**1. Overview**

Through this quickstart guide, we will explore how to get started with [Cortex Analyst](https://docs.snowflake.com/user-guide/snowflake-cortex/cortex-analyst), which is a fully managed service in Snowflake that provides a conversational interface to interact with structured data in Snowflake.

**What is Cortex Analyst?**

Cortex Analyst is a fully managed service in [Cortex AI](https://www.snowflake.com/en/data-cloud/cortex) that provides a conversational interface to interact with structured data in Snowflake. It streamlines the development of intuitive, self-service analytics applications for business users, while providing industry-leading accuracy. To deliver high text-to-SQL accuracy, Cortex Analyst uses an agentic AI setup powered by state-of-the-art LLMs. Available as a convenient REST API, Cortex Analyst can seamlessly integrate into any application. This empowers developers to customize how and where business users interact with results, while still benefiting from Snowflake's integrated security and governance features, including role-based access controls (RBAC), to protect valuable data.

**Why use Cortex Analyst?**

Historically, business users have primarily relied on BI dashboards and reports to answer their data questions. However, these resources often lack the flexibility needed, leaving users dependent on overburdened data analysts for updates or answers, which can take days. Cortex Analyst disrupts this cycle by providing a natural language interface with high text-to-SQL accuracy. With Cortex Analyst organizations can streamline the development of intuitive, conversational applications that can enable business users to ask questions using natural language and receive more accurate answers in near real time

This quickstart will focus on getting started with Cortex Analyst, teaching the mechanics of how to interact with the Cortex Analyst service and how to define the Semantic Model definitions that enhance the precision of results from this conversational interface over your Snowflake data.

What you will learn

* How to construct and configure a Semantic Model for your data
* How to call the Cortex Analyst REST API to use your Semantic Model to enable natural-language question-asking on top of your structured data in Snowflake via Streamlit in Snowflake (SiS) application
* How to integrate Cortex Analyst with Cortex Search to enhance SQL queries generated
* How to enable Join support for Star Schemas
* How to enable multi-turn conversations

Prerequisites

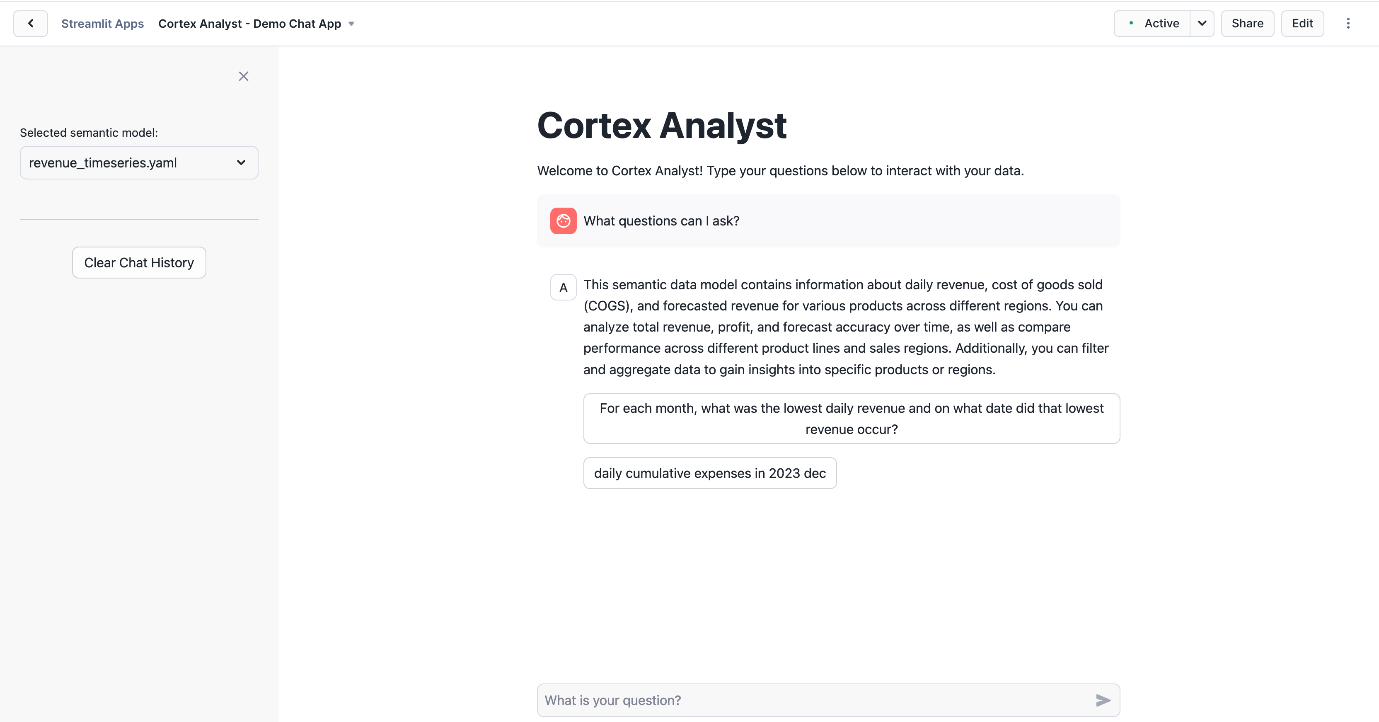
* [Git](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) installed

Download the [git repo](https://github.com/Snowflake-Labs/sfguide-getting-started-with-cortex-analyst" \t "_blank)

* (Optional) [Python >= 3.9, <= 3.11 installed](https://www.python.org/downloads/) to run the OSS Semantic Model Generator
* A [Snowflake account login](https://signup.snowflake.com/) with a role that has the ability to create database, schema, tables, stages, user-defined functions, and stored procedures. If not, you will need to register for a free trial account from any of the supported cloud regions or use a different role.

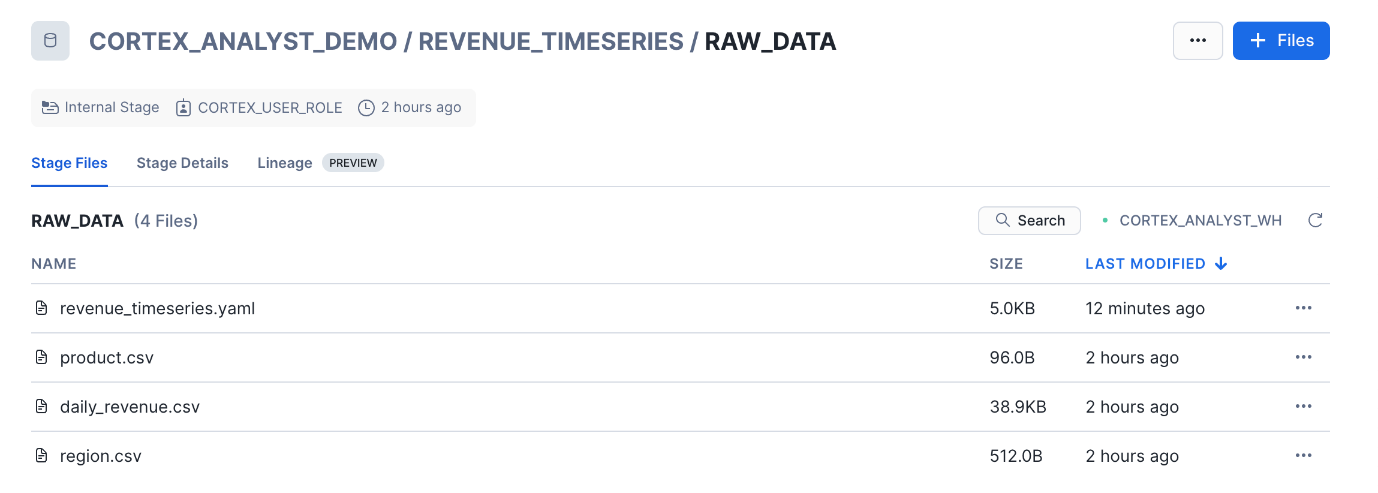
What You'll Build

* A Semantic Model over sample financial data
* A Streamlit in Snowflake (SiS) app with a conversational interface to Cortex Analyst



2. Set up the Snowflake environment

3. Ingest the Revenue Data and Semantic Model YAML

We should see the four files listed in the stage: 

Now, let's load the raw CSV data into the tables.

**4. Integrate Cortex Search**

Now, we will integrate Cortex Search as a way to improve literal string searches to help Cortex Analyst generate more accurate SQL queries. Writing the correct SQL query to answer a question sometimes requires knowing exact literal values to filter on. Since those values can't always be extracted directly from the question, a search of some kind may be needed.

**USE DATABASE cortex\_analyst\_demo;**

**USE SCHEMA revenue\_timeseries;**

**use ROLE cortex\_user\_role;**

**CREATE OR REPLACE CORTEX SEARCH SERVICE product\_line\_search\_service**

**ON product\_dimension**

**WAREHOUSE = cortex\_analyst\_wh**

**TARGET\_LAG = '1 hour'**

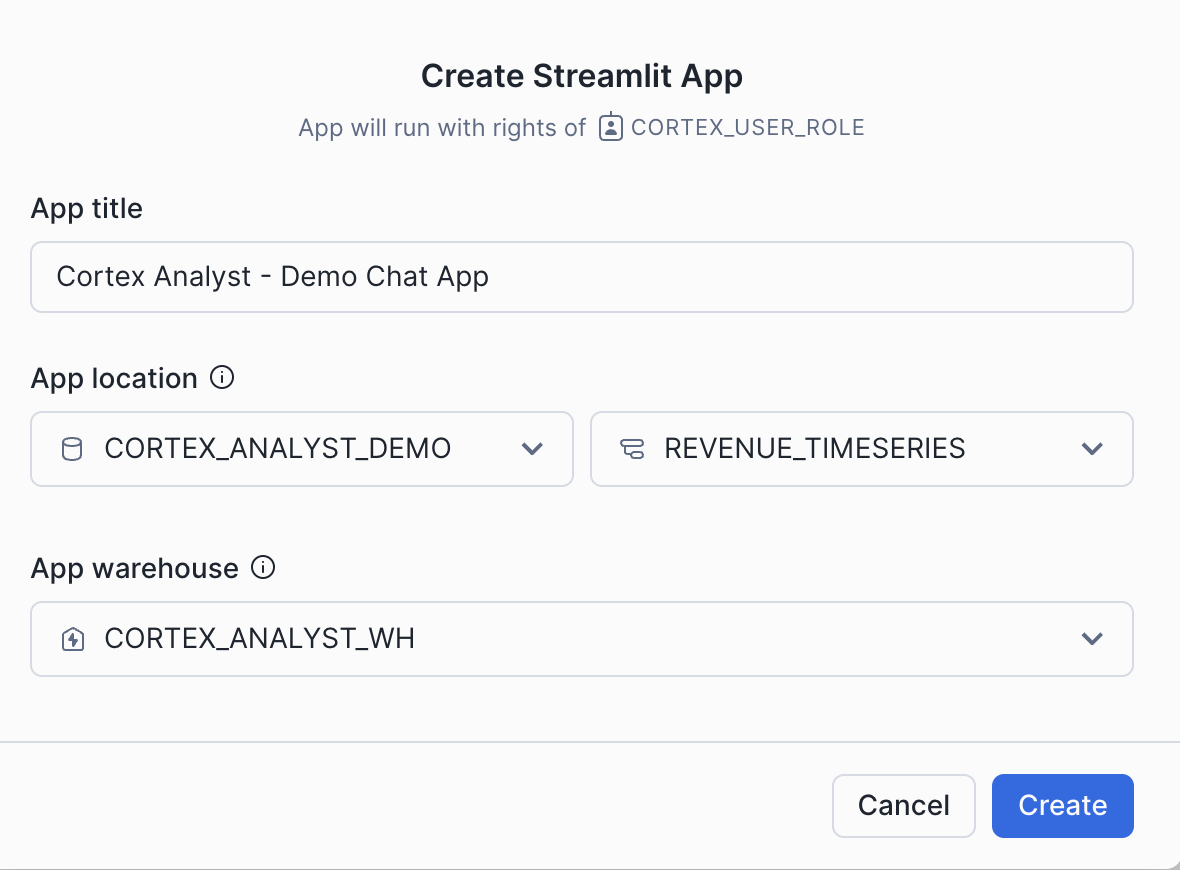
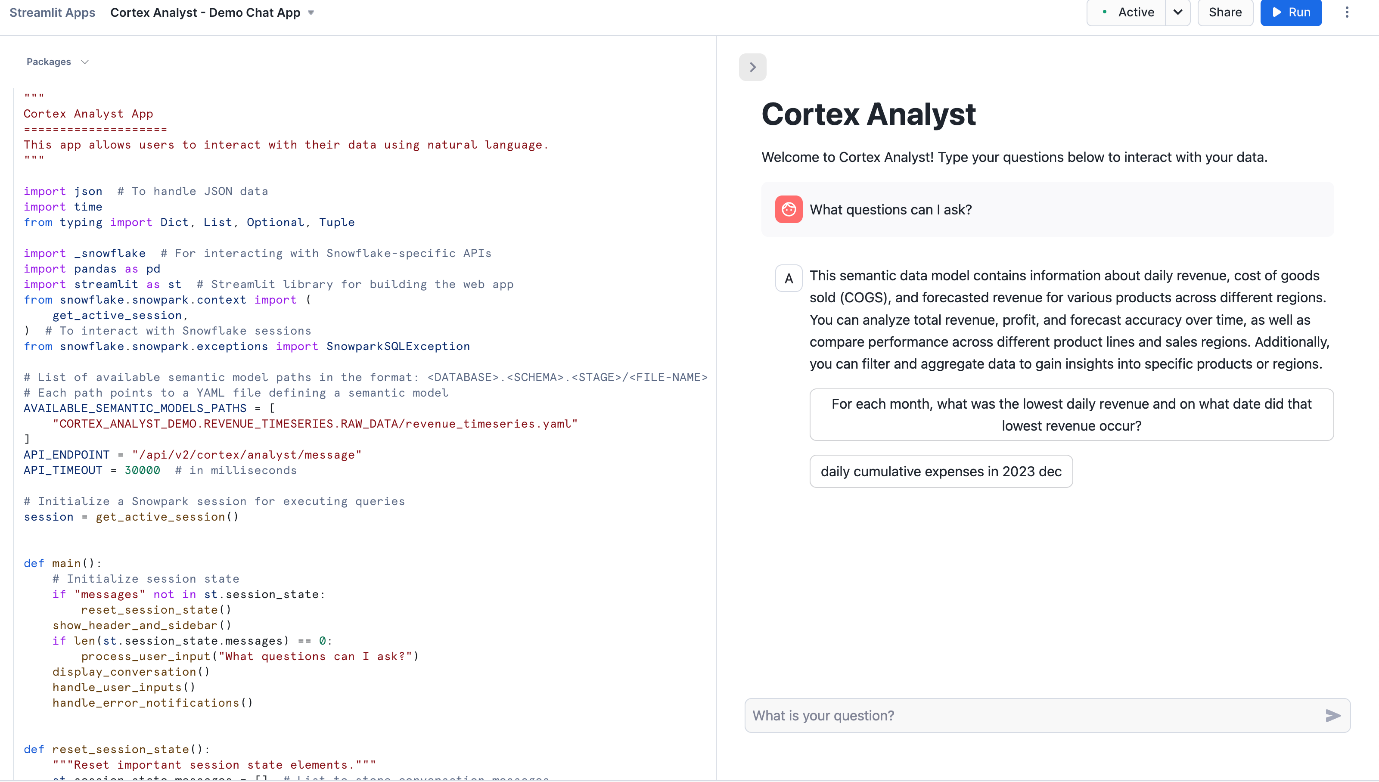
**AS (**

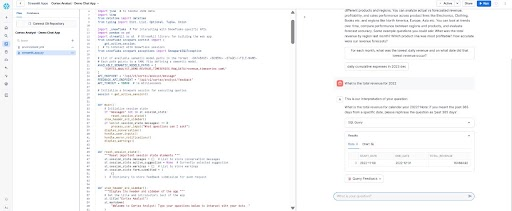
**SELECT DISTINCT product\_line AS product\_dimension FROM product\_dim**

**);**

**5. Create a Streamlit Conversational App**

Now, we will create a demo chat application to call the Cortex Analyst API and ask natural-language questions over our structured revenue datasets. To create the Streamlit in Snowflake application:

* Go to the Streamlit page in Snowsight, and click + Streamlit App, and fill it in with the below details and click create: 
* Open up the .py file in your preferred local code editor
* Copy and paste the SiS app code into the Streamlit editor 
* Click Run and begin asking questions!



Take note of the **get\_analyst\_response** function that is defined in this Python code. This is the function that takes our chat input prompt and history, packages it up as a JSON object, and sends it to the **Cortex Analyst API (with the specified revenue\_timeseries.yaml Semantic Model).**

A screenshot of a computer program

AI-generated content may be incorrect.

You can now begin asking natural language questions about the revenue data in the chat interface (e.g. "What questions can I ask?")

**6. Semantic Model Details**

The semantic model file [revenue\_timeseries.yaml](https://github.com/Snowflake-Labs/sfguide-getting-started-with-cortex-analyst/blob/main/revenue_timeseries.yaml" \t "_blank) is the key that unlocks Cortex Analyst's power. This YAML file dictates the tables, columns, etc. that Analyst can use in order to run queries that answer natural-language questions Let's talk a little about the details of this file:

The [Semantic Model](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/semantic-model-spec) is composed of a number of different fields that help Cortex Analyst understand the specifics of your data:

* Logical Tables which are composed of Logical Columns
* Logical Columns which are one of dimensions, time\_dimensions, or measures
* Relationships that exist between tables to allow for JOINS

Logical Tables are relatively straightforward- these are tables or views within a database. That's it! Pretty simple

Logical Columns get a bit more complicated; a logical column can reference an underlying physical column in a table, or it can be a expression containing one or more physical columns. So, for example, in the [revenue\_timeseries.yaml](https://github.com/Snowflake-Labs/sfguide-getting-started-with-cortex-analyst/blob/main/revenue_timeseries.yaml" \t "_blank), we have a simple logical column daily\_revenue that is a physical column. In the daily\_revenue measure definition, you'll notice that we provide a description, as well as synonyms, data\_type, and a default\_aggregation, but no expr parameter. This is because revenue is simply a physical column in the daily\_revenue table:

**measures:**

**- name: daily\_revenue**

**expr: revenue**

**description: total revenue for the given day**

**synonyms: ["sales", "income"]**

**default\_aggregation: sum**

**data\_type: number**

In contrast, we define a different measure daily\_profit which is not in fact a physical column, but rather an expression of the difference between the revenue and cogs physical columns:

**- name: daily\_profit**

**description: profit is the difference between revenue and expenses.**

**expr: revenue - cogs**

**data\_type: number**

In the semantic model, time\_dimensions specifically capture temporal features of the data, and dimensions are not quantitative fields (e.g. quantitative fields are measures, while categorical fields are dimensions).

An example **time\_dimension:**

**time\_dimensions:**

**- name: date**

**expr: date**

**description: date with measures of revenue, COGS, and forecasted revenue for each product line**

**unique: false**

**data\_type: date**

An example **dimension**:

**dimensions:**

**- name: product\_line**

**expr: product\_line**

**description: product line associated with it's own slice of revenue**

**unique: false**

**data\_type: varchar**

**sample\_values:**

**- Electronics**

**- Clothing**

**- Home Appliances**

**- Toys**

**- Books**

An example **relationship**:

**relationships**:

**- name: revenue\_to\_product**

**left\_table: daily\_revenue**

**right\_table: product**

**relationship\_columns:**

**- left\_column: product\_id**

**right\_column: product\_id**

**join\_type: left\_outer**

**relationship\_type: many\_to\_one**

Here are some tips on building your own semantic model to use with Cortex Analyst:

When generating the semantic model, think from the end user perspective:

* For business user, accuracy and trust is the paramount
* Organize your YAML file in the unit of business domain/topic
* If you are trying to pull a snippet of this data into excel for your business stakeholder, what are the tabs and columns you'd keep? What are the column namings you'd use?
* Use above to guide your selection of tables and columns. Err on the side of only including necessary columns.
* We recommend not exceeding 3-5 tables, 10-20 columns each table to start.

Some additional items that'll significantly improve model performance:

* Capture more difficult or more business-specific queries into expressions and verified queries (please use the Chat tab within streamlit admin app to add verified queries)
  + Verified queries will be provided as few-shot example for model to draw inspiration from, hence significantly improve performance
  + If any organization specific logic cannot be captured via other spec items, we recommend you to add to verified queries.
* Start with a simple and small scope, gradually expanding. YAML building is an iterative process.

For more information about the semantic model, please refer to the [documentation](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/semantic-model-spec).

**7. Using the Verified Query Repository (VQR)**

In addition to the previously discussed Semantic Model information, the [Cortex Analyst Verified Query Repository (VQR)](https://docs.snowflake.com/user-guide/snowflake-cortex/cortex-analyst/verified-query-repository) can help improve accuracy and trustworthiness of results by providing a collection of questions and corresponding SQL queries to answer them. Cortex Analyst will then use these verified queries when answering similar types of questions in the future.

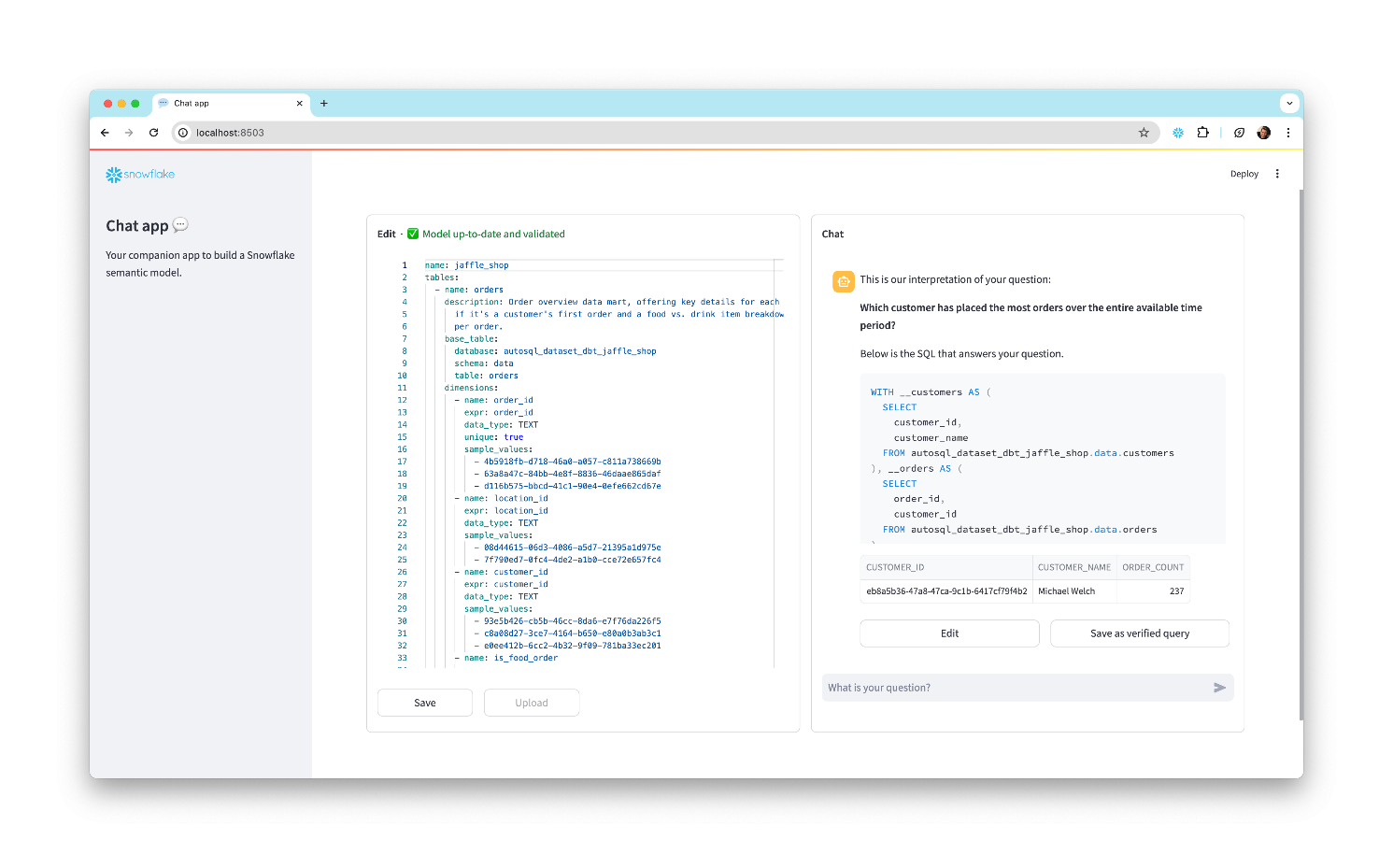
Adding Verified Queries

Verified queries ultimately are specified in the verified\_queries section of the semantic model, e.g.:

A computer screen shot of a computer program

AI-generated content may be incorrect.

While verified queries can be added directly to the Semantic Model, Snowflake also provides an OSS Streamlit application to help add verified queries to your model.



To install and use this app:

1. Clone the [semantic-model-generator repository](https://github.com/Snowflake-Labs/semantic-model-generator). Follow the setup instructions in the [repo's README](https://github.com/Snowflake-Labs/semantic-model-generator/blob/main/README.md) to provide your credentials. Then follow the instructions in the [admin\_app README](https://github.com/Snowflake-Labs/semantic-model-generator/blob/main/admin_apps/README.md" \t "_blank) to install dependencies and start the app.
2. Once the app is running, enter the database, schema, and stage location of your semantic model YAML file in the empty fields. The YAML appears in an interactive editor on the left side of the window.
3. On the right side of the window, ask a question in the chat interface to generate a SQL query.
4. Inspect the generated query and the results it produces. If it worked as you expected, select the **Save as verified query** button below the assistant answer to add the query to your semantic model.
5. If generated query is incorrect, select the **Edit** button and modify the query, then run the modified query and see if it produces the intended results. Continue modifying and testing the query until it works the way you want it to. Finally, select **Save as verified query** to save the query to your semantic model.
6. Select the **Save** button in the bottom left of the window to update the semantic model, then go back to step 2 if you want to add more queries.
7. When you're satisfied with the queries you've added, select the **Upload** button and enter a file name for your new YAML file, then select **Submit Upload**.

Modify your SiS application code to point at the new Semantic Model YAML file location, and use Cortex Analyst as before!