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.

What you'll do:

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

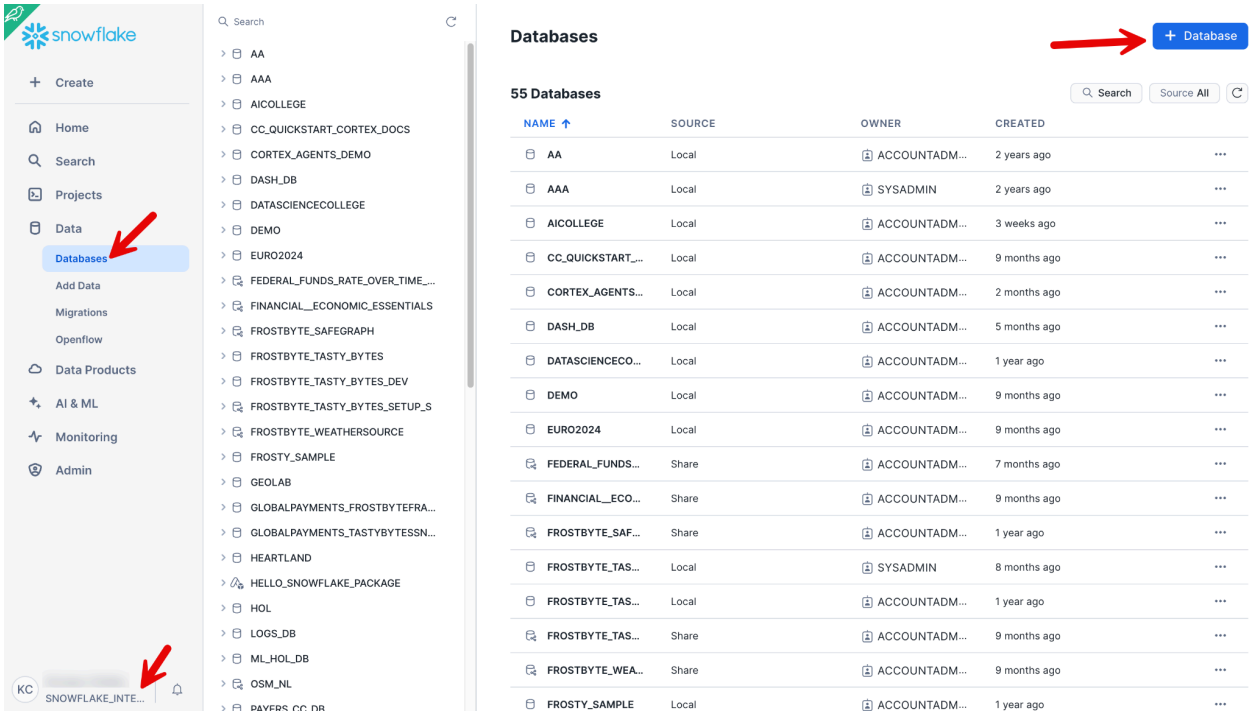
Why it matters:

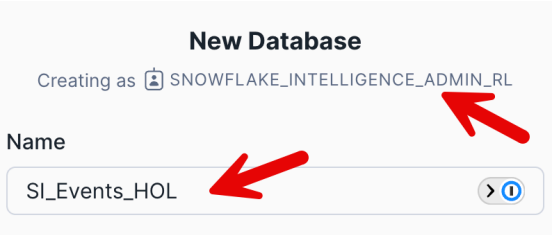
- **Self-serve data exploration:** Create charts and get instant answers using natural language. With an intuitive interface powered by agentic AI, enable teams to discover trends and analyze data without technical expertise or waiting for custom dashboards.
- **Democratize intelligence:** Access and 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 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. In the Data folder you will see all the files needed.

Log into your Snowflake 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 but you can use any database/schema you have write privileges on.



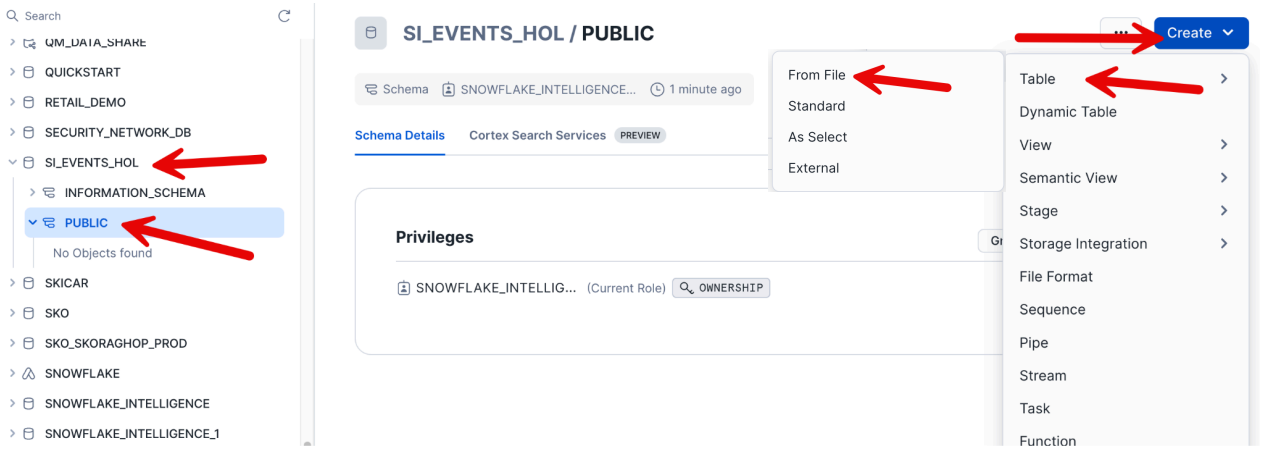


Let's create the tables we need for this lab. There are a total of 4 tables needed for this lab, and they can all be created the same way by creating a table from a file in Snowflake. Please repeat the following steps for all four tables:

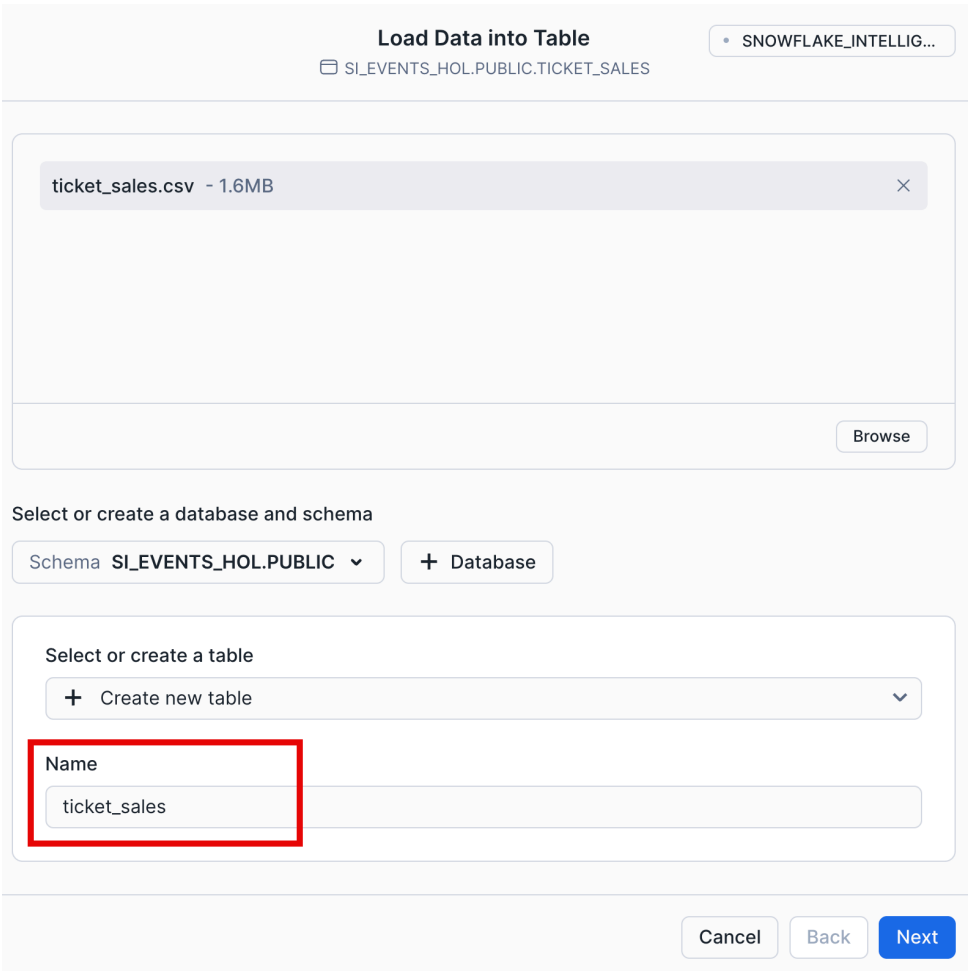
1. Customers
2. Contracts
3. Ticket Sales

4. Events

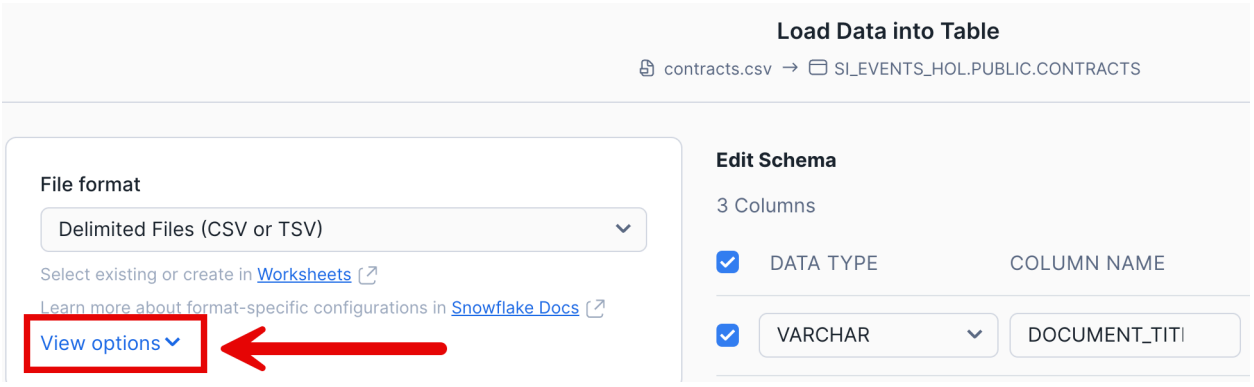
To load a table from a file in Snowsight, go into the database, go to the schema, and click the create button on the top right to create a table from a file:

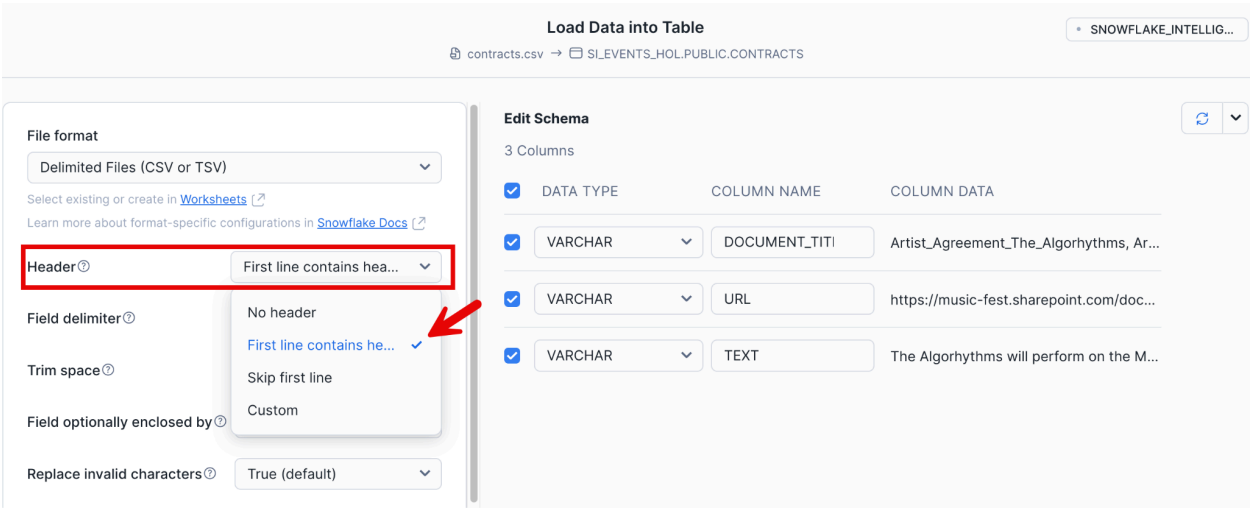


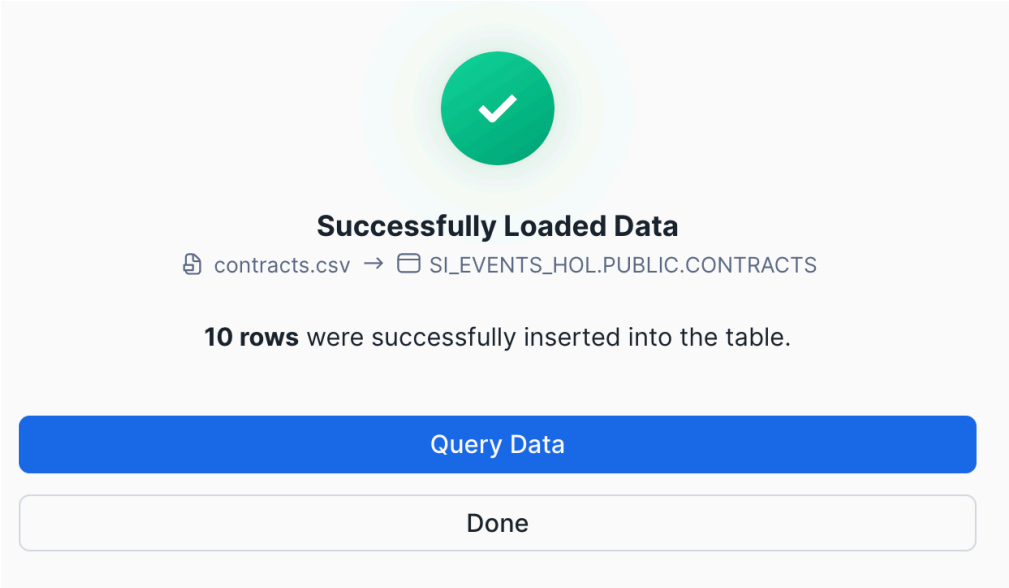
Name each table the same name as the file you unzipped:



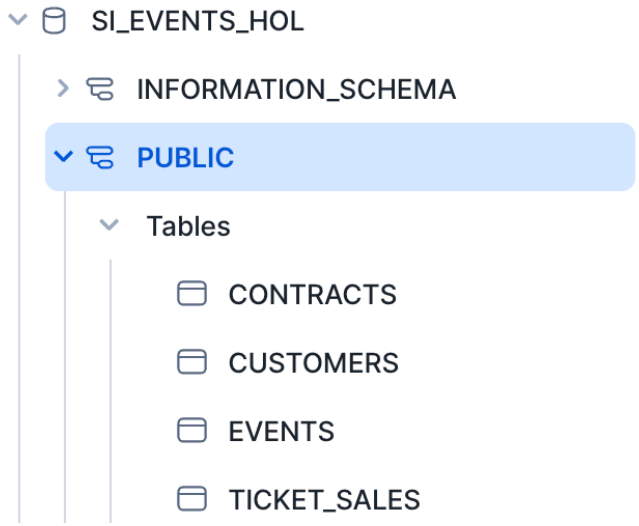
Open the View Options section in the file format area and make sure that the header is the first line of the document:







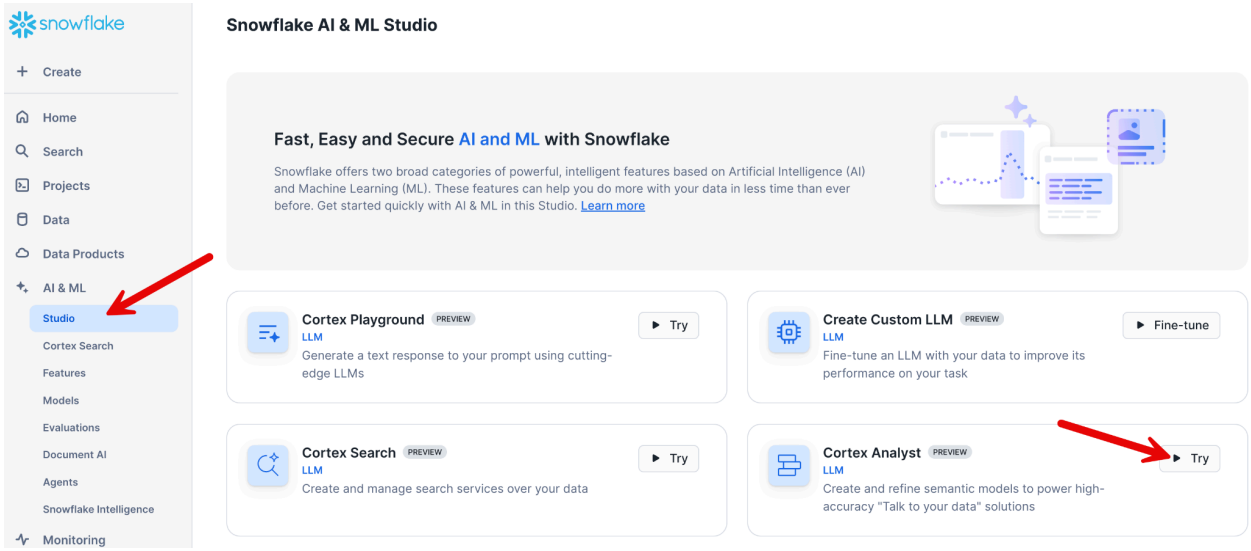
Repeat this for all 4 files so you have all 4 tables needed for the lab. When you are complete, you should see the following under tables in the public schema:



Step 5: Create a Semantic View

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

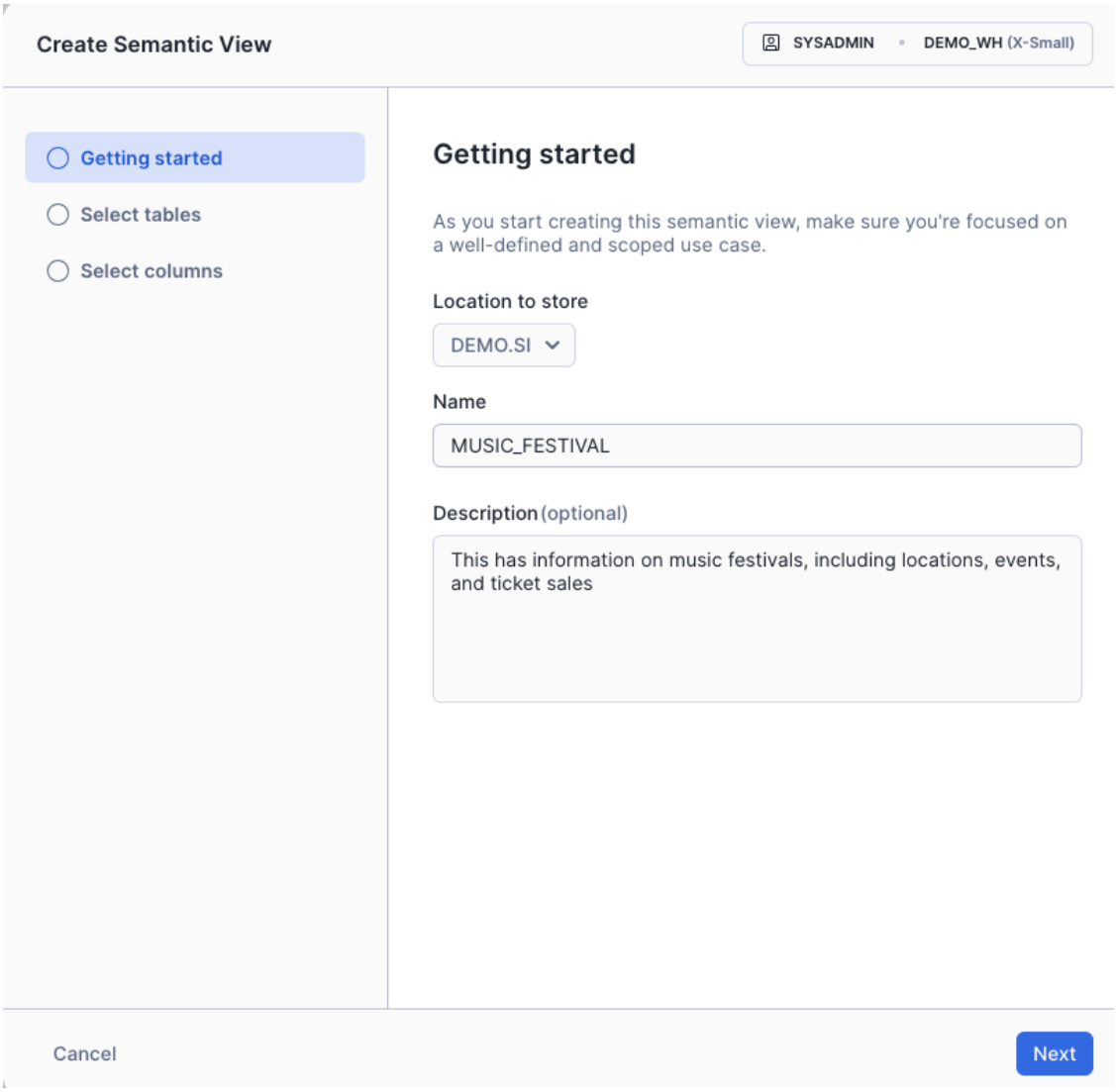
Then we can create a semantic view that we can use in an agent. Let's do that through Snowflake AI & ML Studio using Cortex Analyst:



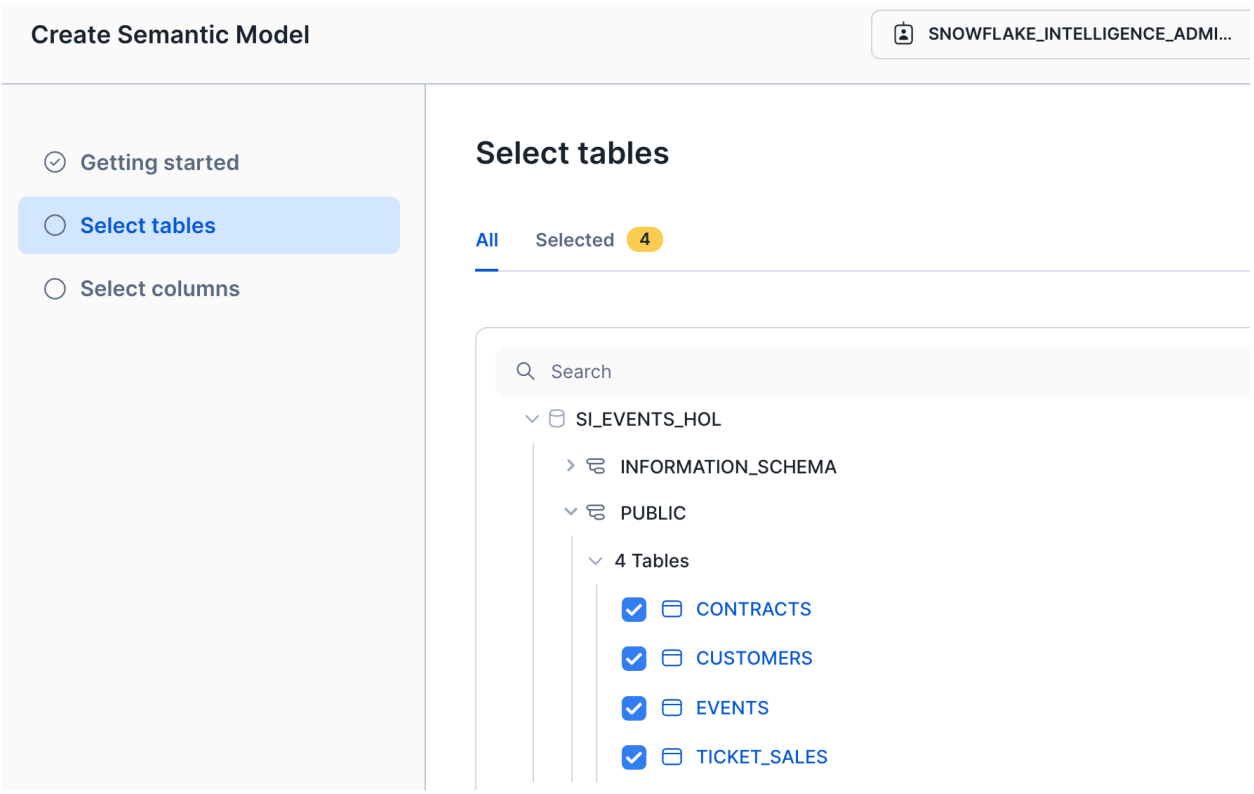
Go ahead and select create new:

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

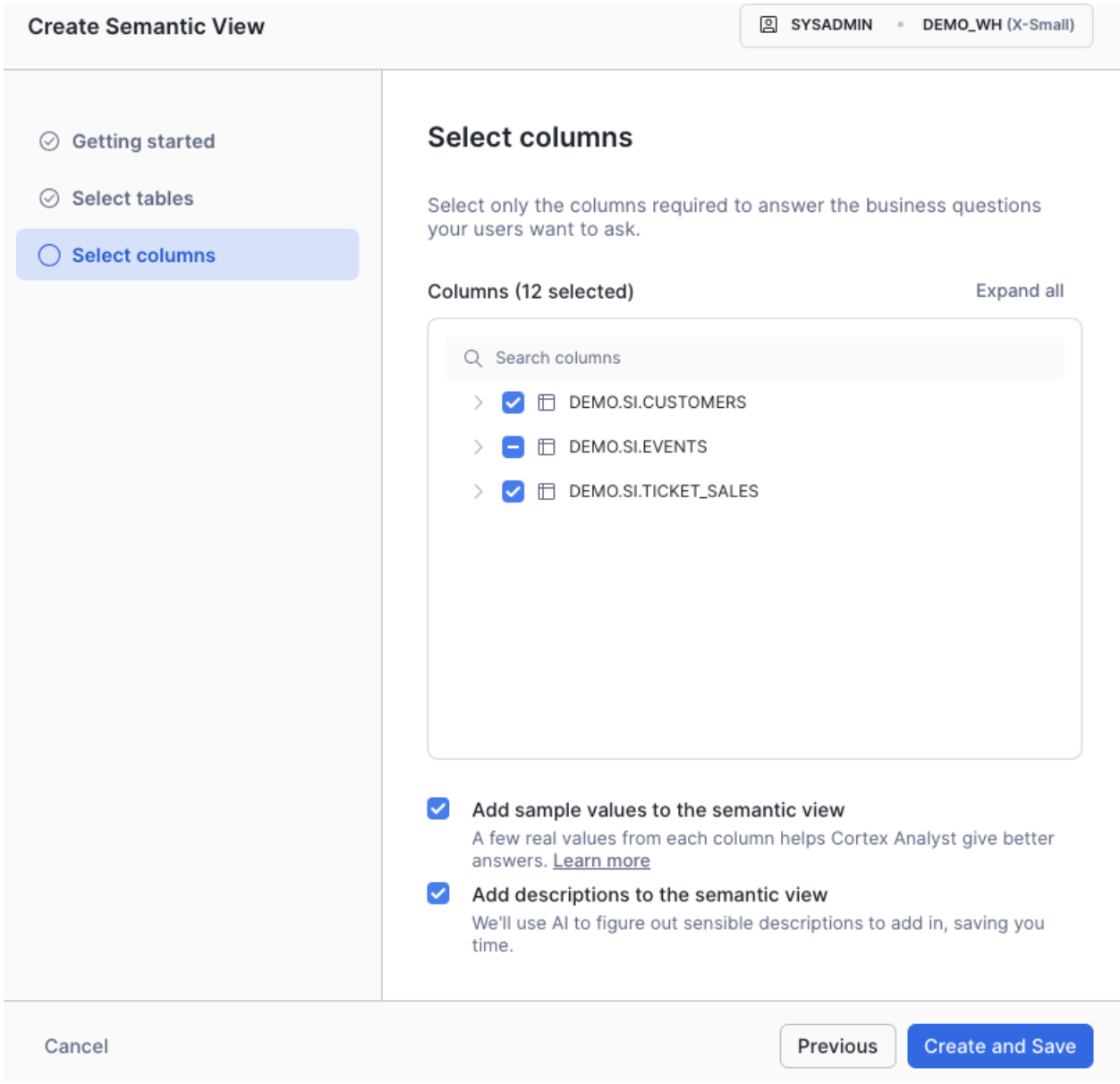


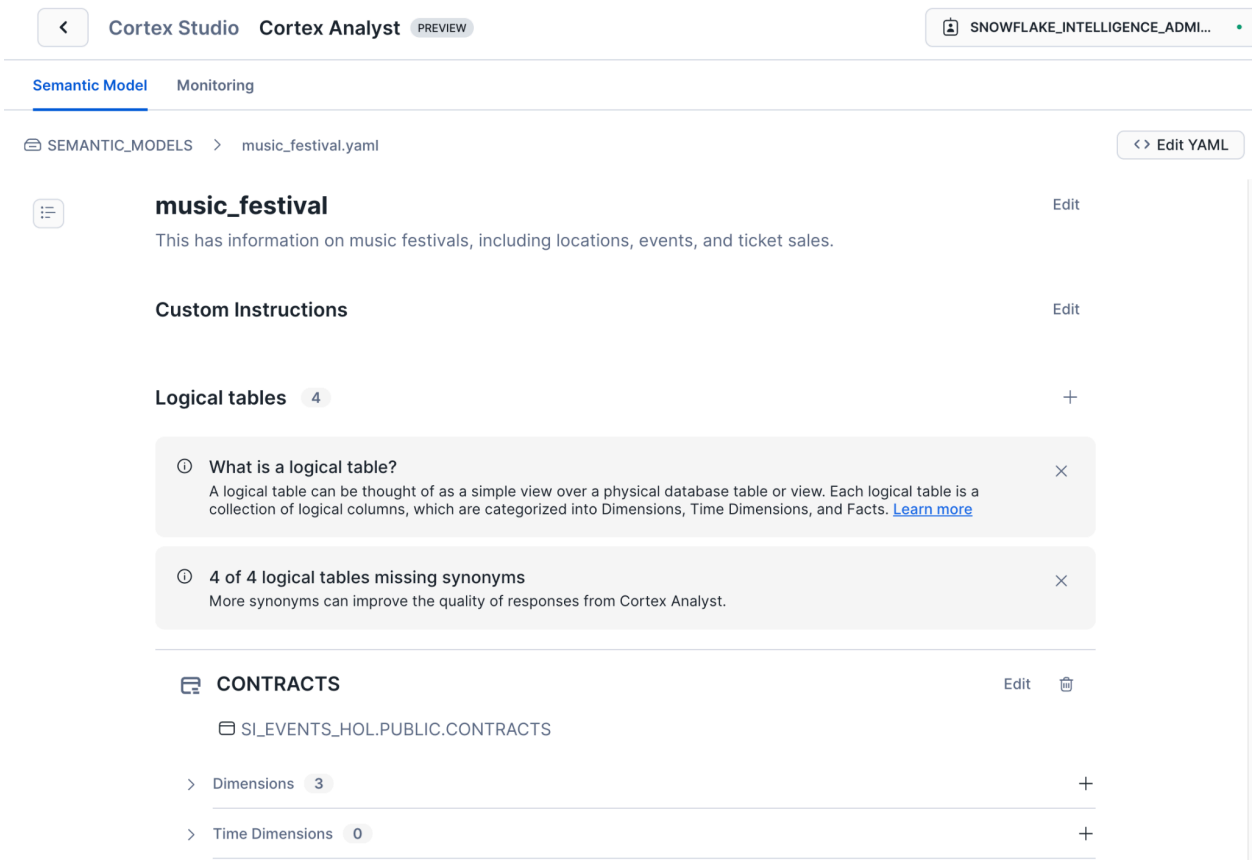
Let's select our tables that we imported at the beginning of the lab:



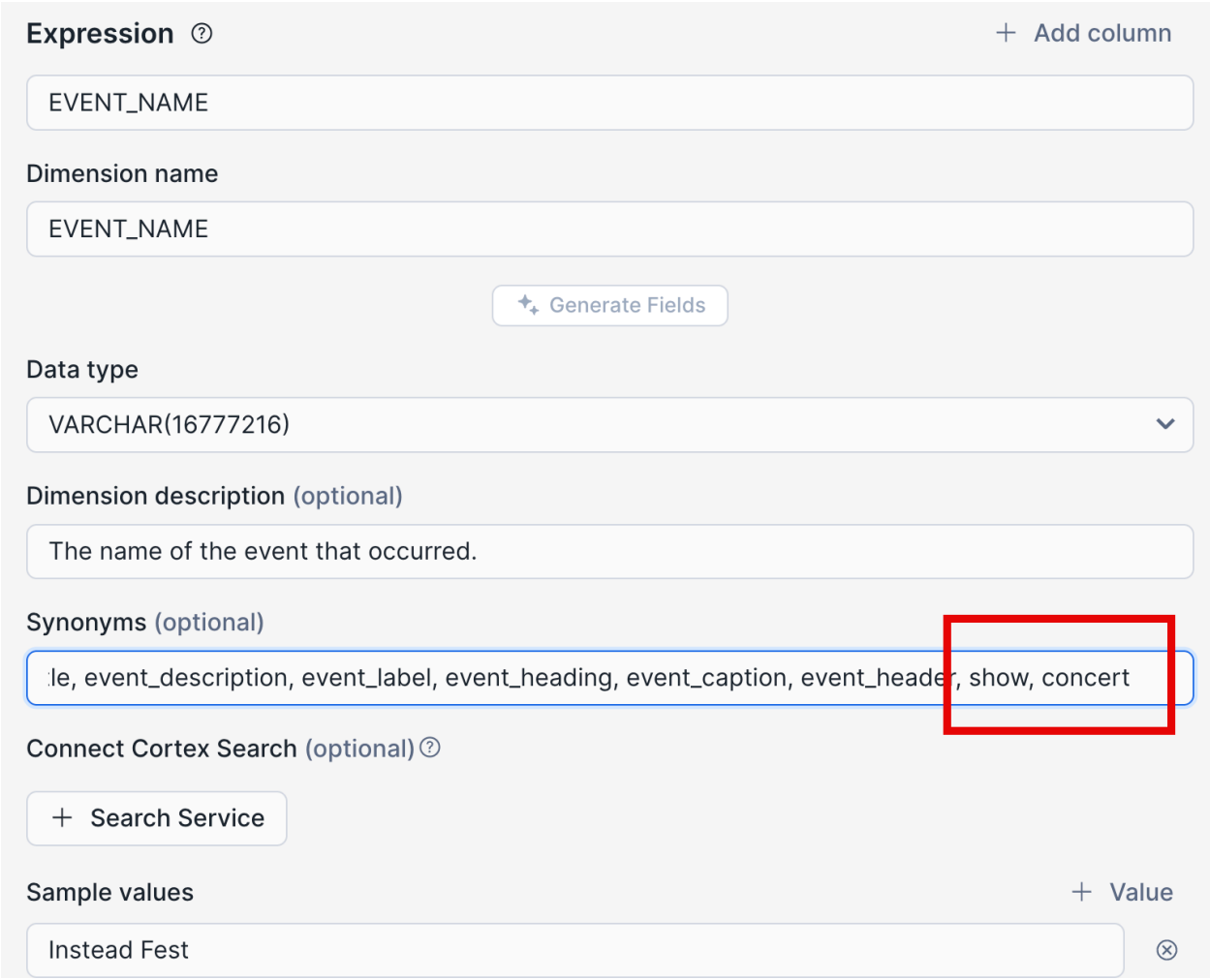
View the Results

Then we can select the columns to choose. Let's just select all columns by choosing the top checkbox:
When we click ‘Create and Save’ it will generate a starter semantic model for us:


Create Semantic View
SYSADMIN • DEMO_WH (X-Small)
Getting started
Select tables
Select columns
Select columns
Select only the columns required to answer the business questions your users want to ask.
Columns (12 selected)
Expand all
Search columns
DEMO.SI.CUSTOMERS
DEMO.SI.EVENTS
DEMO.SI.TICKET_SALES
Add sample values to the semantic view
A few real values from each column helps Cortex Analyst give better answers. Learn more
Add descriptions to the semantic view
We'll use AI to figure out sensible descriptions to add in, saving you time.
Cancel
Previous
Create and Save



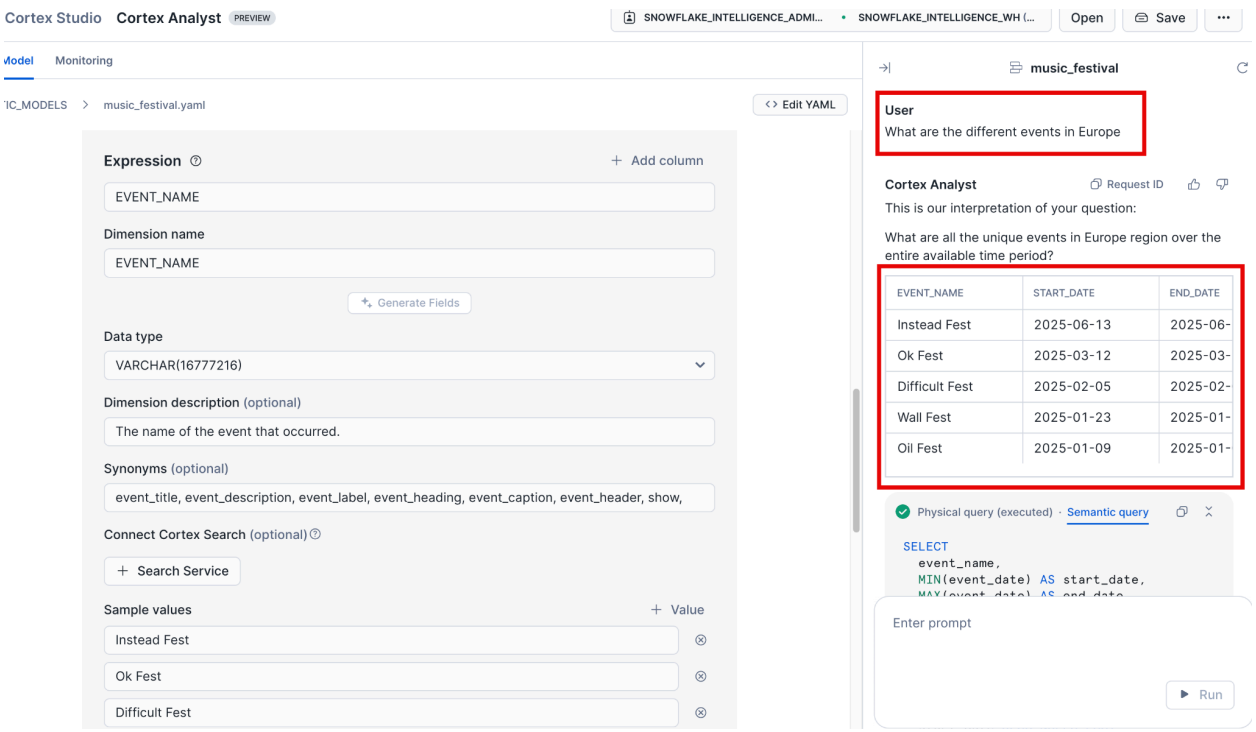
We can see that it has already created a description as well as some other details, like synonyms. Let's go ahead and add ‘show’ and ‘concert’ to the list of synonyms for EVENT_NAME in the EVENTS table:



Test the Model

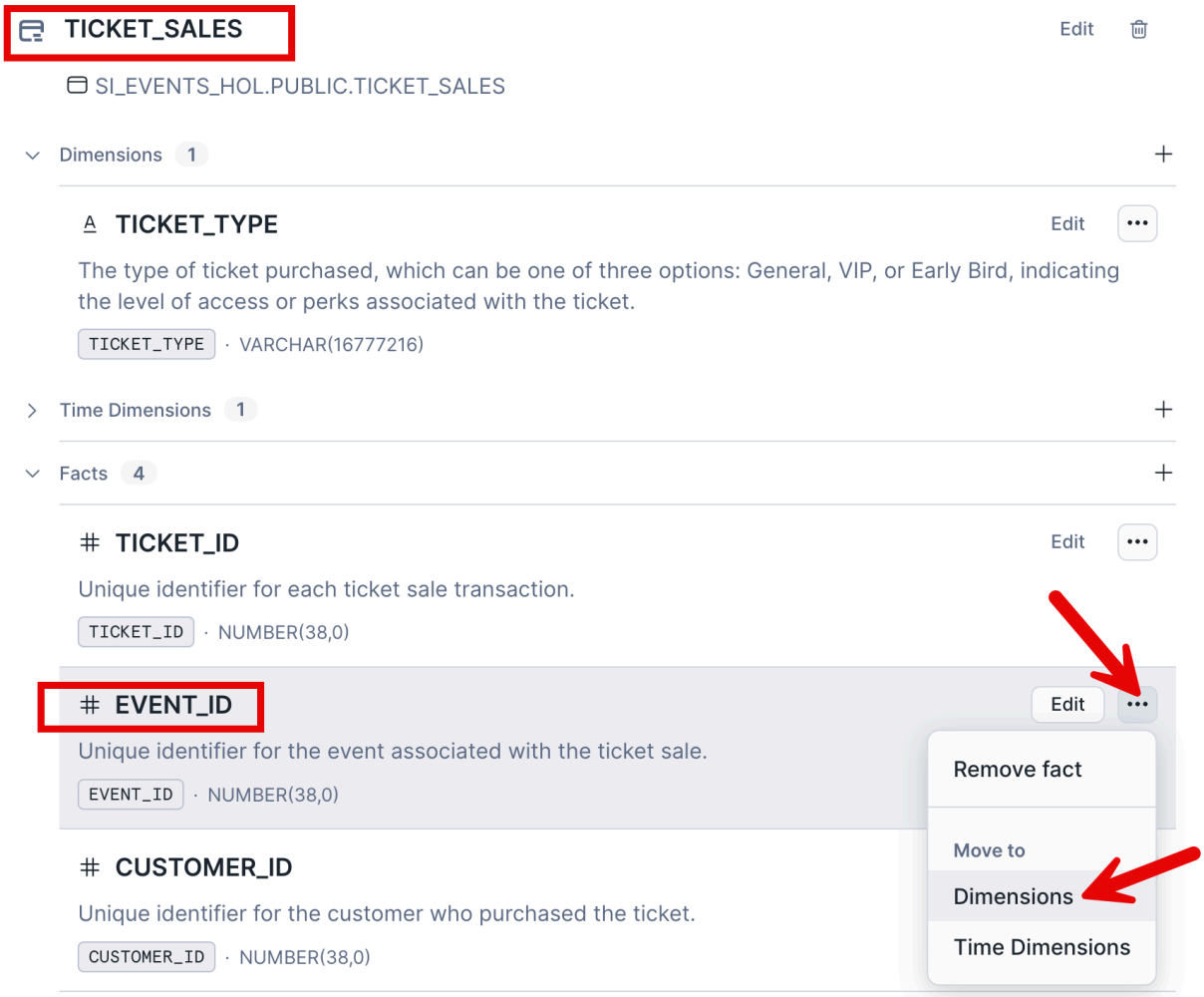
I can test this model right now by going to the side window and putting in the following prompt: *What are the different events in Europe*

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



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.



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

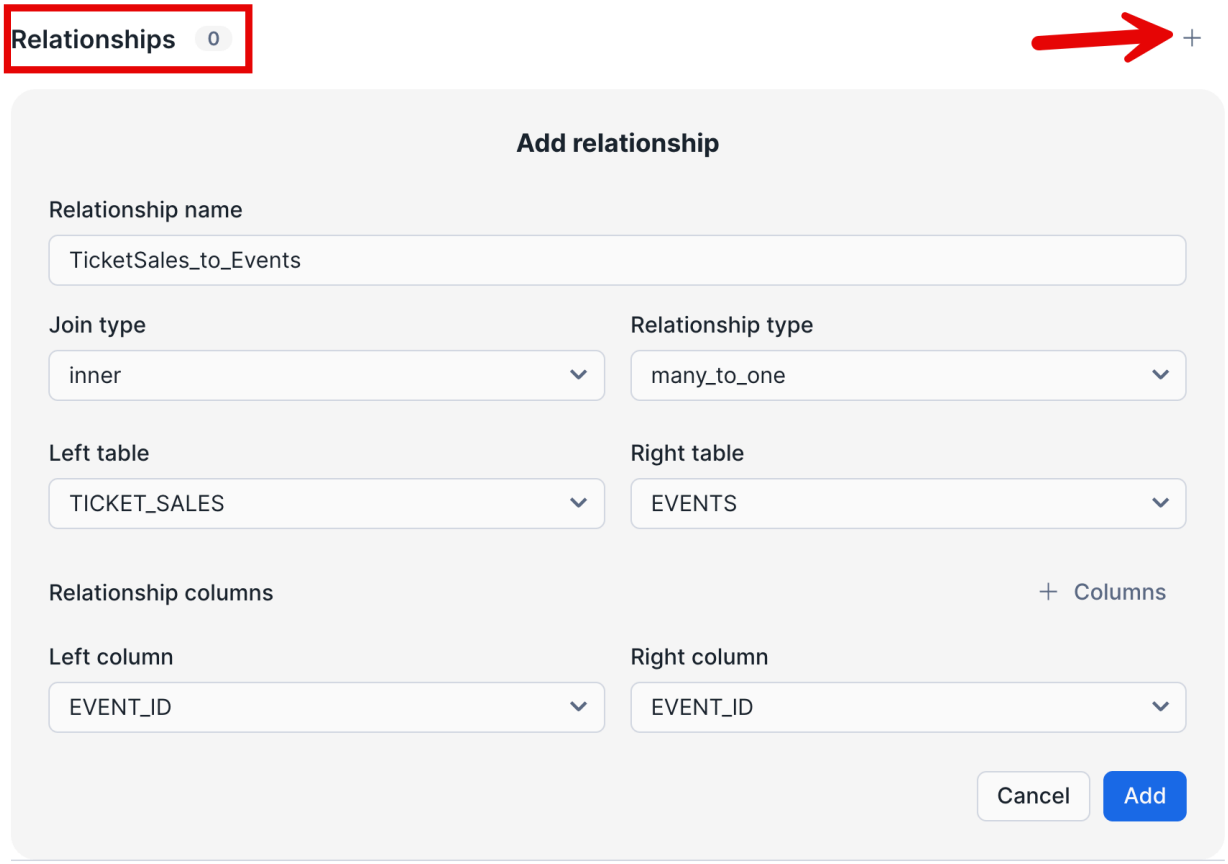
1. TICKET_SALES
    a. TICKET_ID - need to move from fact to dimension
    b. EVENT_ID - need to move from fact to dimension
    c. CUSTOMER_ID - need to move from fact to dimension
2. CUSTOMERS
    a. CUSTOMER_ID - need to move from fact to dimension

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.

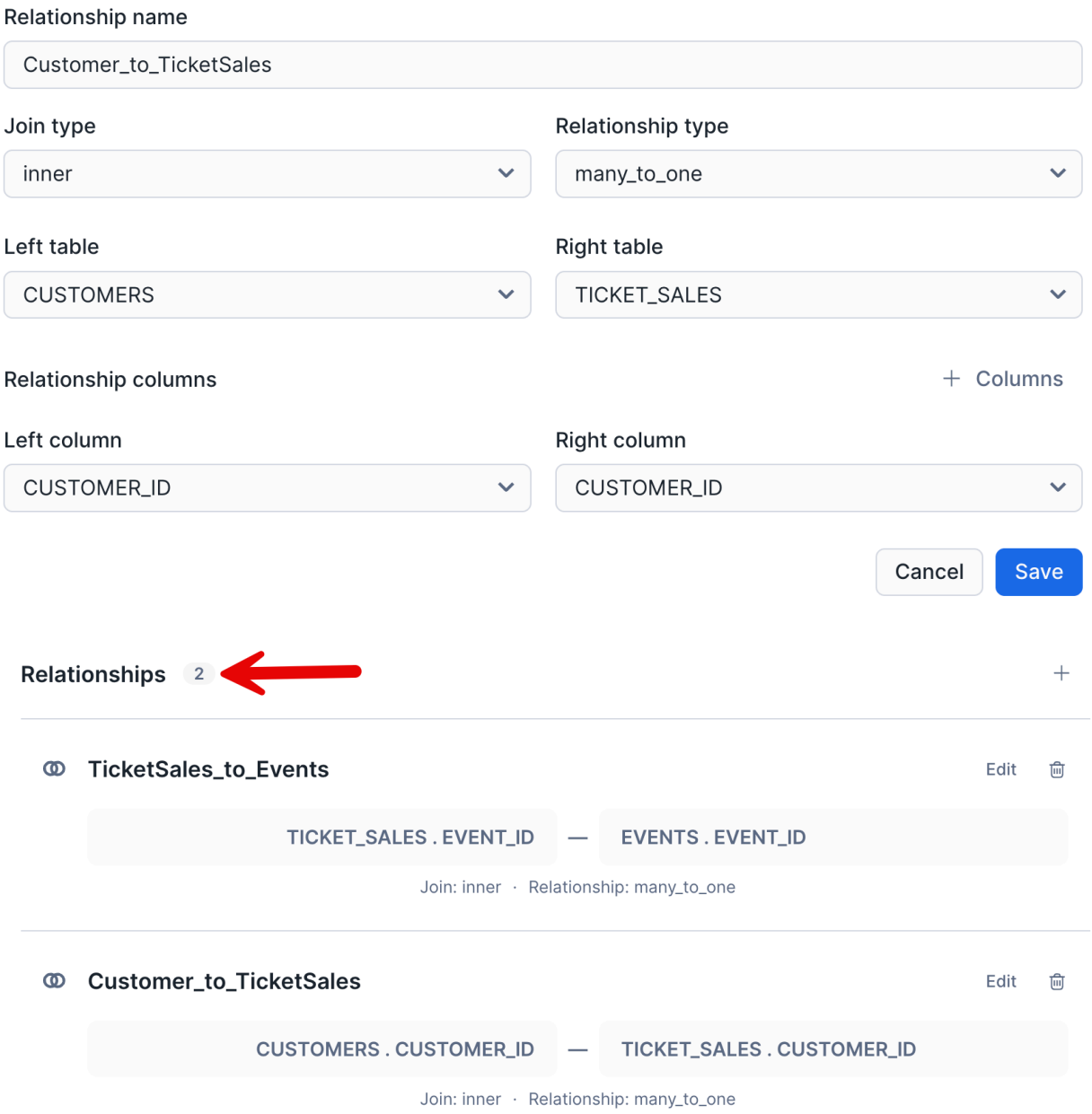
Our first relationship we will define is TICKET_SALES to EVENT. For every EVENT, there are many ticket sales, and the joining column is EVENT_ID. Let's define a relationship that represents that:

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

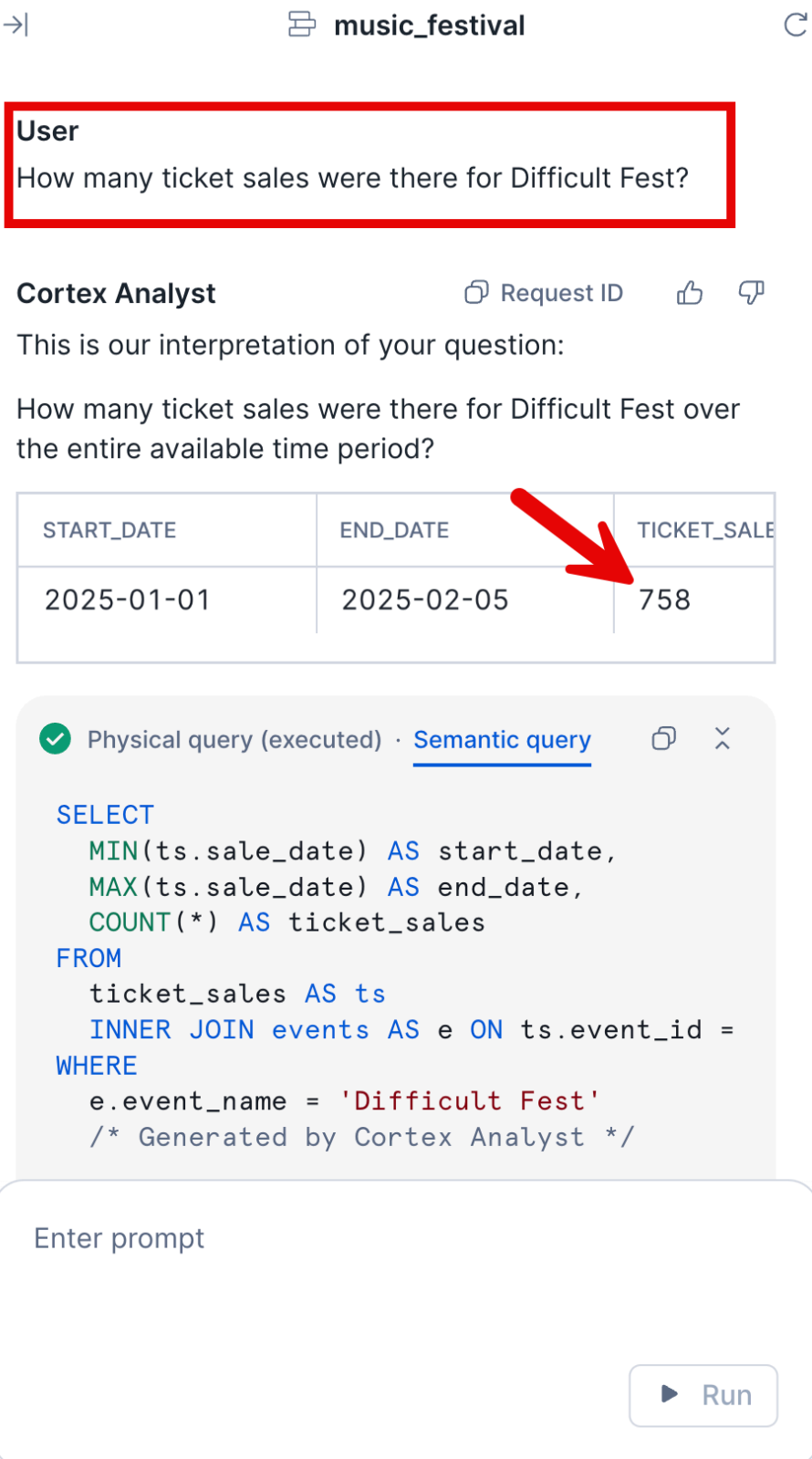


Let's add a second relationship of customers to tickets:

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

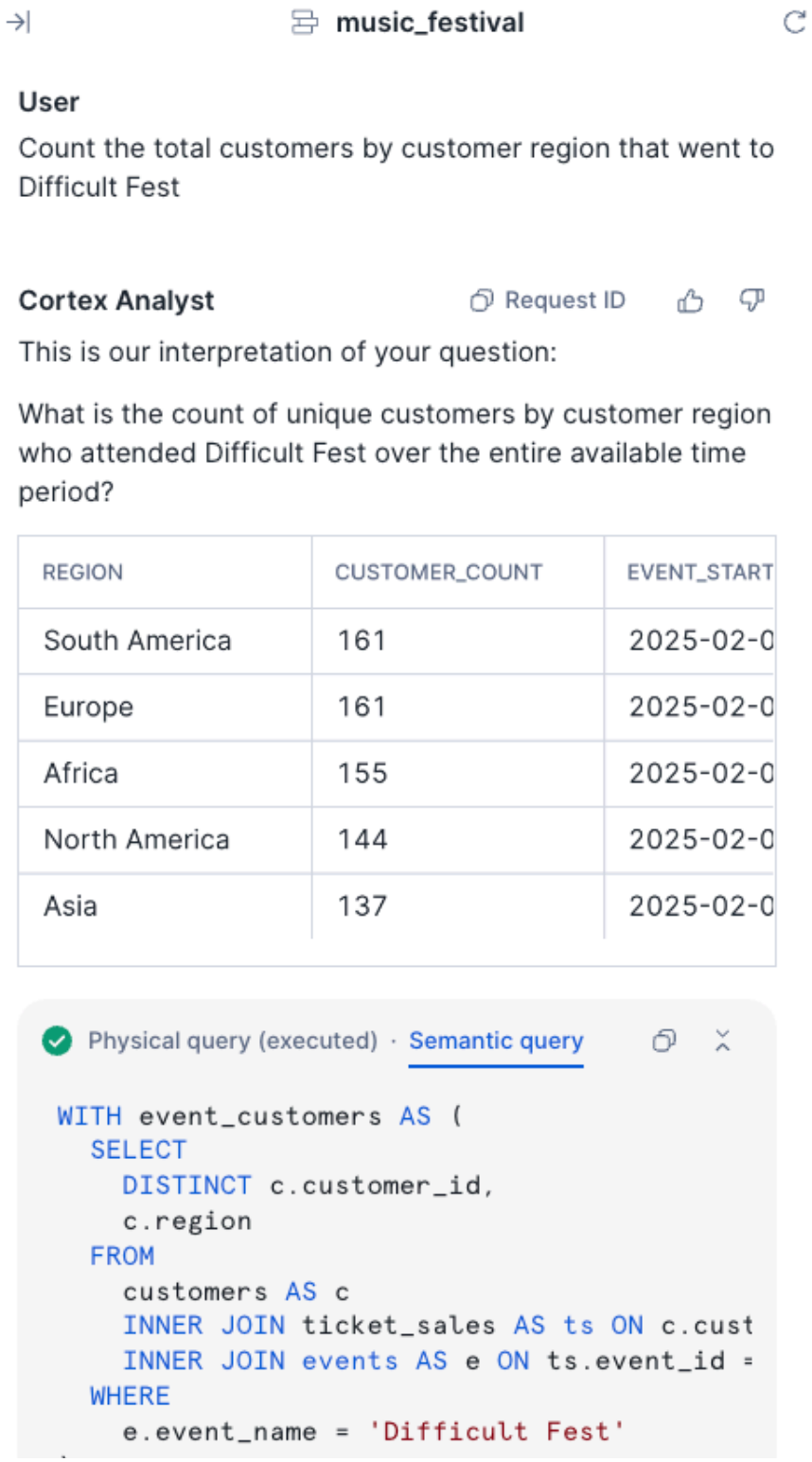


Now we have two relationships set up in our semantic model. Let's go ahead and test them using the prompt on the right side. We can ask this: *How many ticket sales were there for Difficult Fest?*



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.

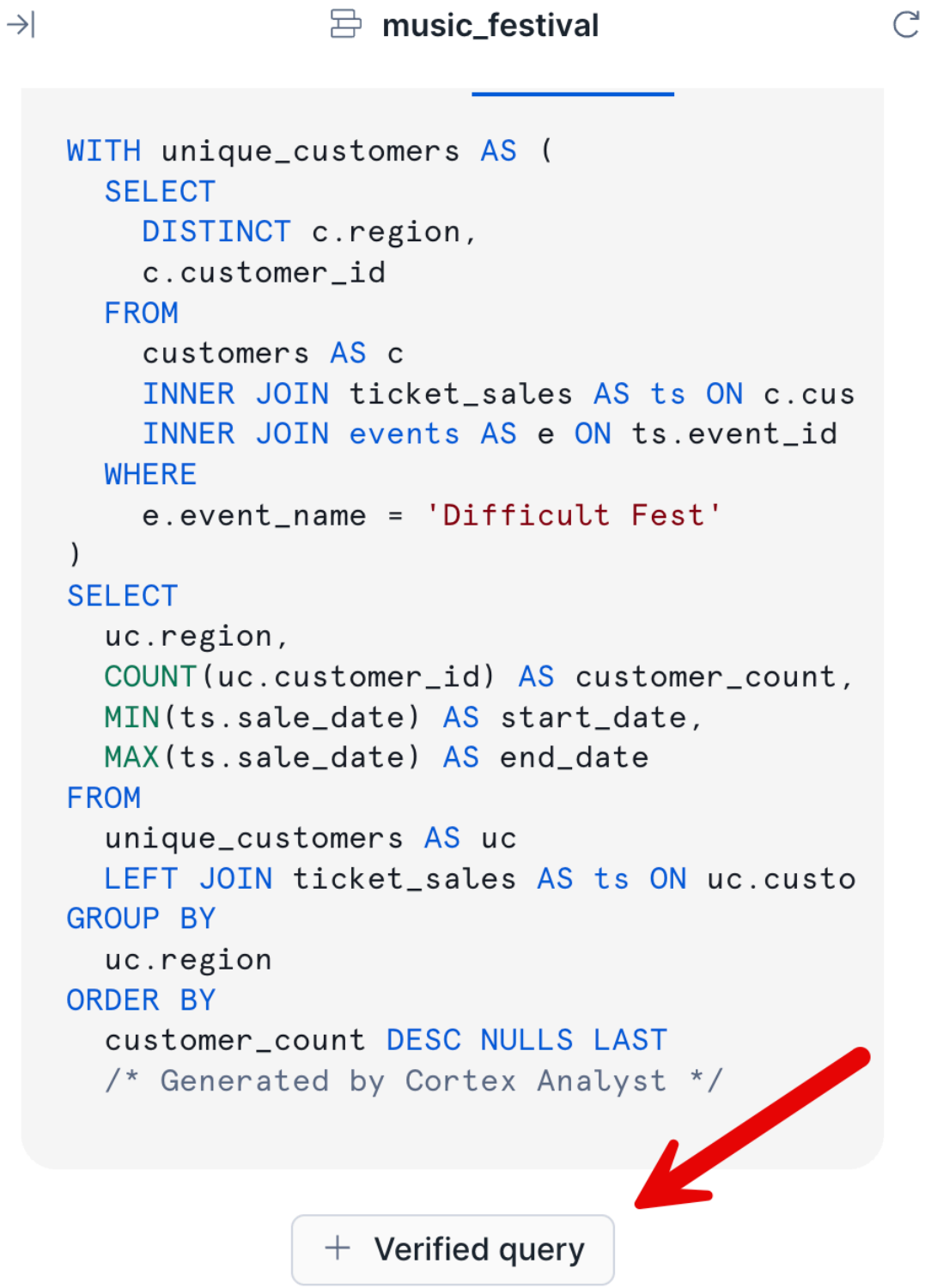
We can also ask it to ‘Count the total customers by customer region that went to Difficult Fest’, and this should span all of our tables in our semantic model and give us values:

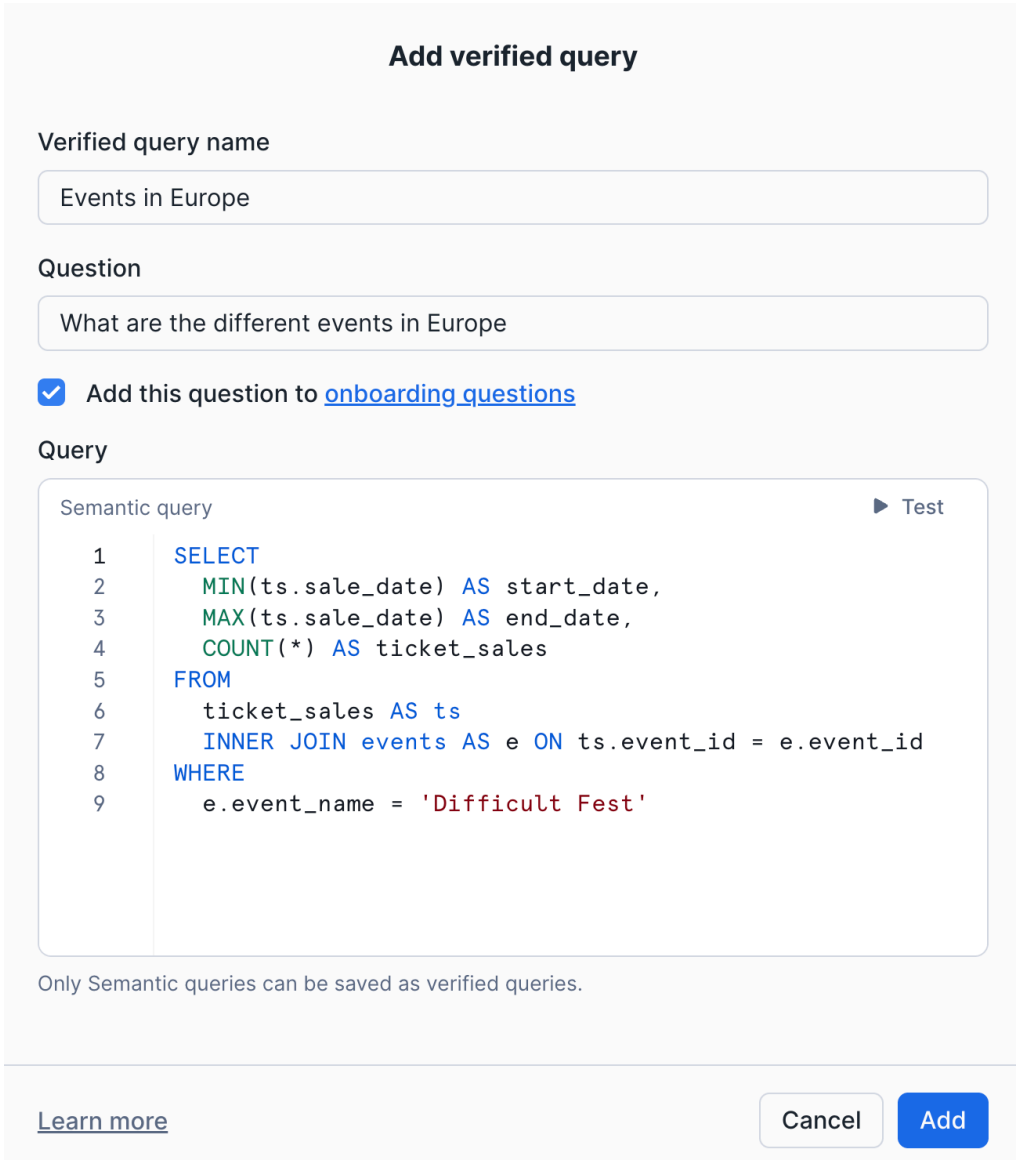


Excellent! This functionality is operating as anticipated.

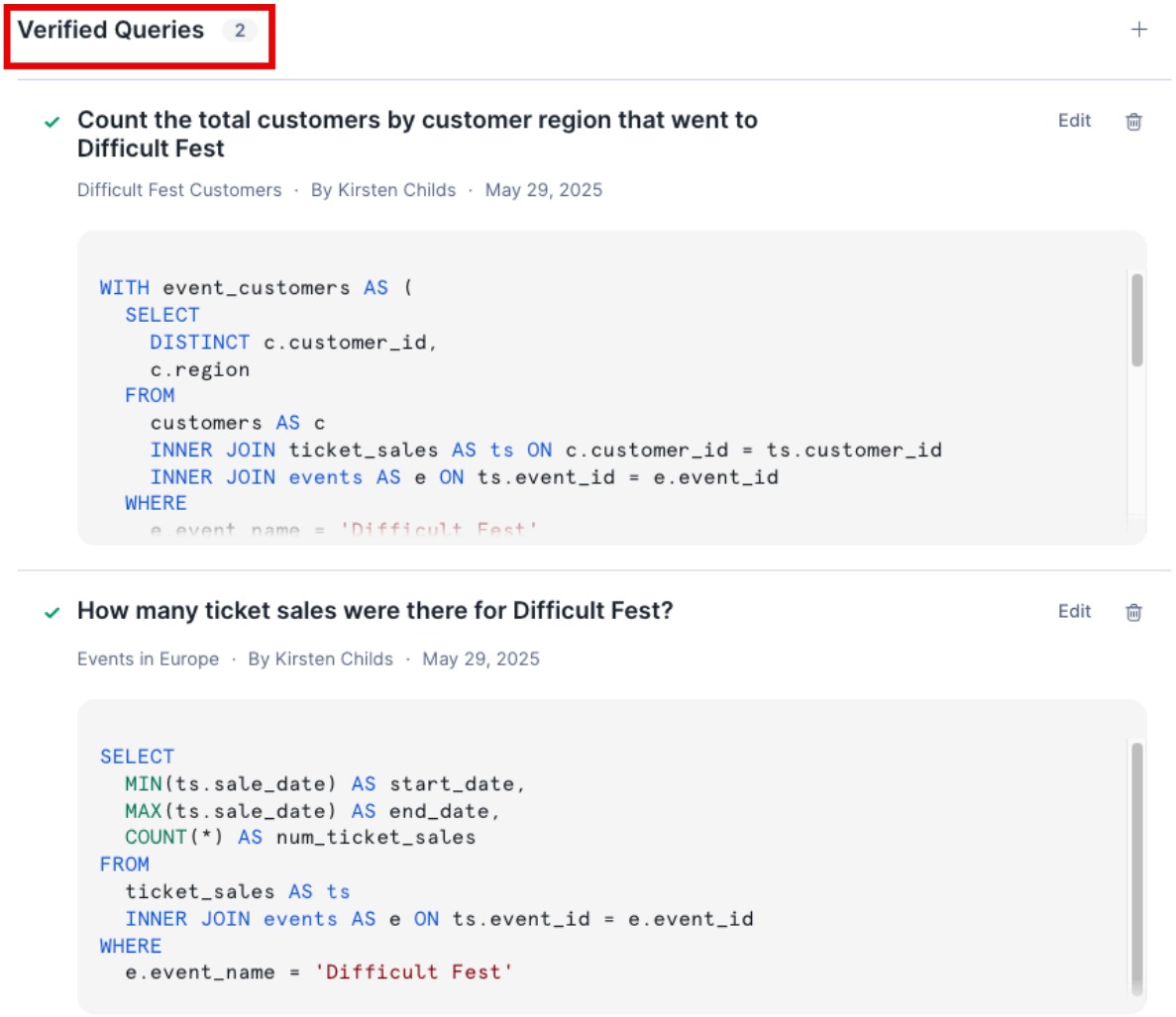
Verify the Queries

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



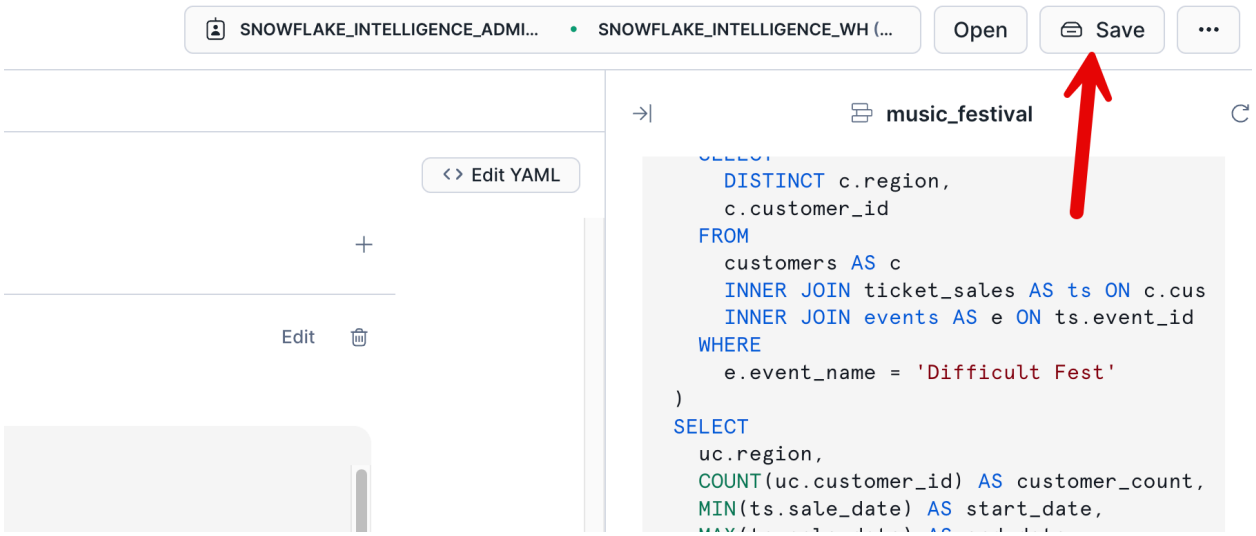


These will now show up under verified queries in our model definition:



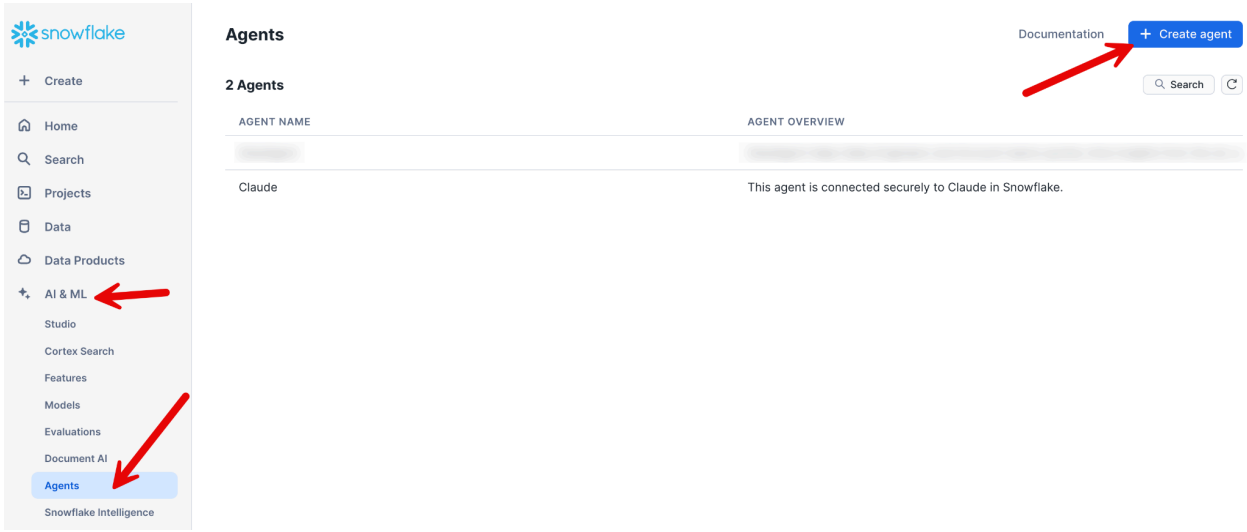
Save the Model

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

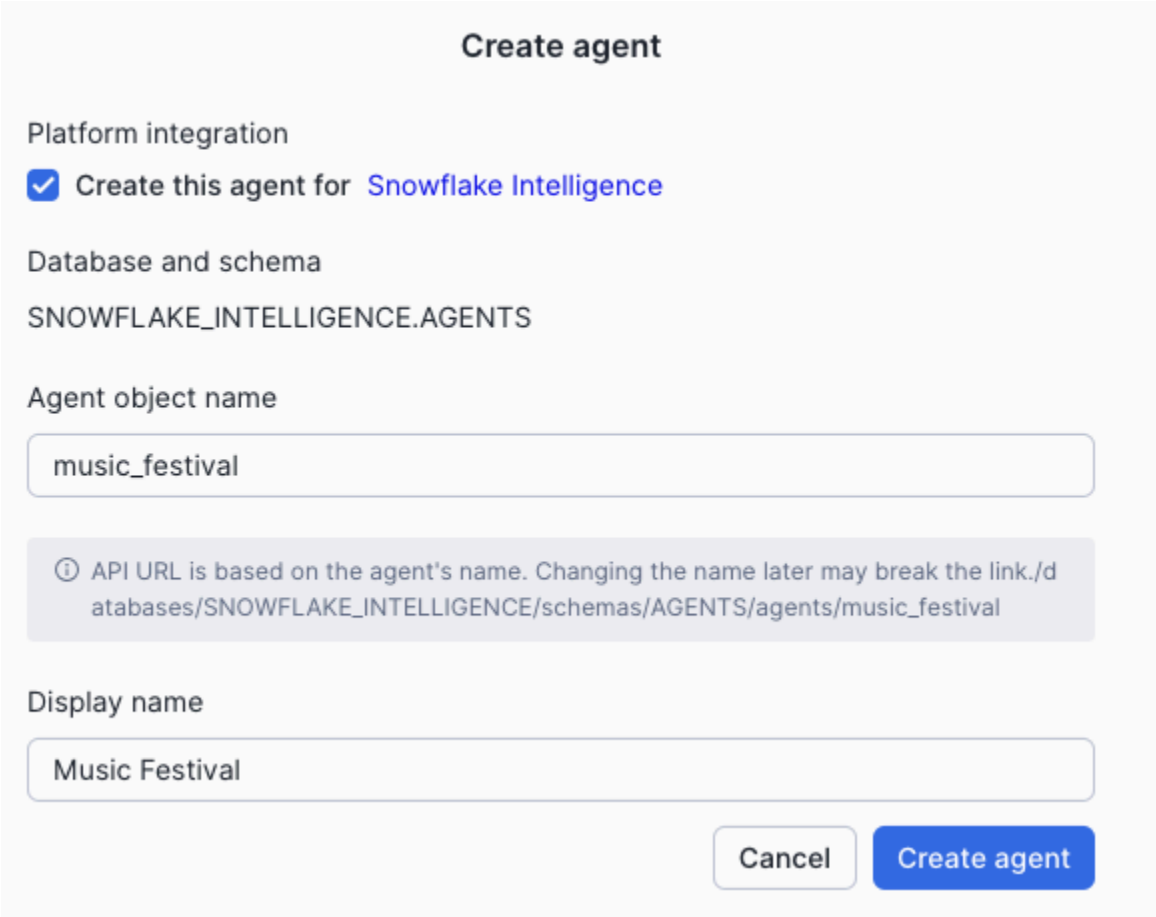


Step 6: Create an Agent to Use the Semantic Model

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



We will use the following information:
Name: Music Festival Agent
Description: An agent that can return information about music festival events and ticket sales



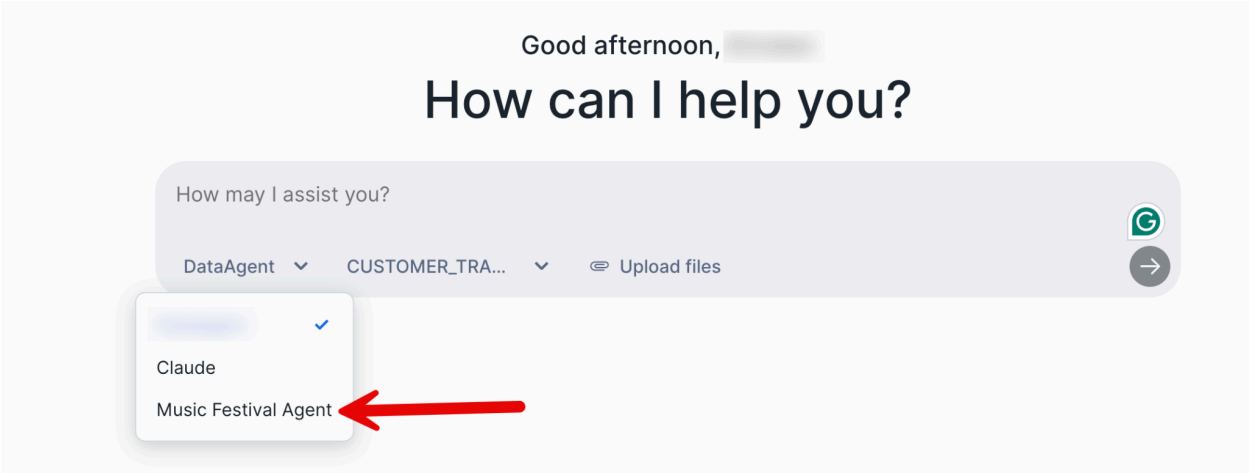
After those are filled in, go ahead and click Create Agent

Now click the agent and in the top right corner click Edit, then head on over to Tools on the left hand side and click + Add to add our Cortex Analyst. Give it a user friendly name and generate the description with cortex.

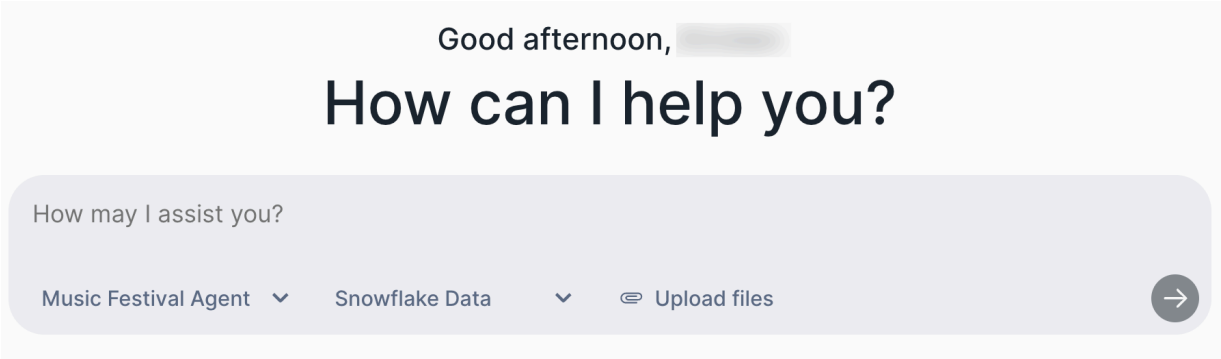
Step 7: Test with Smart Agent

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.

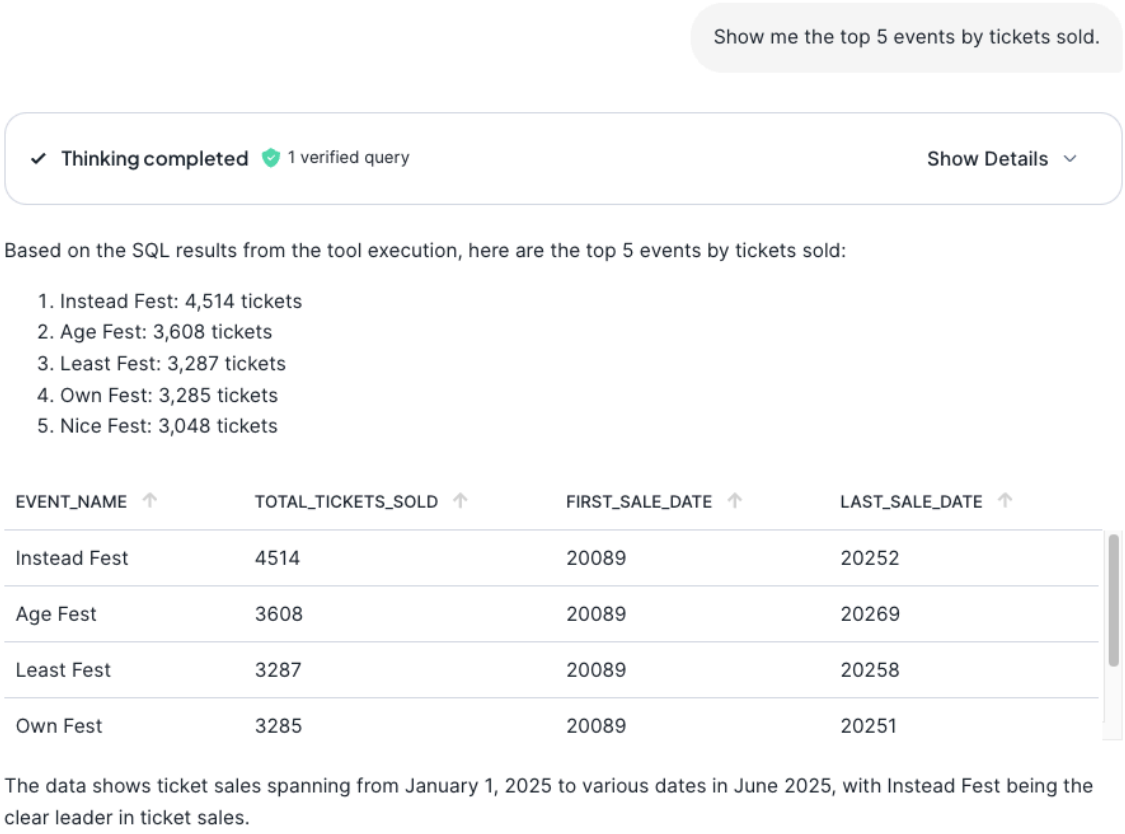
We can now see the Music Festival Agent.



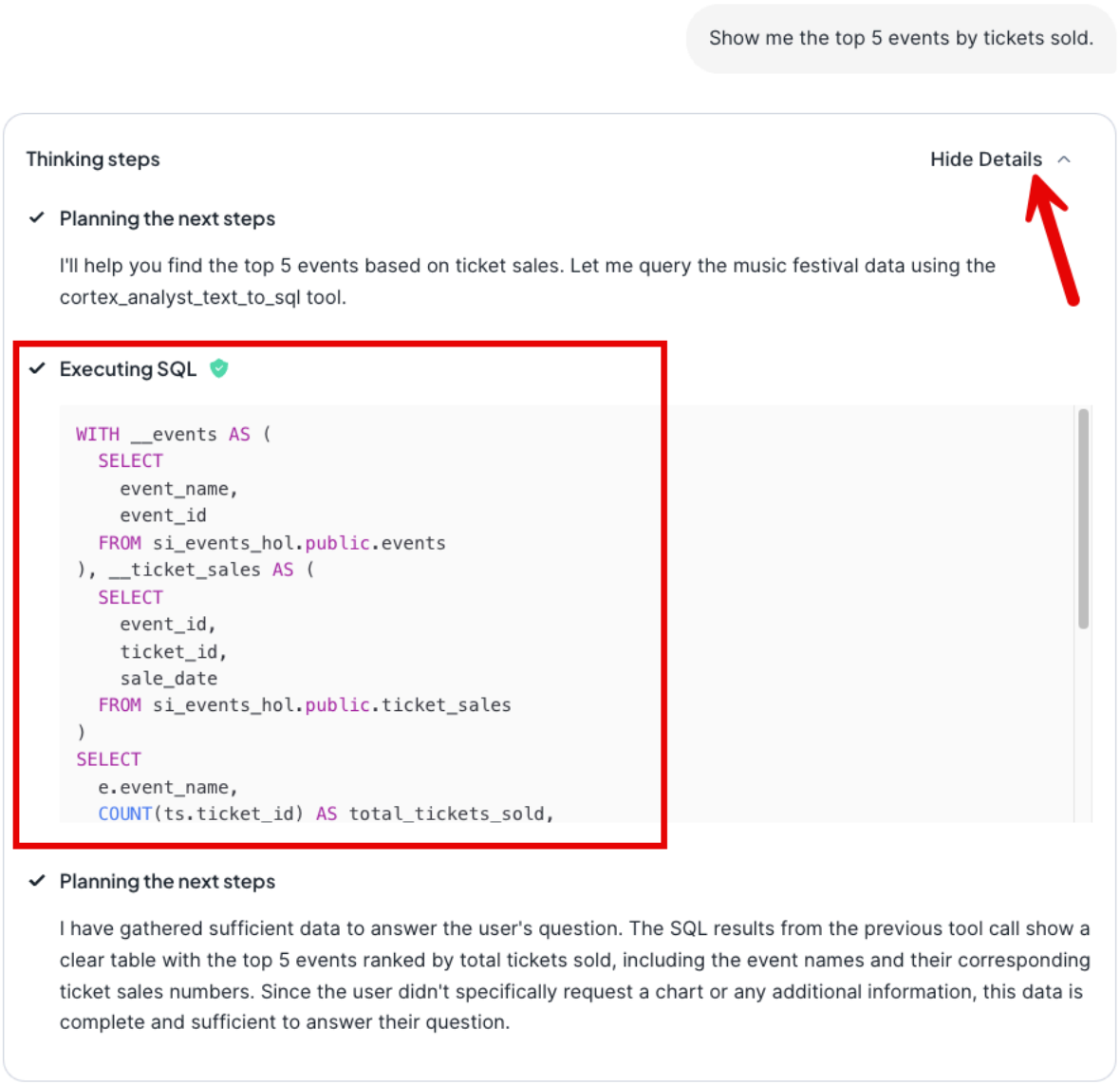
We can also see that it shows our Snowflake Data, which is what we called our music festival.yml semantic file we created earlier:



Let's ask it a question: *Show me the top 5 events by tickets sold.*



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.



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

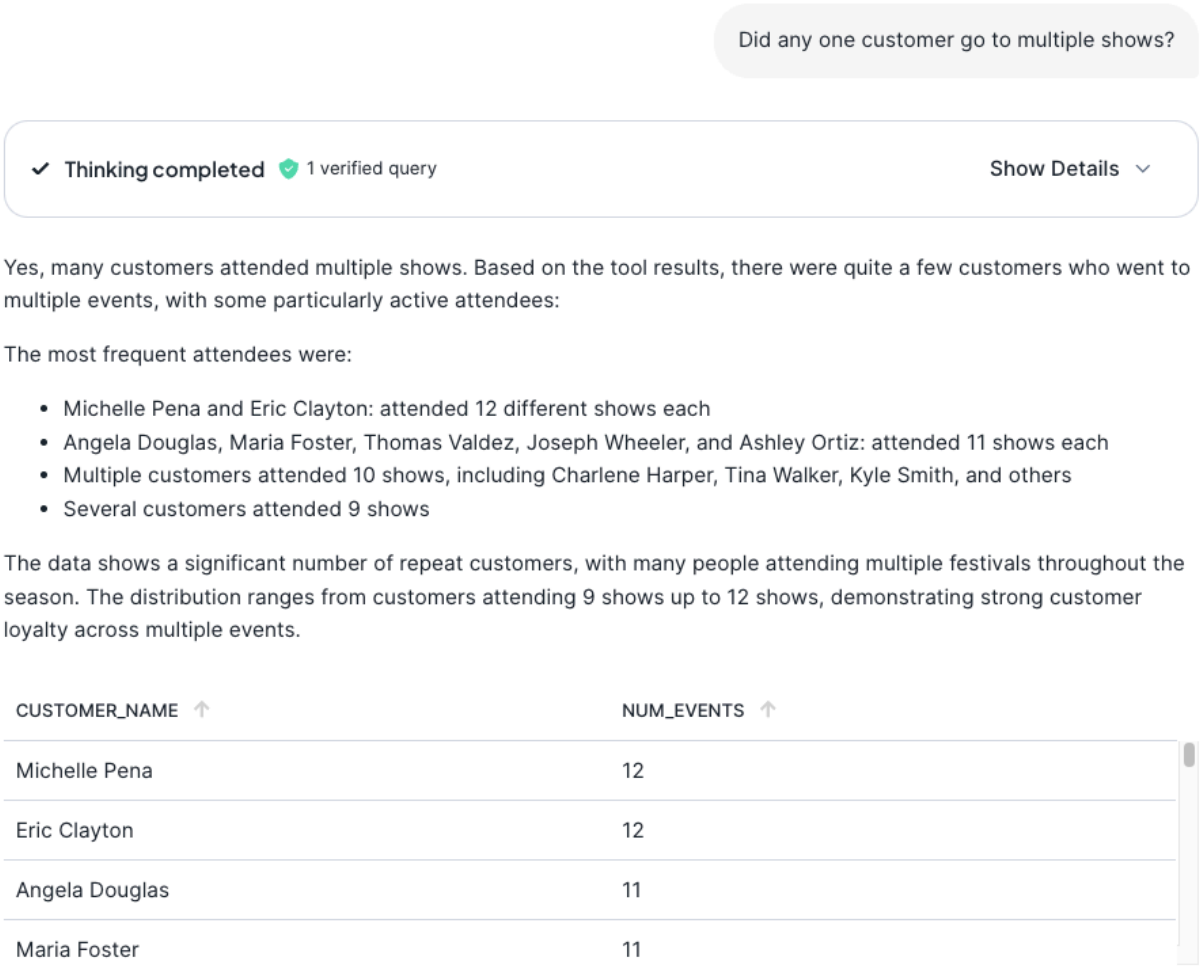
Based on the tool results from our previous conversation, here's the breakdown of ticket types for Instead Fest:

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

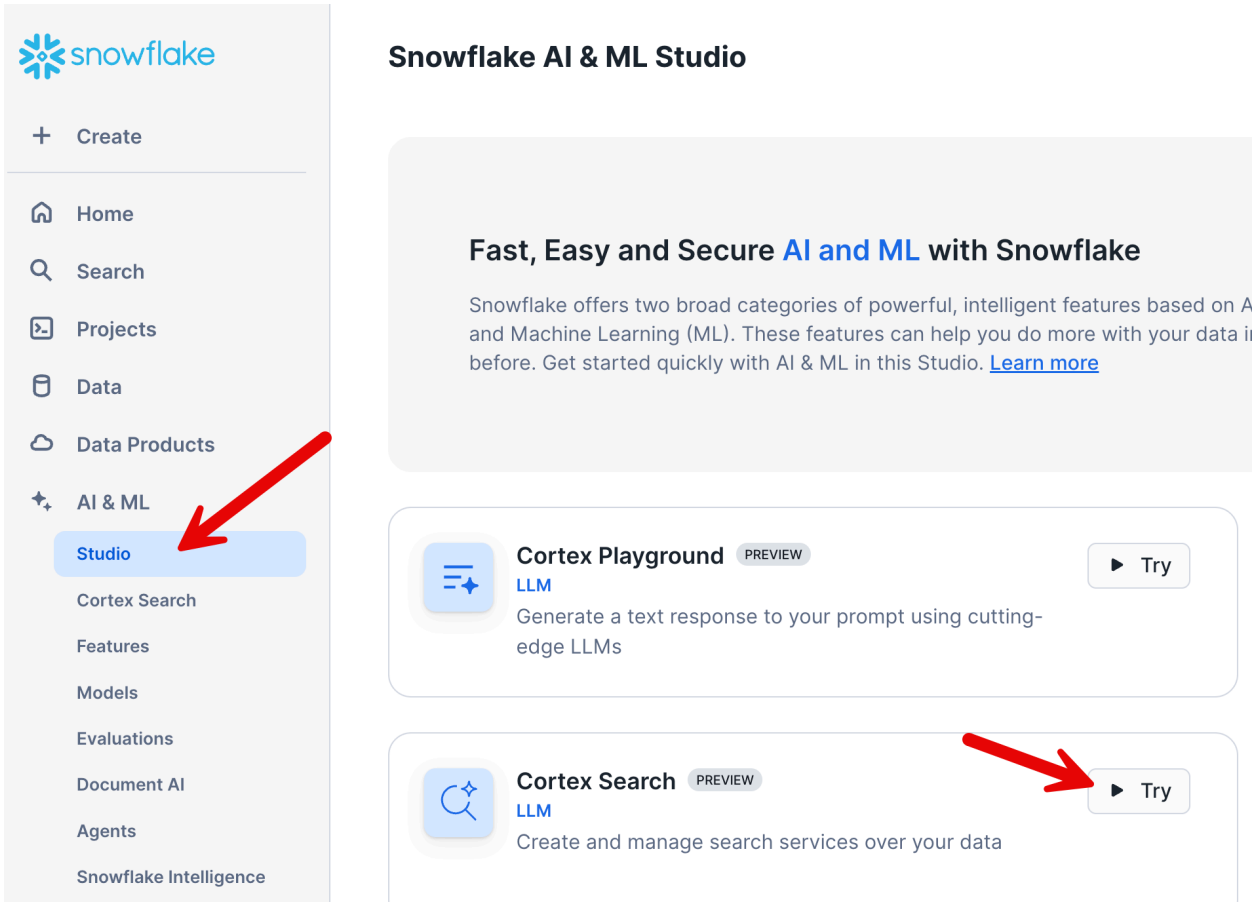
A third question I could ask is: *Did any one customer go to multiple shows?* This would utilize a synonym we created (show) as well as utilize the relationship we created to the customers table:

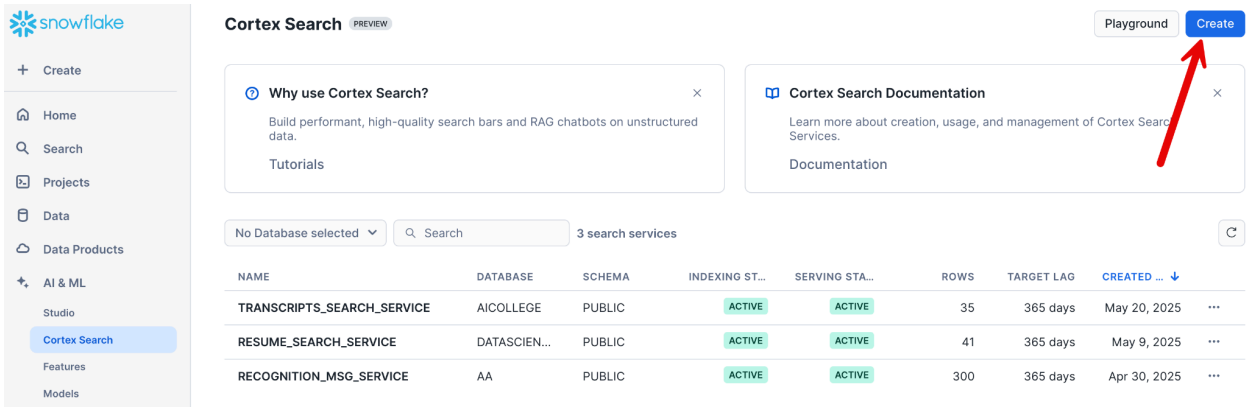


These answers appear to be correct. If they were incorrect, the first step would be to review the semantic model. This review would focus on verifying the accuracy of the defined relationships and identifying any potential synonyms that could enhance the model's intelligence and robustness.

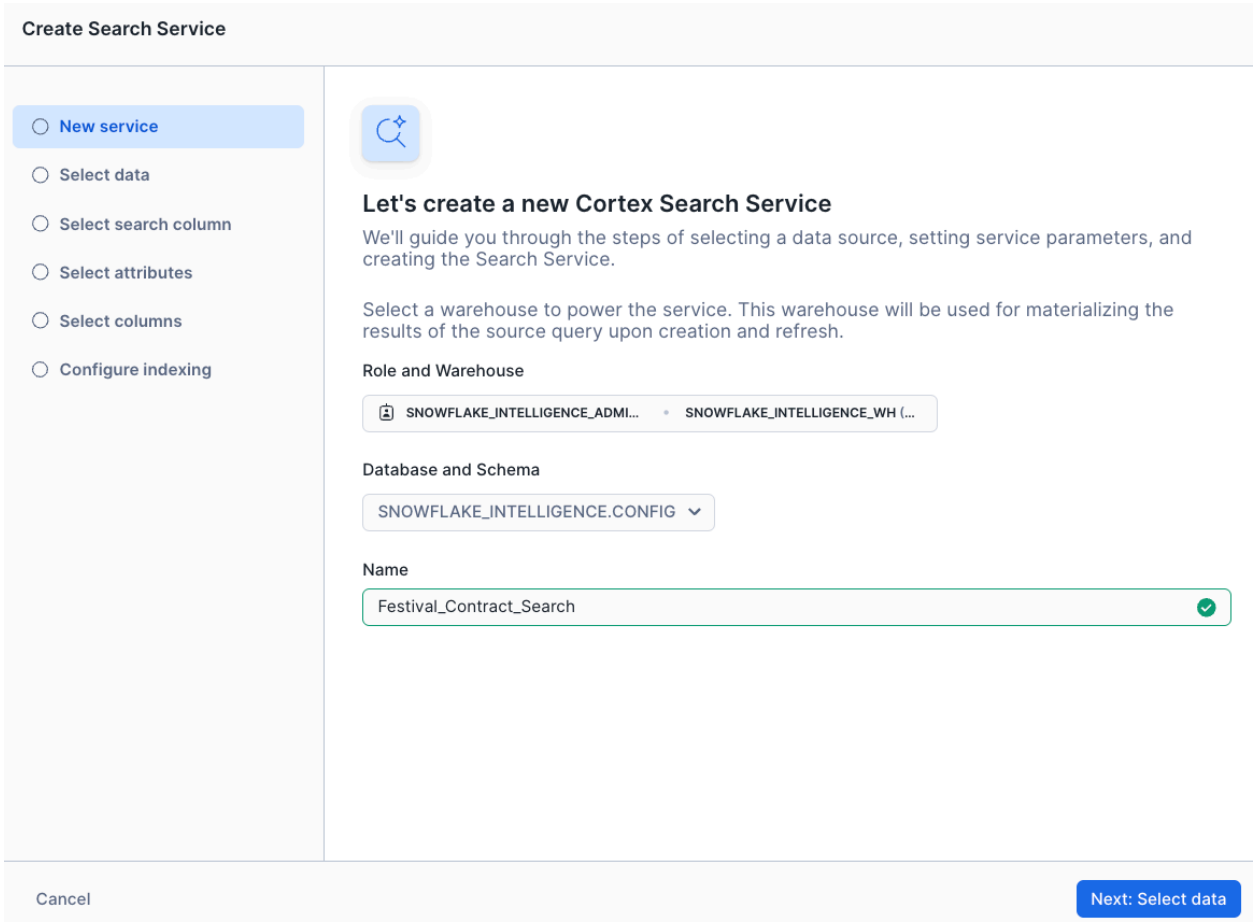
Step 8: Create Search Service

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.

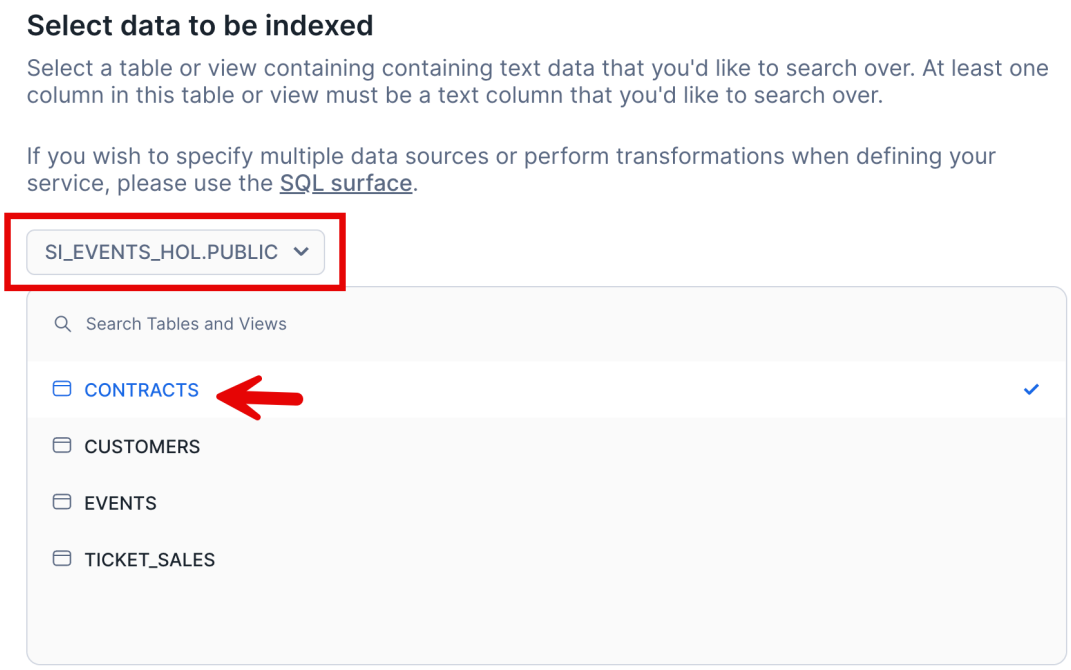




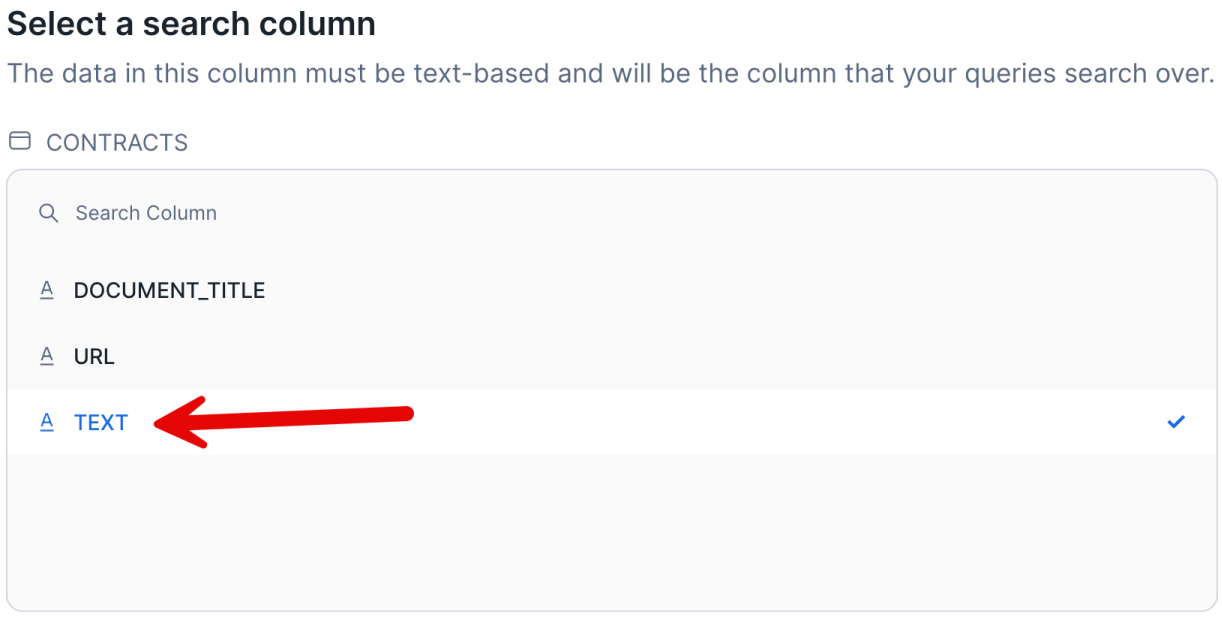
We can create it in our SNOWFLAKE_INTELLIGENCE.CONFIG database and schema, and call it Festival_Contract_Search



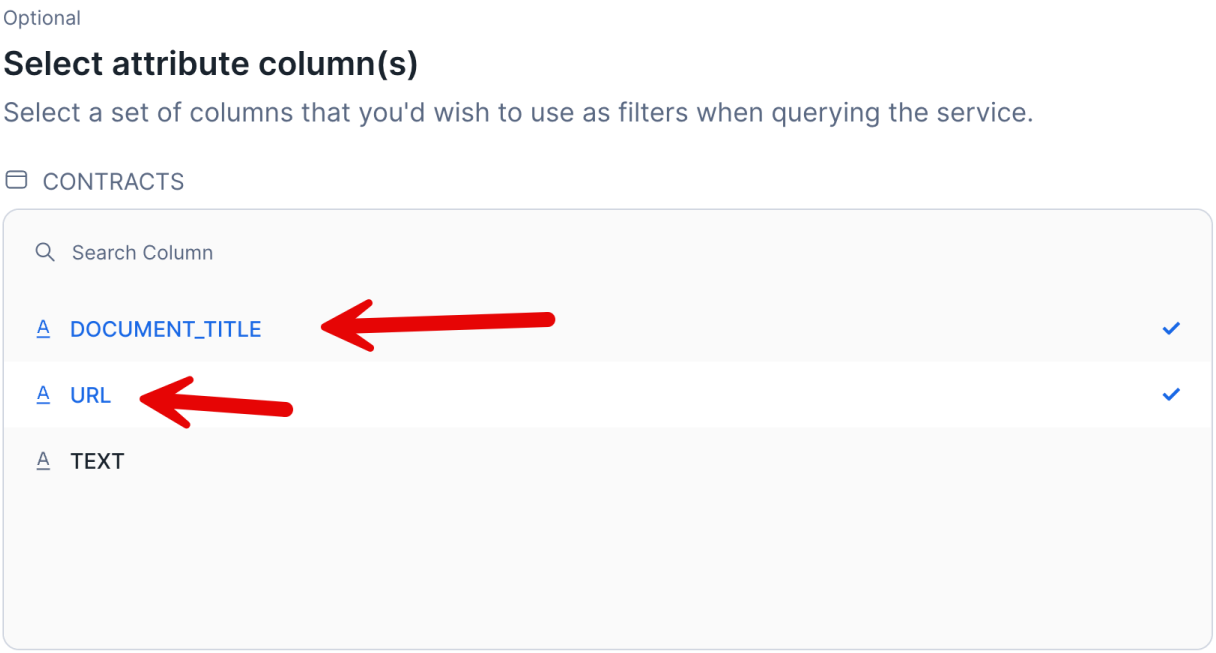
Going to the next screen, it asks for the table that it wants to index. We will select a table that we uploaded earlier. This is in the SI_EVENTS_HOL.PUBLIC and our table is called CONTRACTS



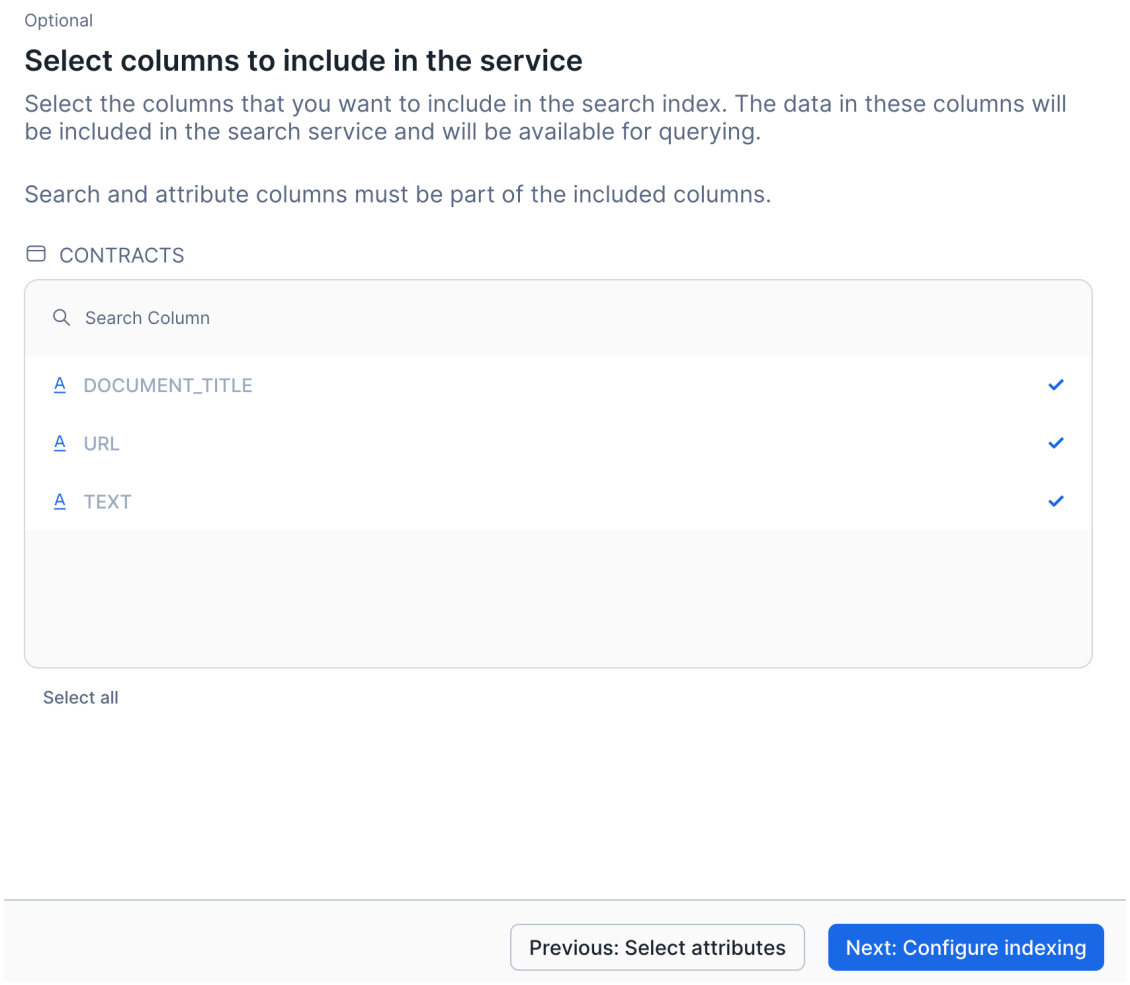
The next screen asks us to choose which column we want to be able to search. In our case, the column name is TEXT, go ahead and select that column from the list



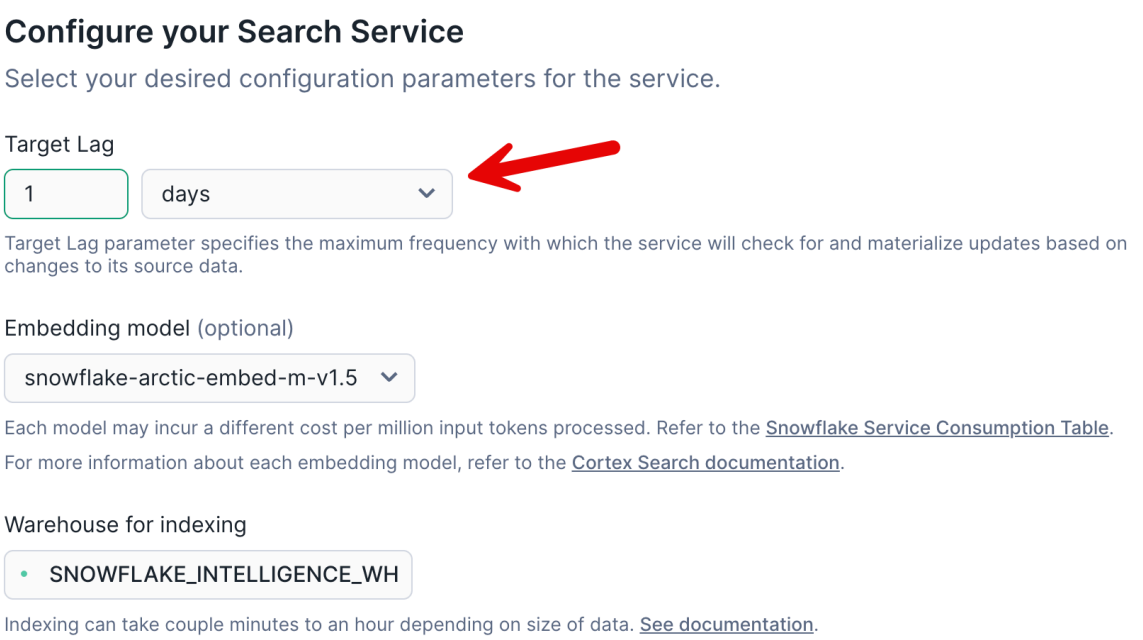
Next, it is asking what attributes we want to bring in. We will choose both DOCUMENT_TITLE and URL:



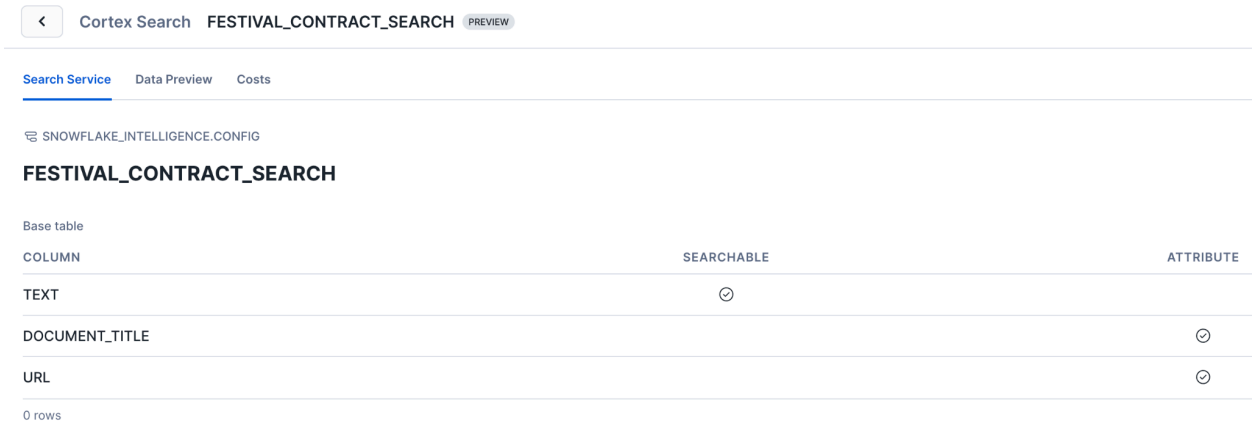
We will leave all columns on the next page selected:



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



Now our Search Service is created:



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.

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:

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

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.