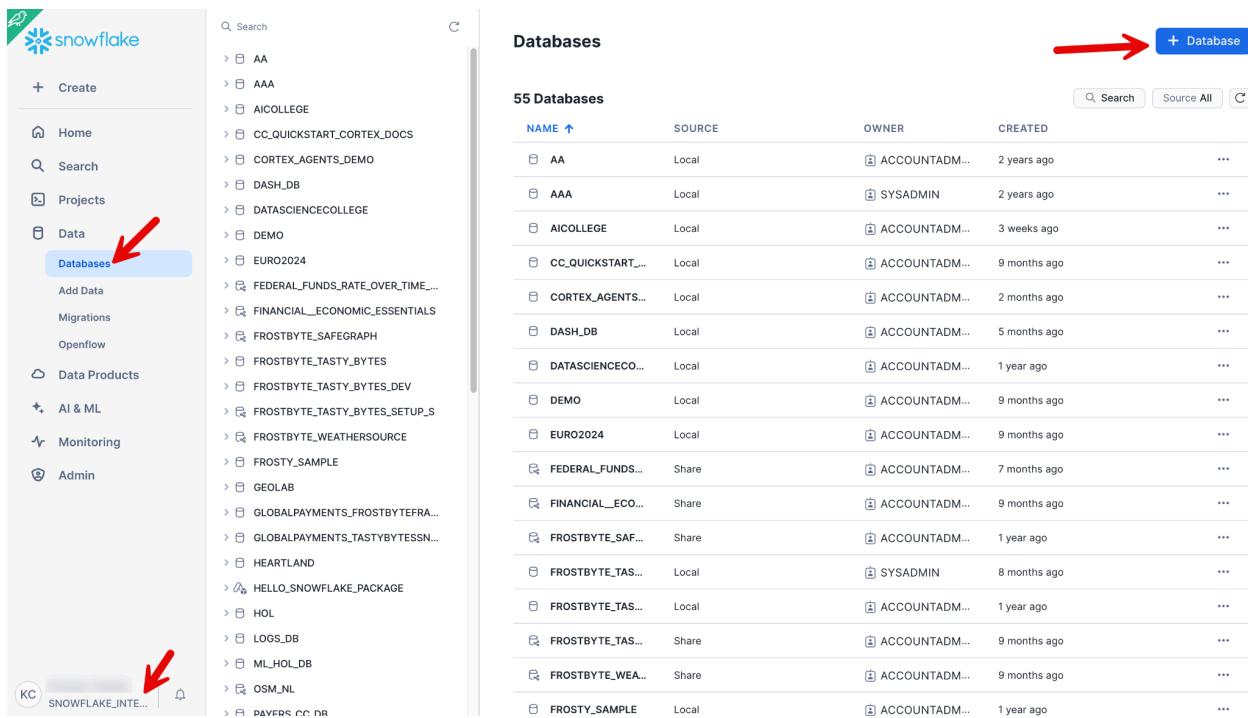
Go ahead and **log out of your account and log back in**, sometimes this resolves some of the permission issues and is a good way to check that your defaults have been set correctly.

Step 2: Load Festival Data

For our lab, we will use some made-up festival data. To support this, we will need to load multiple tables. Please unzip the file below containing all the lab files and then import each as a table into Snowflake through the UI.

[Snowflake Intelligence Festival Data.zip](#)

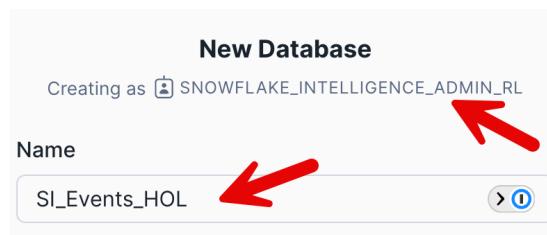
Log into your Snowflake Demo Account and create a space for these tables to live through the UI. Let's create a database we will call it SI_EVENTS_HOL **make sure to create it with the role Snowflake intelligence admin rl:**



Databases

55 Databases

NAME ↑	SOURCE	OWNER	CREATED
AA	Local	ACCOUNTADM...	2 years ago
AAA	Local	SYSADMIN	2 years ago
AICOLLEGE	Local	ACCOUNTADM...	3 weeks ago
CC_QUICKSTART_Cortex_docs	Local	ACCOUNTADM...	9 months ago
Cortex_Agents_demo	Local	ACCOUNTADM...	2 months ago
DASH_DB	Local	ACCOUNTADM...	5 months ago
DATA SCIENCE COLLEGE	Local	ACCOUNTADM...	1 year ago
DEMO	Local	ACCOUNTADM...	9 months ago
EURO2024	Local	ACCOUNTADM...	9 months ago
FEDERAL_FUNDS_RATE_OVER_TIME...	Share	ACCOUNTADM...	7 months ago
FINANCIAL_ECONOMIC_ESSENTIALS	Share	ACCOUNTADM...	9 months ago
FROSTBYTE_SAFEGRAPH	Share	ACCOUNTADM...	1 year ago
FROSTBYTE_TASTY_BYTES	Share	ACCOUNTADM...	8 months ago
FROSTBYTE_TASTY_BYTES_SETUP_S	Share	ACCOUNTADM...	1 year ago
FROSTBYTE_WEATHERSOURCE	Share	ACCOUNTADM...	9 months ago
FROSTY_SAMPLE	Local	ACCOUNTADM...	1 year ago
GEOLAB	Local	ACCOUNTADM...	9 months ago
GLOBALPAYMENTS_FROSTBYTE_TESSN...	Share	ACCOUNTADM...	9 months ago
GLOBALPAYMENTS_TASTY_BYTTESSN...	Share	ACCOUNTADM...	9 months ago
HEARTLAND	Local	ACCOUNTADM...	9 months ago
HELLO_SNOWFLAKE_PACKAGE	Local	ACCOUNTADM...	9 months ago
HOL	Local	ACCOUNTADM...	9 months ago
LOGS_DB	Local	ACCOUNTADM...	9 months ago
ML_HOL_DB	Local	ACCOUNTADM...	9 months ago
OSM_NL	Local	ACCOUNTADM...	9 months ago
PAYERS_CC_DR	Local	ACCOUNTADM...	9 months ago



New Database

Creating as SNOWFLAKE_INTELLIGENCE_ADMIN_RL

Name

Let's create the tables we need for this lab. There are a total of 4 tables needed for this lab, and they can all be created the same way by creating a table from a file in Snowflake. Please repeat the following steps for all four tables:

1. Customers
2. Contracts
3. Ticket Sales
4. Events

We will load these tables into the public schema.

To load a table from a file in Snowsight, go into the SI_Events database we created, go to the Public schema, and go into tables, and then click the create button on the top right to create a table from a file:

The screenshot shows the Snowsight interface for the SI_EVENTS_HOL / PUBLIC schema. On the left, there is a sidebar with various database and schema options. A red arrow points to the 'SI_EVENTS_HOL' schema. Another red arrow points to the 'PUBLIC' schema, which is currently selected and highlighted with a blue background. At the top right, there is a 'Create' button with a dropdown menu. A red arrow points to the 'From File' option in this menu. The dropdown also includes 'Standard', 'As Select', and 'External' options.

Name each table the same name as the file you unzipped:

The screenshot shows the 'Load Data into Table' dialog. At the top, it says 'Load Data into Table' and 'SI_EVENTS_HOL.PUBLIC.TICKET_SALES'. Below this, there is a file selection area labeled 'ticket_sales.csv - 1.6MB' with a 'Browse' button. Underneath, it says 'Select or create a database and schema' with a 'Schema SI_EVENTS_HOL.PUBLIC' dropdown and a '+ Database' button. In the main area, it says 'Select or create a table' with a '+ Create new table' button. A red box highlights the 'Name' input field, which contains 'ticket_sales'. At the bottom, there are 'Cancel', 'Back', and 'Next' buttons.

Open the View Options section in the file format area and make sure that the header is the first line of the document:

Load Data into Table

contracts.csv → SI_EVENTS_HOL.PUBLIC.CONTRACTS

File format

Delimited Files (CSV or TSV)

Select existing or create in [Worksheets](#)

Learn more about format-specific configurations in [Snowflake Docs](#)

View options ↴

Edit Schema

3 Columns

DATA TYPE	COLUMN NAME
VARCHAR	DOCUMENT_TITI

Load Data into Table

contracts.csv → SI_EVENTS_HOL.PUBLIC.CONTRACTS

File format

Delimited Files (CSV or TSV)

Select existing or create in [Worksheets](#)

Learn more about format-specific configurations in [Snowflake Docs](#)

Header First line contains he... ↴

Field delimiter No header First line contains he... ↴

Trim space Skip first line

Field optionally enclosed by Custom

Replace invalid characters True (default)

Edit Schema

3 Columns

DATA TYPE	COLUMN NAME	COLUMN DATA
VARCHAR	DOCUMENT_TITI	Artist_Agreement_The_Algorhythms, Ar...
VARCHAR	URL	https://music-fest.sharepoint.com/doc...
VARCHAR	TEXT	The Algorhythms will perform on the M...

Successfully Loaded Data

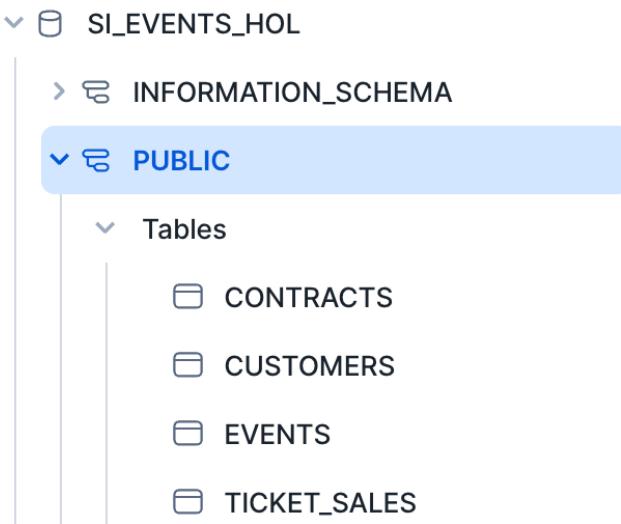
contracts.csv → SI_EVENTS_HOL.PUBLIC.CONTRACTS

10 rows were successfully inserted into the table.

Query Data

Done

Repeat this for all 4 files so you have all 4 tables needed for the lab. When you are complete, you should see the following under tables in the public schema:



Configure DORA for Grading

Why Use DORA?

- **Automated Grading:** DORA enables SE BPs to automatically grade hands-on curriculum tasks, leveraging the grading framework created by the Training team. Developer Relations also uses it for event labs.
- **Functionality:** DORA allows SE BPs to write tests that validate the completion of hands-on steps and track progress toward certification.

DORA API Integration Setup

1. To submit grading requests via external functions, follow the steps below in a new **SQL worksheet** in your Snowflake account.

SQL

```
USE ROLE ACCOUNTADMIN;

-- Create API integration
CREATE OR REPLACE API INTEGRATION dora_api_integration
API_PROVIDER = AWS_API_GATEWAY
API_AWS_ROLE_ARN =
'arn:aws:iam::321463406630:role/snowflakeLearnerAssumedRole'
```

```

ENABLED = TRUE
API_ALLOWED_PREFIXES = (
    'https://awy6hshxy4.execute-api.us-west-2.amazonaws.com/dev/edu_dora'
);

-- Confirm integration
SHOW INTEGRATIONS;

-- Create utility database
CREATE OR REPLACE DATABASE util_db;

-- Create greeting function
CREATE OR REPLACE EXTERNAL FUNCTION util_db.public.se_greeting(
    email VARCHAR,
    firstname VARCHAR,
    middlename VARCHAR,
    lastname VARCHAR
)
RETURNS VARIANT
API_INTEGRATION = dora_api_integration
CONTEXT_HEADERS = (
    CURRENT_TIMESTAMP,
    CURRENT_ACCOUNT,
    CURRENT_STATEMENT,
    CURRENT_ACCOUNT_NAME
)
AS
'https://awy6hshxy4.execute-api.us-west-2.amazonaws.com/dev/edu_dora/greeting';

-- Replace with your Snowflake details
-- Example:
-- SELECT util_db.public.se_greeting(
--     'dan.murphy@snowflake.com', 'Dan', '', 'Murphy');

SELECT util_db.public.se_greeting(
    'your_email@Snowflake.com',
    'Your First Name',
    'Your Middle Name or empty string',
    'Your Last Name'
);

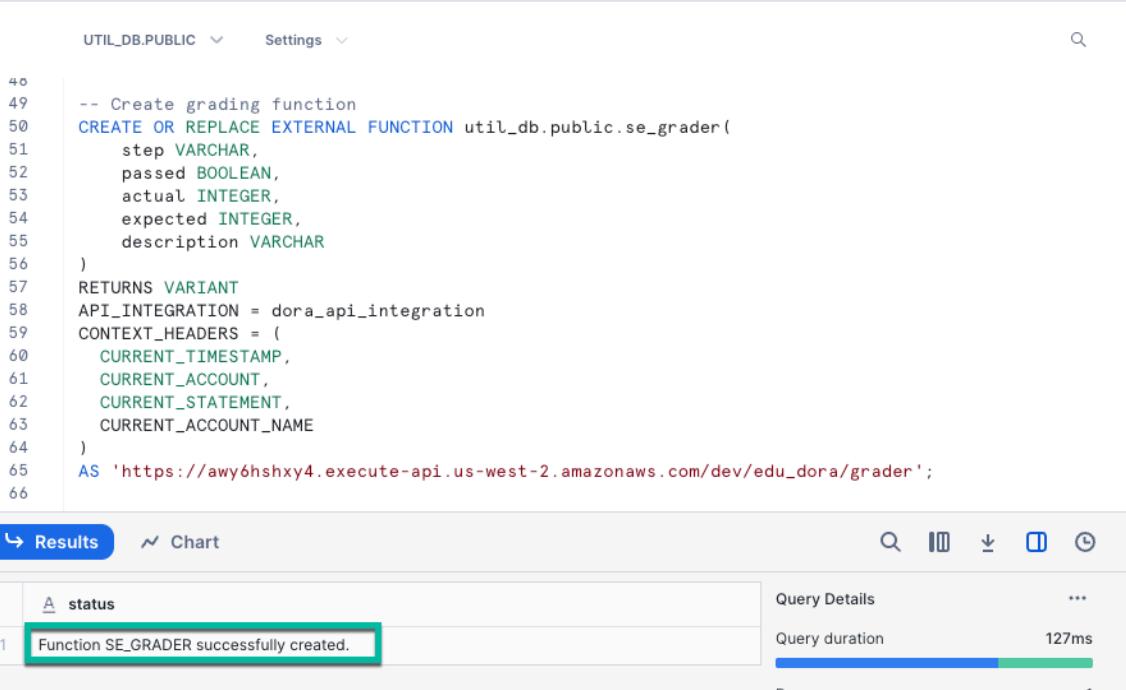
-- Create grading function
CREATE OR REPLACE EXTERNAL FUNCTION util_db.public.se_grader(
    step VARCHAR,

```

```
    passed BOOLEAN,
    actual INTEGER,
    expected INTEGER,
    description VARCHAR
)
RETURNS VARIANT
API_INTEGRATION = dora_api_integration
CONTEXT_HEADERS = (
    CURRENT_TIMESTAMP,
    CURRENT_ACCOUNT,
    CURRENT_STATEMENT,
    CURRENT_ACCOUNT_NAME
)
AS
'https://awy6hshxy4.execute-api.us-west-2.amazonaws.com/dev/edu\_dora/grader';

grant usage on database util_db to role public;
grant usage on schema util_db.public to role public;
grant usage on function
util_db.public.se_grader(varchar,boolean,integer,integer,varchar) to role
public;
```

 **Expected Output:** You should see confirmation messages after each `CREATE` statement and be able to run the `se_greeting` function successfully.



The screenshot shows a Snowflake SQL interface. The top navigation bar includes "UTIL_DB.PUBLIC" and "Settings". The main area contains the following SQL code:

```
40
41 -- Create grading function
42 CREATE OR REPLACE EXTERNAL FUNCTION util_db.public.se_grader(
43     step VARCHAR,
44     passed BOOLEAN,
45     actual INTEGER,
46     expected INTEGER,
47     description VARCHAR
48 )
49 RETURNS VARIANT
50 API_INTEGRATION = dora_api_integration
51 CONTEXT_HEADERS = (
52     CURRENT_TIMESTAMP,
53     CURRENT_ACCOUNT,
54     CURRENT_STATEMENT,
55     CURRENT_ACCOUNT_NAME
56 )
57 AS 'https://awy6hshxy4.execute-api.us-west-2.amazonaws.com/dev/edu_dora/grader';
58
```

Below the code, there are two tabs: "Results" (selected) and "Chart". The results table has one row with the following data:

status	Query Details	...
Function SE_GRADER successfully created.	Query duration 127ms	

Step 3: Create a Simple Agent

What is an agent? Agentic AI is the buzzword of 2025, but with Snowflake agents, these are tangible things we're creating that will include one or more Cortex building blocks, like an analyst semantic model or search service

Go to Snowflake, and the Agents section should now work with the config tables we created above. The page will keep attempting to redraw at this point, but that is expected for now. Go ahead and click on Create Agent.

The screenshot shows the Snowflake AI & ML interface. On the left is a sidebar with various navigation options like Home, Search, Projects, Data, Data Products, AI & ML (with Studio, Cortex Search, Features, Models, Evaluations, Document AI), and Agents. The Agents option is currently selected, highlighted with a blue bar. The main content area is titled 'Agents' and shows '1 Agent'. It has tabs for 'AGENT NAME' and 'AGENT OVERVIEW'. In the top right corner, there are links for 'Documentation' and a prominent blue button labeled '+ Create agent'. A red arrow points from the bottom right towards this '+ Create agent' button.

Let's create a very simple agent

- Let's name it Claude
- Give it a description 'This agent is connected securely to Claude in Snowflake'
- Create a sample question 'What are some time series functions I can use in Snowflake'

New Agent

→|

Name

Claude



Agent Overview

This agent is connected securely to Claude in Snowflake.

Response instruction

Response instruction provides instructions to the model for response generation. Try to clearly define its role, objectives, and key functions.

Knowledge



Snowflake

+ Semantic model

+ Search service

Users

Add role to share with

Sample questions

What are some time series functions I can use in Snowflake



x

+ Add question

Questions will be displayed as onboarding questions in Snowflake Intelligence.

Cancel

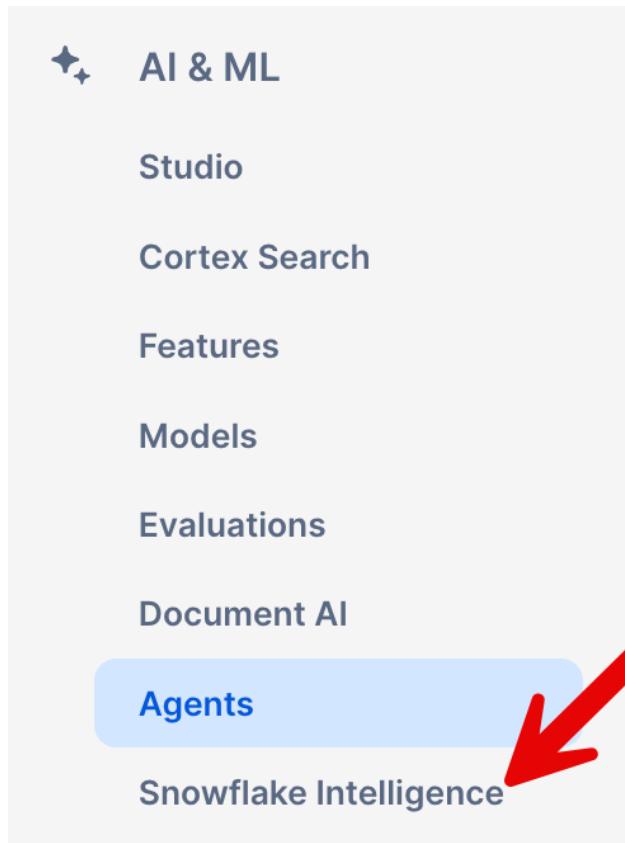
Create

Refresh the screen, and we will see the new agent created:

The screenshot shows the 'Agents' section of the Snowflake interface. At the top right are 'Documentation' and '+ Create agent' buttons. Below is a search bar with a magnifying glass icon and a clear button. The main area shows '2 Agents'. One agent, 'Claude', is listed with an 'AGENT NAME' field and an 'AGENT OVERVIEW' section containing the text: 'This agent is connected securely to Claude in Snowflake.'

Step 4: Go to Snowflake Intelligence

To easily navigate while using the Snowflake Intelligence application, open a new browser tab specifically for Snowsight. This will allow you to switch between the standard Snowsight interface and the Snowflake Intelligence application, which you can access via the Snowflake Intelligence link. After clicking the link, the navigation will change to focus on Snowflake Intelligence.



Now we can see our agent that we created in the dropdown and be able to interact with Snowflake intelligence easily:

Good afternoon,

How can I help you?

How may I assist you?

Claude   Upload files



I can also upload files as needed to have Claude look at the file and complete my analysis.

Go ahead and query Snowflake Intelligence. Let's ask this: [*What can you tell me about the company Snowflake?*](#)

What can you tell me about the company Snowflake?



Snowflake is a cloud-based data warehousing company founded in 2012 by Benoit Dageville, Thierry Cruanes, and Marcin Żukowski. Here are key points about Snowflake:

1. Core Business:

- Provides cloud-based data storage and analytics services
- Allows companies to store and analyze data using cloud-based hardware and software
- Offers data warehouse-as-a-service running on major cloud platforms (AWS, Microsoft Azure, Google Cloud)

2. Key Features:

- Separation of storage and compute resources
- Ability to share data securely across organizations
- Automatic scaling and performance optimization
- Support for structured and semi-structured data

We have currently developed a basic agent lacking access to data within your Snowflake account. It solely interacts with Claude. To enhance its intelligence, we need to integrate a semantic model and connect it to the agent, thereby providing improved context.

Step 5: Create a Semantic Model

Let's now build a more intelligent agent that analyzes the data within Snowflake by utilizing our semantic layer.

First thing we need to do is enable Directory Table for our Semantic_Models stage that we created earlier

The screenshot shows the Snowflake UI with the path: SNOWFLAKE_INTELLIGENCE / CONFIG / SEMANTIC_MODELS. On the left, there is a file tree with various stages like QM_DATA_SHARE, QUICKSTART, RETAIL_DEMO, SECURITY_NETWORK_DB, SI_EVENTS_HOL, SKICAR, SKO, SKO_SKORAGHOP_PROD, SNOWFLAKE, and SNOWFLAKE_INTELLIGENCE. Under SNOWFLAKE_INTELLIGENCE, there are AGENTS, CONFIG, INFORMATION_SCHEMA, PUBLIC, TEMP, TEMPORARY, SNOWFLAKE_SAMPLE_DATA, SNOWHOUSE, SNOWPARK_WORKSHOP, and SNOWPIPE_STREAMING. The 'SEMANTIC_MODELS' stage is highlighted with a blue box and a red arrow pointing to it. On the right, there is a 'View Files with Directory Tables' section with a magnifying glass icon. Below it is a button labeled 'Enable Directory Table' with a red arrow pointing to it. At the bottom, there is a link 'Learn more'.

Then we can create a semantic model that we can use in an agent. Let's do that through Snowflake AI & ML Studio using Cortex Analyst:

The screenshot shows the Snowflake AI & ML Studio interface. The sidebar on the left has options like Home, Search, Projects, Data, Data Products, AI & ML, and Studio. The 'AI & ML' section is expanded, and 'Studio' is highlighted with a blue box and a red arrow. The main area is titled 'Snowflake AI & ML Studio' and features a section 'Fast, Easy and Secure AI and ML with Snowflake'. It includes four cards: 'Cortex Playground' (Try), 'Create Custom LLM' (Fine-tune), 'Cortex Search' (Try), and 'Cortex Analyst' (Try). The 'Cortex Analyst' card has a red arrow pointing to its 'Try' button.

Go ahead and select create new:

Select a semantic model

Cortex Analyst leverages the concept of a semantic model to bridge the gap between business user vocabulary and raw database schemas, providing highly accurate results.

Role and warehouse

 SNOWFLAKE_INTELLIGENCE_ADMIN... • SNOWFLAKE_INTELLIGENCE_WH (...)

Make sure to select a role that has access to the stage to store your semantic model and underlying data. We need a warehouse to fetch data to create a semantic model and execute SQL generated by Cortex Analyst.

Select from

Stages

Semantic information stored in YAML files in a stage

Semantic Views

Semantic information stored natively in a Semantic View

SNOWFLAKE_INTELLIGENCE.CONFIG ▾

SEMANTIC_MODELS ▾



No valid files available



+ Create new

Browse

Open

We can choose the SNOWFLAKE_INTELLIGENCE database and CONFIG schema we created earlier, as well as the stage of SEMANTIC_MODELS:

Name: Music Festival

Description: This has information on music festivals, including locations, events, and ticket sales

Create Semantic Model

Snowflake Intelligence Admin • SNOWFLAKE_INTELLIGENCE_WH (...)

Getting started SNOWFLAKE_INTELLIGENCE.CONFIG

Select tables SEMANTIC_MODELS

Select columns

Getting started

As you start creating this semantic view, make sure you're focused on a well-defined and scoped use case.

Location to store

SNOWFLAKE_INTELLIGENCE.CONFIG SEMANTIC_MODELS

Name

music_festival

Description (optional)

This has information on music festivals, including locations, events, and ticket sales.

File name

music_festival.yaml ✓

[Cancel](#) [Next: Select tables](#)

Let's select our tables from the SI_EVENTS_HOL data that we imported at the beginning of the lab:

Create Semantic Model

SNOWFLAKE_INTELLIGENCE_ADMIN... SNOWFLAKE_INTELLIGENCE_WH (...)

Getting started

Select tables

Select columns

Select tables

All Selected 4

Search

SI_EVENTS_HOL

INFORMATION_SCHEMA

PUBLIC

4 Tables

CONTRACTS

CUSTOMERS

EVENTS

TICKET_SALES

Then we can select the columns to choose. Let's just select all columns by choosing the top checkbox:

Create Semantic Model

SNOWFLAKE_INTELLIGENCE_ADMIN... SNOWFLAKE_INTELLIGENCE_WH (...)

Getting started

Select tables

Select columns

Select columns

Quality of Cortex Analyst maybe affected by the selected columns. Try select columns that are commonly being used for your task.

Columns (20 selected) Expand all

Search columns

SI_EVENTS_HOL.PUBLIC.CONTRACTS

SI_EVENTS_HOL.PUBLIC.CUSTOMERS

SI_EVENTS_HOL.PUBLIC.EVENTS

SI_EVENTS_HOL.PUBLIC.TICKET_SALES

Include example data from selected columns to improve its quality.
Sample values are considered metadata. [Learn more](#)

View the Results

When we click ‘Create and Save’ it will generate a starter semantic model for us:

The screenshot shows the Cortex Studio interface with the 'Cortex Analyst' tab selected. The main navigation bar includes 'Semantic Model' (which is currently selected), 'Monitoring', and a 'PREVIEW' button. A connection to 'SNOWFLAKE_INTELLIGENCE_ADM...' is shown. The left sidebar lists 'SEMANTIC_MODELS' and 'music_festival.yaml'. On the right, there's a 'Edit YAML' button. The main content area displays the 'music_festival' semantic model. It includes a 'Custom Instructions' section with an 'Edit' button. Below that is a 'Logical tables' section with 4 entries, each with an 'Edit' button and a close 'X' icon. The first entry is 'What is a logical table?' with a detailed description. The second entry is '4 of 4 logical tables missing synonyms' with a note about improving response quality. Underneath is a 'CONTRACTS' section with an 'Edit' button and a 'SI_EVENTS_HOL.PUBLIC.CONTRACTS' entry. There are also 'Dimensions' and 'Time Dimensions' sections with counts of 3 and 0 respectively, each with an 'Edit' button and a plus '+' icon.

We can see that it has already created a description as well as some other details, like synonyms. Let's go ahead and add ‘show’ and ‘concert’ to the list of synonyms for EVENT_NAME in the EVENTS table:

Expression ⓘ

+ Add column

EVENT_NAME

Dimension name

EVENT_NAME

Generate Fields

Data type

VARCHAR(16777216)

Dimension description (optional)

The name of the event that occurred.

Synonyms (optional)

le, event_description, event_label, event_heading, event_caption, event_header, show, concert

Connect Cortex Search (optional) ⓘ

+ Search Service

Sample values

+ Value

Instead Fest

×

Test the Model

I can test this model right now by going to the side window and putting in the following prompt:
What are the different events in Europe

Then it will go against my model and then write SQL, and then execute that SQL against my model:

The screenshot shows the Cortex Studio interface. On the left, a 'Model' configuration for 'music_festival.yaml' is displayed. It includes fields for 'Expression' (EVENT_NAME), 'Dimension name' (EVENT_NAME), 'Data type' (VARCHAR(16777216)), 'Dimension description (optional)' (The name of the event that occurred.), 'Synonyms (optional)' (event_title, event_description, event_label, event_heading, event_caption, event_header, show,), 'Connect Cortex Search (optional)', and 'Sample values' (Instead Fest, Ok Fest, Difficult Fest). On the right, a semantic search interface shows a query 'User What are the different events in Europe'. The results are a table of events in Europe from January to June 2025:

EVENT_NAME	START_DATE	END_DATE
Instead Fest	2025-06-13	2025-06-
Ok Fest	2025-03-12	2025-03-
Difficult Fest	2025-02-05	2025-02-
Wall Fest	2025-01-23	2025-01-
Oil Fest	2025-01-09	2025-01-

Below the table, a query editor shows a physical query (executed) and a semantic query:

```

Physical query (executed) - Semantic query
SELECT
    event_name,
    MIN(event_date) AS start_date,
    MAX(event_date) AS end_date
  
```

Enter prompt

Move Columns

The model incorrectly identified EVENT_ID as a fact due to its numerical data type, but it is actually a dimensional column. This can be easily corrected by moving it to the dimensions section in the edit menu. It is crucial to review and adjust the model after Snowflake's initial column categorization.

TICKET_SALES

SI_EVENTS_HOL.PUBLIC.TICKET_SALES

Dimensions 1 +

A TICKET_TYPE

The type of ticket purchased, which can be one of three options: General, VIP, or Early Bird, indicating the level of access or perks associated with the ticket.

TICKET_TYPE · VARCHAR(16777216)

Time Dimensions 1 +

Facts 4 +

TICKET_ID

Unique identifier for each ticket sale transaction.

TICKET_ID · NUMBER(38,0)

EVENT_ID

Unique identifier for the event associated with the ticket sale.

EVENT_ID · NUMBER(38,0)

CUSTOMER_ID

Unique identifier for the customer who purchased the ticket.

CUSTOMER_ID · NUMBER(38,0)

We will need to do this for the following incorrectly identified columns:

1. TICKET_SALES
 - a. TICKET_ID - need to move from fact to dimension
 - b. EVENT_ID - need to move from fact to dimension
 - c. CUSTOMER_ID - need to move from fact to dimension
2. CUSTOMERS
 - a. CUSTOMER_ID - need to move from fact to dimension

Define Table Relationships

We then need to define our relationships with the other tables. By default, no relationships are created, but they are needed for more complex queries that span multiple tables.

Our first relationship we will define is TICKET_SALES to EVENT. For every EVENT, there are many ticket sales, and the joining column is EVENT_ID. Let's define a relationship that represents that:

Relationship Name: TicketSales_to_Events
Join Type: Inner
Relationship type: many_to_one
Left Table: Ticket_Sales
Right Table: Events
Relationship Columns: EVENT_ID and EVENT_ID

Relationships 0 +

Add relationship

Relationship name

Join type Relationship type

Left table Right table

Relationship columns
+ Columns

Left column Right column

Cancel Add

Let's add a second relationship of customers to tickets:

Relationship Name: TicketSales_to_Customers
Join Type: Inner
Relationship type: many_to_one
Left Table: Customers
Right Table: Ticket_Sales
Relationship Columns: CUSTOMER_ID and CUSTOMER_ID

Relationship name

Join type

Relationship type

Left table

Right table

Relationship columns

Left column

Right column

+ Columns

Cancel

Save

- Relationships** 2 ← +
-
- TicketSales_to_Events** Edit
- TICKET_SALES . EVENT_ID — EVENTS . EVENT_ID
- Join: inner · Relationship: many_to_one
-
- Customer_to_TicketSales** Edit
- CUSTOMERS . CUSTOMER_ID — TICKET_SALES . CUSTOMER_ID
- Join: inner · Relationship: many_to_one

Now we have two relationships set up in our semantic model. Let's go ahead and test them using the prompt on the right side. We can ask this: *How many ticket sales were there for Difficult Fest?*

→ |  music_festival | ↵

User

How many ticket sales were there for Difficult Fest?

Cortex Analyst

Request ID  

This is our interpretation of your question:

How many ticket sales were there for Difficult Fest over the entire available time period?

START_DATE	END_DATE	TICKET_SALE
2025-01-01	2025-02-05	758



 Physical query (executed) · Semantic query  

```
SELECT
    MIN(ts.sale_date) AS start_date,
    MAX(ts.sale_date) AS end_date,
    COUNT(*) AS ticket_sales
FROM
    ticket_sales AS ts
    INNER JOIN events AS e ON ts.event_id =
WHERE
    e.event_name = 'Difficult Fest'
    /* Generated by Cortex Analyst */
```

Enter prompt

▶ Run

We can see that it was able to utilize our join and see how many tickets were sold for that event by joining the two tables together we defined in our relationship.

We can also ask it to ‘Count the total customers by customer region that went to Difficult Fest’, and this should span all of our tables in our semantic model and give us values:

→ music_festival

User

Count the total customers by customer region that went to Difficult Fest

Cortex Analyst

Request ID



This is our interpretation of your question:

What is the count of unique customers by customer region who attended Difficult Fest over the entire available time period?

REGION	CUSTOMER_COUNT	EVENT_START
South America	161	2025-02-0
Europe	161	2025-02-0
Africa	155	2025-02-0
North America	144	2025-02-0
Asia	137	2025-02-0

Physical query (executed) · Semantic query

```
WITH event_customers AS (
  SELECT
    DISTINCT c.customer_id,
    c.region
  FROM
    customers AS c
    INNER JOIN ticket_sales AS ts ON c.cust
    INNER JOIN events AS e ON ts.event_id =
  WHERE
    e.event_name = 'Difficult Fest'
```

Excellent! This functionality is operating as anticipated.

Verify the Queries

Since these queries returned the correct values, we can add them as verified queries, which will fuel our model's improvement.

→  music_festival 

```
WITH unique_customers AS (
    SELECT
        DISTINCT c.region,
        c.customer_id
    FROM
        customers AS c
    INNER JOIN ticket_sales AS ts ON c.cus
    INNER JOIN events AS e ON ts.event_id
    WHERE
        e.event_name = 'Difficult Fest'
)
SELECT
    uc.region,
    COUNT(uc.customer_id) AS customer_count,
    MIN(ts.sale_date) AS start_date,
    MAX(ts.sale_date) AS end_date
FROM
    unique_customers AS uc
    LEFT JOIN ticket_sales AS ts ON uc.custo
GROUP BY
    uc.region
ORDER BY
    customer_count DESC NULLS LAST
/* Generated by Cortex Analyst */
```



+ Verified query

Add verified query

Verified query name

Events in Europe

Question

What are the different events in Europe

Add this question to [onboarding questions](#)

Query

Semantic query

▶ Test

```
1  SELECT
2      MIN(ts.sale_date) AS start_date,
3      MAX(ts.sale_date) AS end_date,
4      COUNT(*) AS ticket_sales
5  FROM
6      ticket_sales AS ts
7      INNER JOIN events AS e ON ts.event_id = e.event_id
8  WHERE
9      e.event_name = 'Difficult Fest'
```

Only Semantic queries can be saved as verified queries.

[Learn more](#)

Cancel

Add

These will now show up under verified queries in our model definition:

Verified Queries 2

+

- ✓ Count the total customers by customer region that went to Difficult Fest

Edit



Difficult Fest Customers · By Kirsten Childs · May 29, 2025

```
WITH event_customers AS (
  SELECT
    DISTINCT c.customer_id,
    c.region
  FROM
    customers AS c
  INNER JOIN ticket_sales AS ts ON c.customer_id = ts.customer_id
  INNER JOIN events AS e ON ts.event_id = e.event_id
  WHERE
    e.event_name = 'Difficult Fest'
```

- ✓ How many ticket sales were there for Difficult Fest?

Edit



Events in Europe · By Kirsten Childs · May 29, 2025

```
SELECT
  MIN(ts.sale_date) AS start_date,
  MAX(ts.sale_date) AS end_date,
  COUNT(*) AS num_ticket_sales
FROM
  ticket_sales AS ts
  INNER JOIN events AS e ON ts.event_id = e.event_id
WHERE
  e.event_name = 'Difficult Fest'
```

Save the Model

When you are done, make sure you save the model:

The screenshot shows the Snowflake Semantic Model editor interface. At the top, there are tabs for 'SNOWFLAKE_INTELLIGENCE_ADMIN...' and 'SNOWFLAKE_INTELLIGENCE_WH (...)', along with 'Open', 'Save', and '...' buttons. Below the tabs, there's a large text area containing a SQL query:

```
SELECT
    DISTINCT c.region,
    c.customer_id
FROM
    customers AS c
INNER JOIN ticket_sales AS ts ON c.customer_id = ts.customer_id
INNER JOIN events AS e ON ts.event_id = e.event_id
WHERE
    e.event_name = 'Difficult Fest'
)
SELECT
    uc.region,
    COUNT(uc.customer_id) AS customer_count,
    MIN(ts.sale_date) AS start_date,
```

Step 6: Create an Agent to Use the Semantic Model

Let's go back to the agents now and create a new Agent

The screenshot shows the Snowflake Agents page. On the left, there's a sidebar with 'AI & ML' selected, which has 'Agents' highlighted. The main area shows '2 Agents' with one named 'Claude'. A red arrow points to the '+ Create agent' button at the top right. Another red arrow points to the 'Agents' link in the sidebar.

We will use the following information:

Name: Music Festival Agent

Description: An agent that can return information about music festival events and ticket sales

After those are filled in, go ahead and click '+ Semantic Model', and attach the semantic model we created earlier.

New Agent



Name

Music Festival Agent



Agent Overview

An agent that can return information about music festival events and ticket sales



Response instruction

Response instruction provides instructions to the model for response generation. Try to clearly define its role, objectives, and key functions.

Knowledge

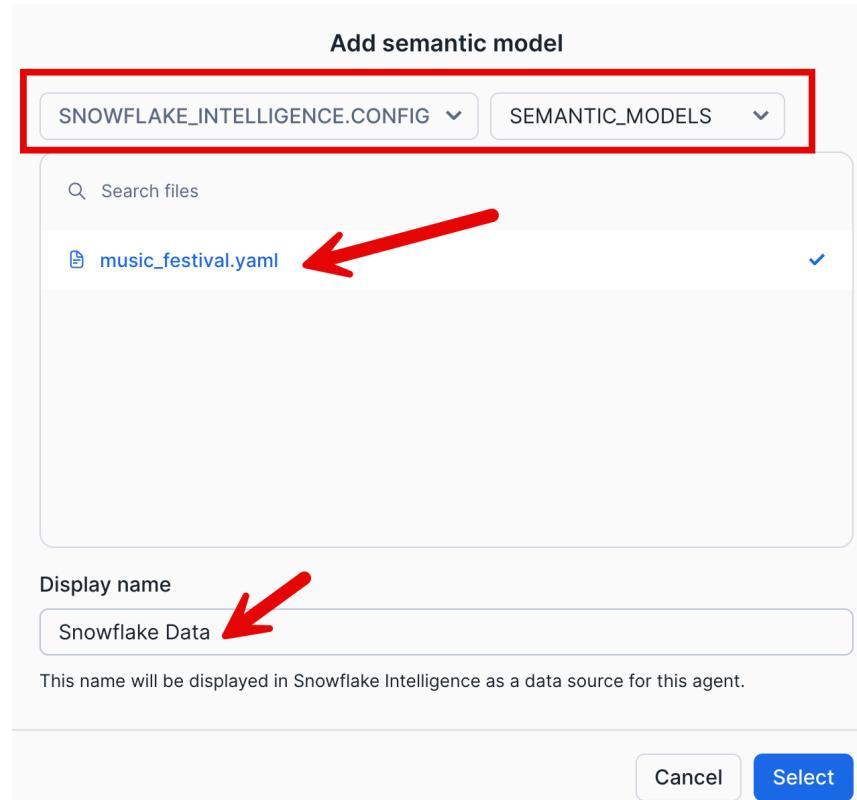
 Snowflake



+ Semantic model

+ Search service

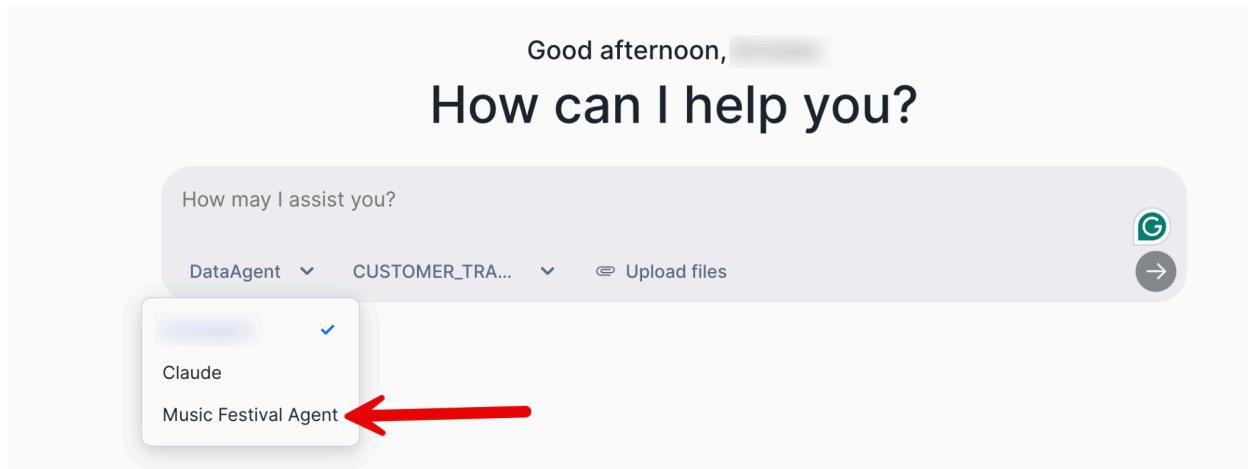
We will choose where we made that model, which was in SNOWFLAKE_INTELLIGENCE.CONFIG and in the SEMANTIC_MODELS stage, and give it the display name Snowflake Data.



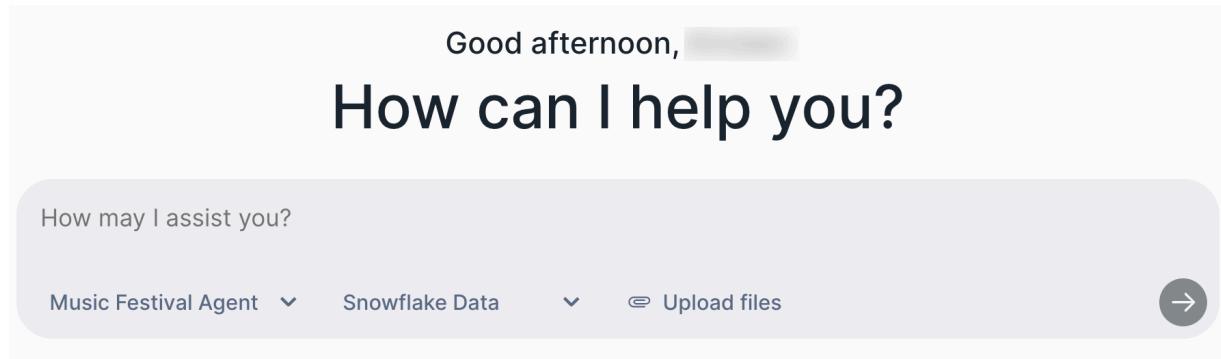
Step 7: Test with Smart Agent

Let's go back to Snowflake Intelligence to see our new agent, which leverages our semantic model. Either click Snowflake Intelligence or go to your Snowflake Intelligence tab and refresh the page.

We can now see the Music Festival Agent.



We can also see that it shows our Snowflake Data, which is what we called our music festival.yml semantic file we created earlier:



Let's ask it a question: [Show me the top 5 events by tickets sold.](#)

The screenshot shows the AI interface again. The user has entered the query "Show me the top 5 events by tickets sold.". Below the input field, a message indicates "Thinking completed" with a green checkmark and "1 verified query". There is also a "Show Details" button with a dropdown arrow.

Based on the SQL results from the tool execution, here are the top 5 events by tickets sold:

1. Instead Fest: 4,514 tickets
2. Age Fest: 3,608 tickets
3. Least Fest: 3,287 tickets
4. Own Fest: 3,285 tickets
5. Nice Fest: 3,048 tickets

EVENT_NAME ↑	TOTAL_TICKETS SOLD ↑	FIRST_SALE_DATE ↑	LAST_SALE_DATE ↑
Instead Fest	4514	20089	20252
Age Fest	3608	20089	20269
Least Fest	3287	20089	20258
Own Fest	3285	20089	20251

The data shows ticket sales spanning from January 1, 2025 to various dates in June 2025, with Instead Fest being the clear leader in ticket sales.

Fantastic, it gave us an answer with a chart of the top 5 events! I can open the SQL section and see the join in the SQL.

Show me the top 5 events by tickets sold.

Thinking steps

Hide Details ^



✓ Planning the next steps

I'll help you find the top 5 events based on ticket sales. Let me query the music festival data using the cortex_analyst_text_to_sql tool.

✓ Executing SQL ✨

```
WITH __events AS (
  SELECT
    event_name,
    event_id
  FROM si_events_hol.public.events
), __ticket_sales AS (
  SELECT
    event_id,
    ticket_id,
    sale_date
  FROM si_events_hol.public.ticket_sales
)
SELECT
  e.event_name,
  COUNT(ts.ticket_id) AS total_tickets_sold,
```

✓ Planning the next steps

I have gathered sufficient data to answer the user's question. The SQL results from the previous tool call show a clear table with the top 5 events ranked by total tickets sold, including the event names and their corresponding ticket sales numbers. Since the user didn't specifically request a chart or any additional information, this data is complete and sufficient to answer their question.

We can ask another question: *What is the ratio of ticket types purchased for Instead Fest?*

What is the ratio of ticket types purchased for Instead Fest?

✓ Thinking completed ✅ 1 verified query

Show Details ▾

Based on the tool results from our previous conversation, here's the breakdown of ticket types for Instead Fest:

1. General Admission: 1,565 tickets (34.67% of total)
2. Early Bird: 1,492 tickets (33.05% of total)
3. VIP: 1,457 tickets (32.28% of total)

The distribution is remarkably even across all three ticket types, with each type representing roughly one-third of total ticket sales. General Admission has a slight lead with just over 34% of sales, followed by Early Bird tickets at 33%, and VIP tickets making up the remaining 32%.

TICKET_TYPE ↑	TYPE_COUNT ↑	PERCENTAGE ↑
General	1565	0.3466991582
Early Bird	1492	0.3305272486
VIP	1457	0.3227735933

A third question I could ask is: *Did any one customer go to multiple shows?* This would utilize a synonym we created (show) as well as utilize the relationship we created to the customers table:

Did any one customer go to multiple shows?

✓ Thinking completed ✅ 1 verified query

Show Details ▾

Yes, many customers attended multiple shows. Based on the tool results, there were quite a few customers who went to multiple events, with some particularly active attendees:

The most frequent attendees were:

- Michelle Pena and Eric Clayton: attended 12 different shows each
- Angela Douglas, Maria Foster, Thomas Valdez, Joseph Wheeler, and Ashley Ortiz: attended 11 shows each
- Multiple customers attended 10 shows, including Charlene Harper, Tina Walker, Kyle Smith, and others
- Several customers attended 9 shows

The data shows a significant number of repeat customers, with many people attending multiple festivals throughout the season. The distribution ranges from customers attending 9 shows up to 12 shows, demonstrating strong customer loyalty across multiple events.

CUSTOMER_NAME ↑	NUM_EVENTS ↑
Michelle Pena	12
Eric Clayton	12
Angela Douglas	11
Maria Foster	11

These answers appear to be correct. If they were incorrect, the first step would be to review the semantic model. This review would focus on verifying the accuracy of the defined relationships and identifying any potential synonyms that could enhance the model's intelligence and robustness.

Step 8: Create Search Service

We have document data that has been parsed and loaded into a table named 'Contracts' from a 'Contracts.csv' file. To enable searching of this text data, we will create a search service. This can be done by navigating to AI & ML, then Cortex Search, and following the wizard.

Snowflake AI & ML Studio

Fast, Easy and Secure AI and ML with Snowflake

Snowflake offers two broad categories of powerful, intelligent features based on AI and Machine Learning (ML). These features can help you do more with your data in the cloud. Get started quickly with AI & ML in this Studio. [Learn more](#)

Cortex Playground PREVIEW

LLM
Generate a text response to your prompt using cutting-edge LLMs

Cortex Search PREVIEW

LLM
Create and manage search services over your data

Cortex Search PREVIEW

Why use Cortex Search?

Build performant, high-quality search bars and RAG chatbots on unstructured data.

Tutorials

Cortex Search Documentation

Learn more about creation, usage, and management of Cortex Search Services.

Documentation

NAME	DATABASE	SCHEMA	INDEXING ST...	SERVING STA...	ROWS	TARGET LAG	CREATED ...
TRANSCRIPTS_SEARCH_SERVICE	AICOLLEGE	PUBLIC	ACTIVE	ACTIVE	35	365 days	May 20, 2025
RESUME_SEARCH_SERVICE	DATASCIENTIST	PUBLIC	ACTIVE	ACTIVE	41	365 days	May 9, 2025
RECOGNITION_MSG_SERVICE	AA	PUBLIC	ACTIVE	ACTIVE	300	365 days	Apr 30, 2025

We can create it in our SNOWFLAKE_INTELLIGENCE.CONFIG database and schema, and call it Festival_Contract_Search

Create Search Service

New service

- Select data
- Select search column
- Select attributes
- Select columns
- Configure indexing



Let's create a new Cortex Search Service

We'll guide you through the steps of selecting a data source, setting service parameters, and creating the Search Service.

Select a warehouse to power the service. This warehouse will be used for materializing the results of the source query upon creation and refresh.

Role and Warehouse

SNOWFLAKE_INTELLIGENCE_ADMI... • SNOWFLAKE_INTELLIGENCE_WH (...)

Database and Schema

SNOWFLAKE_INTELLIGENCE.CONFIG ▾

Name

Festival_Contract_Search

Cancel

Next: Select data

Going to the next screen, it asks for the table that it wants to index. We will select a table that we uploaded earlier. This is in the SI_EVENTS_HOL.PUBLIC and our table is called CONTRACTS

Select data to be indexed

Select a table or view containing containing text data that you'd like to search over. At least one column in this table or view must be a text column that you'd like to search over.

If you wish to specify multiple data sources or perform transformations when defining your service, please use the [SQL surface](#).

SI_EVENTS_HOL.PUBLIC ▾

Search Tables and Views

CONTRACTS ✓

CUSTOMERS

EVENTS

TICKET_SALES

The next screen asks us to choose which column we want to be able to search. In our case, the column name is TEXT. Go ahead and select that column from the list

Select a search column

The data in this column must be text-based and will be the column that your queries search over.

CONTRACTS

Search Column

DOCUMENT_TITLE

URL

TEXT ✓

Next, it is asking what attributes we want to bring in. We will choose both DOCUMENT_TITLE and URL:

Optional

Select attribute column(s)

Select a set of columns that you'd wish to use as filters when querying the service.

CONTRACTS

Q Search Column

Column	Selected
A DOCUMENT_TITLE	<input checked="" type="checkbox"/>
A URL	<input checked="" type="checkbox"/>
A TEXT	

We will leave all columns on the next page selected:

Optional

Select columns to include in the service

Select the columns that you want to include in the search index. The data in these columns will be included in the search service and will be available for querying.

Search and attribute columns must be part of the included columns.

CONTRACTS

Search Column

A DOCUMENT_TITLE

A URL

A TEXT

Select all

[Previous: Select attributes](#)

[Next: Configure indexing](#)

As this is a demonstration with unchanging data, the chosen lag time is inconsequential. Therefore, a 1-day lag can be applied.

Configure your Search Service

Select your desired configuration parameters for the service.

Target Lag

1 days



Target Lag parameter specifies the maximum frequency with which the service will check for and materialize updates based on changes to its source data.

Embedding model (optional)

snowflake-arctic-embed-m-v1.5

Each model may incur a different cost per million input tokens processed. Refer to the [Snowflake Service Consumption Table](#). For more information about each embedding model, refer to the [Cortex Search documentation](#).

Warehouse for indexing

- SNOWFLAKE_INTELLIGENCE_WH

Indexing can take couple minutes to an hour depending on size of data. [See documentation](#).

Now our Search Service is created:

The screenshot shows the Cortex Search interface for the 'FESTIVAL_CONTRACT_SEARCH' service. At the top, there's a back arrow, the text 'Cortex Search', the service name 'FESTIVAL_CONTRACT_SEARCH', and a 'PREVIEW' button. Below this, there are tabs for 'Search Service' (which is active), 'Data Preview', and 'Costs'. Under the 'Search Service' tab, it says 'SNOWFLAKE_INTELLIGENCE.CONFIG'. The main section is titled 'FESTIVAL_CONTRACT_SEARCH'. It shows a table with three columns: 'COLUMN', 'SEARCHABLE', and 'ATTRIBUTE'. The rows are 'TEXT' (with a checkbox in the SEARCHABLE column), 'DOCUMENT_TITLE' (with a checkbox in the ATTRIBUTE column), and 'URL' (with a checkbox in the ATTRIBUTE column). At the bottom left, it says '0 rows'.

NOTE:

If your search service hangs in 'Initialize' for more than a few minutes, it might have failed. This is going to be fixed in the UI in the future. However, right now you can run: [DESC CORTEX SEARCH SERVICE FESTIVAL_CONTRACT_SEARCH](#); and look at the indexing_error column, chances are it is a missing permission on an object such as a database, schema, table, or warehouse.

Step 9: Add to Agent

We will reuse the previously created "Music Festival Agent" and integrate the Search Service we built into it. To do this, select the existing agent and then proceed to add the Search Service.

The screenshot shows the Snowflake AI & ML interface. On the left, a sidebar lists various categories: Home, Search, Projects, Data, Data Products, AI & ML (with Studio, Cortex Search, Features, Models, Evaluations, Document AI), and Agents. The Agents category is highlighted with a blue background and a red arrow pointing to it. The main area displays a table titled 'Agents' with three rows: 'DataAgent', 'Claude', and 'Music Festival Agent'. A red arrow points to the 'Music Festival Agent' row. To the right, there are sections for 'Response instruction' and 'Knowledge'. The 'Knowledge' section includes a 'Snowflake' icon, 'Semantic models', 'Snowflake Data', and 'Search services'. Below these are two buttons: '+ Semantic model' and '+ Search service', with a red arrow pointing to the '+ Search service' button.

We will then add the search service by selecting where we created it, which was in SNOWFLAKE_INTELLIGENCE.CONFIG we will call it Festival Contract Documents, and make sure we define the URL column in our search service, which is called URL:

Add search service

SNOWFLAKE_INTELLIGENCE.CONFIG

FESTIVAL_CONTRACT_SEARCH 

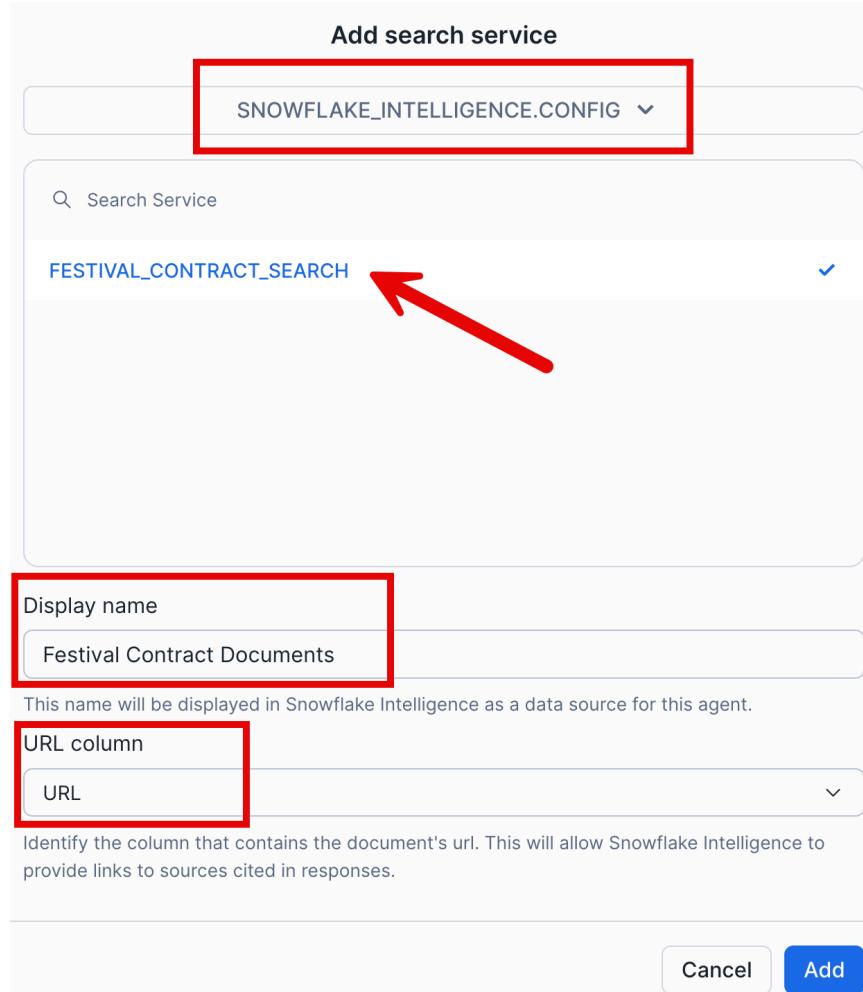
Display name
Festival Contract Documents

This name will be displayed in Snowflake Intelligence as a data source for this agent.

URL column
URL

Identify the column that contains the document's url. This will allow Snowflake Intelligence to provide links to sources cited in responses.

Cancel **Add**



The service needs to be active before it can function, which will take a few minutes. Completion is indicated when the Serving State shows as ACTIVE.

- You may run into an error that resembles this: Manual refresh failed. SQL compilation error: Failed to refresh dynamic table with refresh_trigger MANUAL at data_timestamp 1747812204711 because of the error: SQL compilation error: Some inputs failed to refresh
- To resolve this
 - a) Make sure the default warehouse has USAGE privileges granted to the SNOWFLAKE_INTELLIGENCE_ADMIN_RL role used in this lab, particularly if the state never changes. Cortex search will create a new dynamic table to store this information. If you are having problems, check out the dynamic tables in the Admin section and make sure that table is working properly.
 - b) Ensure that SNOWFLAKE_INTELLIGENCE_ADMIN_RL has access to the table by doing a GRANT on SELECT to the role.

NAME	DATABASE	SCHEMA	INDEXING ST...	SERVING ST...
FESTIVAL_CONTRACT_SEARCH	SNOWFLAK...	CONFIG	ACTIVE	INITIALIZING
TRANSCRIPTS_SEARCH_SERVICE	AICOLLEGE	PUBLIC	ACTIVE	ACTIVE
RESUME_SEARCH_SERVICE	DATASCIE...	PUBLIC	ACTIVE	ACTIVE
RECOGNITION_MSG_SERVICE	AA	PUBLIC	ACTIVE	ACTIVE

Step 10: Test with Updated Agent

Let's go back to Snowflake Intelligence and make sure our new search service is available to query, and then

Agents

3 Agents

AGENT NAME
DataAgent
Claude
Music Festival Agent

Agents

Snowflake Intelligence

Good afternoon,

How can I help you?

How may I assist you?

Music Festival Agent ▾ Festival Contract ... ▾ Upload files

Festival Contract Documents ▾

Snowflake Data ▾



Now that we see our Snowflake Data there, we know it will utilize our search service we created. Let's ask it a question:

What is the cancellation policy for the band Algorhythms?

Notice that no SQL is generated or written, as it is using our search service. It gives us the answer and the source of the document from which this information came.

What is the cancellation policy for the band Algorhythms?

✓ Thinking completed Show Details ▾

1 🎵 The Algorhythms wil... 2 🎵 The Algorhythms wil... 3 🎵 Sponsor agrees to p... Show More

Here's the cancellation policy for the Algorhythms:

If the Festival cancels:

- 25% of the guarantee fee (\$100,000) will be paid if canceled between 60 and 30 days before the performance ①
- 50% of the guarantee fee will be paid if canceled less than 30 days before the performance ①

If the Algorhythms (Artist) cancels:

- No fee is owed by the Promoter ①

Step 11: Review

This lab explored Snowflake Intelligence and its applications. Snowflake Intelligence enables analysts to efficiently query Snowflake data by leveraging a semantic model or the Cortex search service for data indexing.

Our steps were:

1. Setup metadata for Snowflake Intelligence (this part will go away in the future)
2. Create a dumb agent that just used Claude with no Snowflake data
3. Created a Semantic Model
4. Created a new agent to utilize the new semantic model and Snowflake Data
5. Created a cortex search service
6. Updated our agent to utilize that search service
7. Wrote prompts in Snowflake Intelligence that utilized our semantic model and/or our search service to get data from our Snowflake instance

Customers can now easily build a chatbot-like interface for analysts and search their Snowflake assets using simple wizards, requiring minimal coding. This eliminates the need for complex custom Python scripts (though that remains an option), enabling faster value realization with GenAI within Snowflake.

Step 12: Validate Using DORA

Congrats! You have completed the lab. Please run the following commands in Snowsight to confirm your completion.

 **Validate Agents were created**

```
SQL
use database SNOWFLAKE_INTELLIGENCE;
use schema AGENTS;

SELECT
    util_db.public.se_grader(
        step,
        (actual = expected),
        actual,
        expected,
        description
    ) AS graded_results
FROM
(
    SELECT
        'SEDW30' AS step,
        (
            SELECT CASE WHEN COUNT(*) > 1 THEN 1 ELSE 0 END FROM CONFIG
        ) AS actual,
        1 AS expected,
        'Agents created for Snowflake Intelligence' AS description
);
```

Validate YML and semantic layer were created

```
SQL
use database SNOWFLAKE_INTELLIGENCE;
use schema CONFIG;

SELECT
    util_db.public.se_grader(
        step,
        (actual = expected),
        actual,
        expected,
        description
    ) AS graded_results
FROM
(
    SELECT
        'SEDW31' AS step,
        (
            SELECT COUNT(*)
            FROM DIRECTORY(@semantic_models)
            WHERE RELATIVE_PATH LIKE 'music_festival.yaml'
        ) AS actual,
        1 AS expected,
        'Semantic Model was created for Snowflake Intelligence' AS
description
);
```

Validate Cortex Search Service Created

```
SQL
use database SNOWFLAKE_INTELLIGENCE;
use schema CONFIG;

SELECT
    util_db.public.se_grader(
        step,
        (actual = expected),
        actual,
        expected,
        description
    ) AS graded_results
FROM
(

```

```
SELECT
    'SEDW32' AS step,
    (
        SELECT COUNT(*)
        FROM INFORMATION_SCHEMA.CORTEX_SEARCH_SERVICES
        WHERE SERVICE_NAME = 'FESTIVAL_CONTRACT_SEARCH'
    ) AS actual,
    1 AS expected,
    'Cortex Search Service Created for Snowflake Intelligence' AS
description
);
```

If all validations return , you have **completed** the Snowflake Intelligence Lab 