# College of AI: Data Agent HOL: Operationalize Cortex Agents for Natural Language Analytics with Slack + Snowflake

## Deliver trusted insights from both structured and unstructured enterprise data—no SQL, no dashboards, no fine-tuning required.

In this hands-on lab, you’ll step into the shoes of a **customer success analyst** responding to live stakeholder questions using meeting transcripts and CRM data—all inside Slack. Instead of writing SQL or wrangling multiple dashboards, you’ll build a **Cortex Agent** that understands natural language, retrieves the right insights using Cortex Search and Cortex Analyst, and delivers clear, auditable answers from Snowflake.

### What You’ll Do:

* Set up a Snowflake **semantic model** and verified queries for Cortex Analyst
* Ingest and index meeting transcripts using **Cortex Search**
* Connect a Slack bot that lets business users ask questions in plain English 💬
* Build a Python-based agent that routes across Cortex tools (Search, Analyst, Complete)
* Run real-world queries across structured and unstructured data 🔍
* Deliver trusted, explainable answers directly in Slack with the SQL attached 📑

### Why this matters:

* **Natural-language access** to governed data: Enable text-to-SQL using semantic models
* **Hybrid retrieval** in one step: Cortex Agents blend structured + unstructured sources automatically
* **No-code governance**: Define joins, filters, synonyms, business logic in YAML or the UI
* **Slack integration**: Meet users where they work, with in-thread answers and SQL transparency
* **Agent Orchestration**: reasoning + dynamic tool selection returns the right answer, not just any answer💡

### Lab Overview Video:

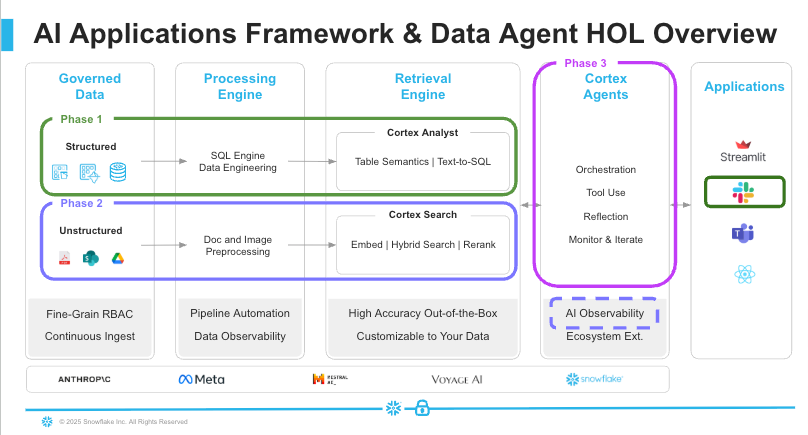
Watch this [5‑minute video](https://drive.google.com/file/d/1TpGMciju3WnL_XNX4S2gU8V8SHr3xSCu/view?usp=sharing) for a quick walkthrough of the 3-phase Data Agent HOL workflow.

### Use these discovery questions with customers:

* Do business users still rely on data teams to write SQL for common questions?
* How are unstructured documents (transcripts, notes, emails) analyzed today?
* Would Slack-native, data-backed answers help your teams move faster?
* Do you currently join structured CRM data with support tickets or meeting transcripts?
* Are dashboards struggling to keep up with ad-hoc, exploratory questions?
* Could agents that bridge structured and unstructured data reduce time-to-answer?

## **🧪 Lab Overview: Data Agents with Cortex**

This lab walks through a 3-phase agent orchestration flow that integrates Snowflake Cortex tools into a Slack-based assistant.



### **Phase 1 – Text-to-SQL via Cortex Analyst**

* Load customer + meeting datasets into Snowflake
* Define semantic model with synonyms, filters, metrics, joins
* Set up Slack integration for /askcortex natural-language querying
* Run verified queries and see SQL + answers in Slack

### **Phase 2 – Add Unstructured Retrieval with Cortex Search**

* Upload transcripts (PDFs) to a Snowflake stage
* Parse and chunk content using PARSE\_DOCUMENT + SPLIT\_TEXT\_RECURSIVE\_CHARACTER
* Analyze with Cortex Complete: extract sentiment, risk, next steps, key phrases
* Add transcript facts to the semantic model + verify queries across both data types
* Enable **Dynamic Literal Retrieval** for smarter question matching

### **Phase 3 – Full Cortex Agent Orchestration**

* Build a Python-based agent (slackapp\_agent.py) that:
  + Selects the right tool (Search, Analyst, or both)
  + Generates SQL, runs it, and summarizes results
  + Handles ambiguity and open-ended questions
* Deploy the Slack-integrated agent

Helpful Docs:

* [Cortex Analyst REST API Documentation](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/rest-api)
* [Cortex Agents Documentation](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents)
* [Snowflake Cortex Agents Tutorials](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents-tutorials)
* [Create Search Services in Snowflake](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-search/cortex-search-overview)
* [Complete Structured Output Docs](https://docs.snowflake.com/en/user-guide/snowflake-cortex/complete-structured-outputs)

👋 **Heads-up College of AI SEs**: *This* Data Agent HOL is the **only** agentic lab required for your Q2 certification. It contains an optional **Snowflake Intelligence add-on** ([Step 3.4](#_9yak03464el1)) for anyone who wants to try the new point-and-click UI. All other SE College tracks will require completion of a standalone [**Snowflake Intelligence HOL**](https://snowflake.seismic.com/Link/Content/DCCWPh9MV3jBgGfP8bm29qqgmgcB).

📌 Optional deep-dive – see how much effort a DIY stack takes 🎓

* [DataCamp - Designing Agentic Systems with LangChain](https://app.datacamp.com/learn/courses/designing-agentic-systems-with-langchain) (3 hrs, intermediate). Build an open-source LangChain agent from scratch—prompts, LLM wiring, tool wrappers, error handling—and you’ll quickly appreciate how Snowflake’s turnkey Cortex Agents eliminate that complexity.

### HOL Instructions Legend:

🛠️ **Actions**: hands-on steps to follow

📌 **Note** - tips and gotchas

🔹 **Info** - important background information or context

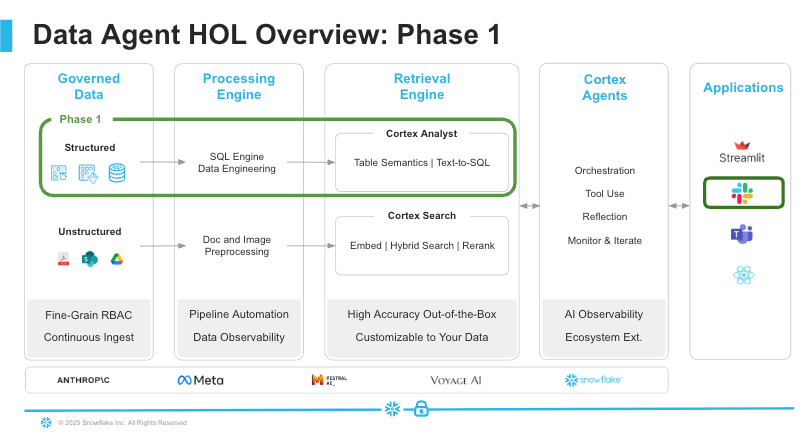
## **Phase 1: Cortex Analyst & Slack Integration**

We want teammates to ask questions in **Slack** and get trusted answers straight from **Snowflake**—no SQL, no guessing joins. Phase 1 sets that up in three quick moves:

| **Step** | **What you do** | **Why it pays off** |
| --- | --- | --- |
| **1. Load demo data** | Create four small tables (**MEETINGS, OUTCOMES, INTERACTIONS, INDUSTRY**) and a **CUSTOMER\_INSIGHTS** view that ties them together. | Gives Cortex a single, tidy table of customers, features, and deal metrics—ready for analytics. |
| **2. Add a semantic model** | Upload a YAML file with column **synonyms**, table **relationships**, fiscal-year logic, and a few **verified “golden” queries**. | Cortex Analyst now understands “demo POC,” “Gen AI feature,” or “this fiscal quarter” without ambiguity. |
| **3. Connect Slack** | Run the provided Slack bot: **PAT** for Cortex REST, **key-pair JWT** for Snowflake queries. | Anyone can type /askcortex … and instantly see answers (plus the generated SQL) inside Slack. |

Completing Phase 1 gives you a secure, auditable **text-to-insight** pipeline—business users stay in Slack, data remains in Snowflake, and every answer links back to SQL. Example prompts, **ready out-of-the-box**:

* “Which customers asked for a Gen AI POC?”
* “What features are customers most curious about?”
* “Who requested a demo and for which feature?”



To start, you’ll provision a **dedicated Snowflake service user**, **role**, **database**, and **warehouse**, then configure both **Programmatic Access Tokens (PATs)** for calling Cortex Analyst and **key-pair authentication** for your Python agent.

**📌 Why two credentials?**

* [**PAT (Programmatic Access Token)**](https://docs.snowflake.com/en/user-guide/programmatic-access-tokens) **is GA, but its usage supports REST API-based workflows** such as calling Cortex services (e.g., Analyst, Complete, Agents, Document AI) and external integrations like Slack bots or third-party applications using Snowflake’s REST APIs.
* [**Key-pair (JWT) authentication**](https://docs.snowflake.com/user-guide/key-pair-auth#configuring-key-pair-authentication) for Snowpark and Python-based programmatic access to Snowflake.

### **🛠️ Step 1.1: Local Project Setup**

**🚧 Quick reminder for the Data Agent HOL:**

• **Don’t run scripts in macOS’s default Terminal.**

• Open your DATAAGENT folder in **VS Code** **and use the integrated terminal**.

• This avoids .venv activation, permissions, or missing-dependency errors.

* **Create a working folder**
  + In your terminal (or Finder/File Explorer), make a new directory (Folder) called DATAAGENT
* **Download** all the files in this [GDrive](https://drive.google.com/drive/folders/1Gt3qj7vbcRb_xM2fEAeTsl8kR3qu4rz0?usp=drive_link):
  + Download *all* of the provided files (including dataagent.env, the semantic‐model YAML, PDFs, etc.) into ~/DATAAGENT/.
  + This will become your project root, and where you’ll also store your Snowflake private key.
* **Generate & Upload Key-Pair for Python Access**
  + **Open VS Code\***
  + In VS Code’s Menu Bar, **select File > Open Folder, and select** your **DATAAGENT** folder
  + In VS Code’s Menu Bar, **select** **Terminal > New Terminal**
  + In your newly created VS Code terminal window, **run the following commands** to generate a private and public key:

openssl genrsa -out rsa\_private\_key.pem 2048

openssl rsa -in rsa\_private\_key.pem -pubout -out rsa\_public\_key.pem

* + Validate that your **rsa\_private\_key.pem** is in your ~/DATAAGENT/. path

###### 📌 \*Why VS Code?

* **Environment picker** in the bottom bar lets you ensure you’re using ./.venv
* Integrated terminal for all your commands
* Easy editing of .env, .py, and immediate rerun

### **Step 1.2: Snowflake Setup**

This step provisions your environment so the Data Agent can query structured meeting data using Cortex Analyst. You will:

* Create a **dedicated role and service user**
* Configure the **Snowflake database, warehouse, and stage**
* Set up required **access privileges and integrations**
* Enable **Cortex cross-region inference**
* Generate a **PAT (Programmatic Access Token)**
* Configure a **key-pair for Python authentication**
* Load structured **customer meeting data**
* Create a semantic **insights view** for Cortex queries

#### **🛠️ Configure Snowflake Environment**

* In Snowflake, use **ACCOUNTADMIN** role
* Navigate to **Snowflake Notebooks**
* Select the "**v**” beside the “**+ Notebook**" button
* Select **Import .ipynb file** option
* Import the [**Data Agent Cortex Agent Setup Notebook**](https://drive.google.com/file/d/1Qso0th4-lZIpsPH19gFlLSpJ35GCgMTs/view?usp=drive_link)
* For **Notebook location**, pick an available DB & schema (ideally AICOLLEGE.PUBLIC if already created)
  + You will create AICOLLEGE.PUBLIC in this notebook.
* Select **Run on warehouse** for the Python environment
* Select an available **X-Small warehouse**
* Select **Create**
* *No packages need to be installed*
* **Run each cell in order**, following the prompts to:
  + Replace <YOUR\_USERNAME> with your **Snowflake username**
  + Copy and securely store the **PAT token** when it appears (it cannot be retrieved later)
  + Open your rsa\_public\_key.pem file in VS Code, and **paste the key contents** into the indicated cell:
    - Include -----BEGIN PUBLIC KEY----- and -----END PUBLIC KEY----- as separate lines
    - Your actual public key must be **a single contiguous line** (no spaces or line breaks)
* **Verify completion** by running the last cell to query the CUSTOMER\_INSIGHTS view.

### **Step 1.3: Snowflake Cortex Analyst Semantic Model Setup**

##### **🔹 Why a Semantic Model?**

Without one, Cortex Analyst must reverse-engineer your schema on the fly—struggling with complex joins, business jargon, and context. A semantic model gives:

* **Business-Friendly Vocabulary** Human-friendly names, descriptions, and synonyms (e.g., “customer\_name” instead of CUST\_NM). Cortex Analyst maps varied terms—“client,” “account holder,” “prospect”—directly to the right columns.
* **Explicit Relationships** Explicit joins (e.g., MEETING\_ID → CUSTOMER\_MEETING\_OUTCOMES.MEETING\_ID) remove guesswork, so multi-table queries work reliably.
* **Time & Fact Dimensions** Flag date, time, and metric fields. Questions like “How many calls last quarter?” translate into precise SQL using your fiscal calendars and measures.
* **Verified Queries & Onboarding** Embed sample prompts with their trusted SQL. This jump-starts user adoption (e.g. “What is our pipeline value?”) and steers the model toward patterns you’ve approved.

**📌** All of this metadata translates into dramatically higher text-to-SQL accuracy. In Snowflake’s own benchmarks, Cortex Analyst with a semantic model achieves **over 90% real-world BI accuracy**, nearly **2× more accurate** than a single-shot GPT-4 prompt, and **14% more accurate** than other market solutions.

*📌 NEW! As of June 2025, the* [*Cortex Analyst semantic model specification*](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/semantic-model-spec) *now requires that each logical table having a defined primary\_key in the YAML.*

##### **🔹Primary Keys and Relationships**

Primary keys are essential for your semantic model to accurately understand table relationships and generate reliable SQL. Without properly defined primary keys:

* Cortex Analyst may struggle to determine the correct join paths
* Many-to-one and one-to-one relationships can't be properly established
* Query performance may suffer due to suboptimal join strategies

###### **Why Primary Keys Matter**

**Data Integrity:** Primary keys ensure each record is uniquely identifiable, preventing duplicate data and maintaining referential integrity.

**Join Accuracy:** When Cortex Analyst generates SQL with joins, it relies on primary keys to determine the correct join columns and cardinality.

**Performance:** Queries using primary key joins typically execute faster than those using non-indexed columns.

Adding primary keys will significantly improve your semantic model's ability to generate accurate SQL, especially for complex multi-table queries involving customer meetings, outcomes, and interactions.

#### **🛠️ Create Cortex Analyst No-Code Semantic Model**

* In Snowflake Snowsight, switch to your **AICOLLEGE** role
* In the left sidebar, expand **AI & ML** and click **Studio**
* Select **Cortex Analyst**
* Select **Create new model** option

**📌 What’s the difference between Stages** (**Semantic Model Files) and Semantic Views?** Both let you supply the same YAML-based model to Cortex Analyst—your choice is how you manage updates and storage.

###### **Semantic model files (YAML in a stage)**

* **How it works**: Upload a plain-text YAML file (data\_agent\_semantic\_model.yaml) into a Snowflake stage (@AICOLLEGE.PUBLIC.SETUP/…).In your Python code, pass: {"semantic\_model\_file":"@AICOLLEGE.PUBLIC.TRANSCRIPTS/data\_agent\_semantic\_model.yaml",…}. Cortex reads that file at runtime (with built-in caching).
* **Pros**
  + True “no-code” updates: edit your local YAML, re-upload, and you’re live
  + Treated like any other artifact in Git or CI/CD
  + Ideal for rapid iteration during development
* **Cons**
  + You must maintain stage paths and keep files synced
  + Cortex fetches the file on each request (though cached)

###### **Semantic views (native Snowsight object)**

* **How it works**: Create a “Semantic View” in Snowsight (AICOLLEGE.PUBLIC.CORTEX\_MODEL\_VIEW), which stores your YAML in Snowflake metadata tables rather than as a flat file. In Cortex Analyst UI, select the view name, and it pulls model definitions directly from the view’s metadata.
* **Pros**
  + All definitions live inside Snowflake—no external files or stages
  + GUI edits immediately apply to the model
  + Leverage Snowflake’s versioning and RBAC on the view object
* **Cons**
  + Less transparent for code-first workflows
  + Updates require Snowsight UI or DDL, not a simple file replace

###### **📌 Stages vs. Semantic Views: Which should you choose?**

* **Development / Infra-as-Code:** Use stage-based YAML files for Git-centric, fast updates.
* **Production / RBAC-Locked:** Use Semantic Views to centralize definitions inside Snowflake while still scripting via DDL.

Either way, Cortex Analyst gets the same rich business terms, relationships, and verified queries—just maintained in the way that best fits your workflow.

#### **🛠️ Add Logical Tables to your Semantic Model**

* Since this Data Agents HOL is a development effort, choose the **Stages** card
* Create Semantic Model: Getting started page:
  + Location to store: choose **AICOLLEGE.PUBLIC** → **TRANSCRIPTS**
  + Name: **data\_agent\_semantic\_model**
  + Description: **Unified model for analyzing customer meetings, outcomes and interactions for Analyst reasoning.**
  + File name: **data\_agent\_semantic\_model.yaml**
* **Click Next: Select tables**
* Create Semantic Model: Select tables page:
  + Expand **AICOLLEGE › PUBLIC** and check only these five objects:
    - **CUSTOMER\_INDUSTRY**
    - **CUSTOMER\_INTERACTIONS**
    - **CUSTOMER\_MEETINGS**
    - **CUSTOMER\_MEETING\_OUTCOMES**
    - **CUSTOMER\_INSIGHTS**
* **Click Next: Select columns**
* Create Semantic Model: Select columns page:
  + **Select** **all columns** (37 selected) to be included
  + Leave **“Include example data from selected columns to improve its quality”** checked.
  + You can optionally deselect any column you don’t want exposed. Typically, you’ll want all.
* Click **Create and Save**.

##### **🔹 Logical Tables**

A [**logical table**](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/semantic-model-spec) in Cortex Analyst is a **virtual, business-centric schema** that presents one or more source tables or views as a **flat, curated entity**—all without altering your raw data. Users simply query the logical table name, and Cortex Analyst generates the correct SQL.

**Key benefits of logical tables:**

* **Simplified Queries:** Users query the existing CUSTOMER\_INSIGHTS object via Cortex Analyst instead of writing raw SQL against CUSTOMER\_MEETINGS, CUSTOMER\_INTERACTIONS, CUSTOMER\_INDUSTRY, and CUSTOMER\_MEETING\_OUTCOMES.
* **Encapsulated Logic:** Define all JOIN and UNION operations in the model so users never see the SQL plumbing.
* **Easy Maintenance:** Tweak joins or add sources in one place—every downstream query updates automatically.

By modeling your base tables—CUSTOMER\_MEETINGS, CUSTOMER\_INTERACTIONS, CUSTOMER\_INDUSTRY, CUSTOMER\_MEETING\_OUTCOMES—and CUSTOMER\_INSIGHTS view **as logical tables**, **you create a single**, **11-column denormalized schema**. This unified dataset lets Cortex Analyst:

* **Generate accurate SQL** every time, thanks to predefined joins and relationships
* **Speed up performance** by avoiding ad-hoc join logic
* **Simplify natural-language queries** so business users never have to write SQL

In Phase 1, these logical tables turn your raw data and view into a **curated, business-focused foundation for seamless, reliable analytics**.

📌 When you see **“5 of 5 logical tables missing synonyms”**, Cortex Studio is warning that none of your logical tables have **alternate names** defined. **Synonyms** let users refer to the same data in many ways—adding them makes Cortex Analyst understand natural-language queries without guessing.

###### **🔹 Why Synonyms Matter?**

* **Bridging Vocabulary Gaps:** Map varied terms like **“client data,” “account info,”** or **“customer overview”** to your logical table.
* **Boosting Accuracy:** Models with rich synonyms achieve a **20–30% lift** in correct SQL generation versus sparse models.
* **Smoother Onboarding:** Early adopters use familiar terms (client\_insights, customer\_summary, or sales\_overview as synonyms), reducing friction caused by “it didn’t understand”.

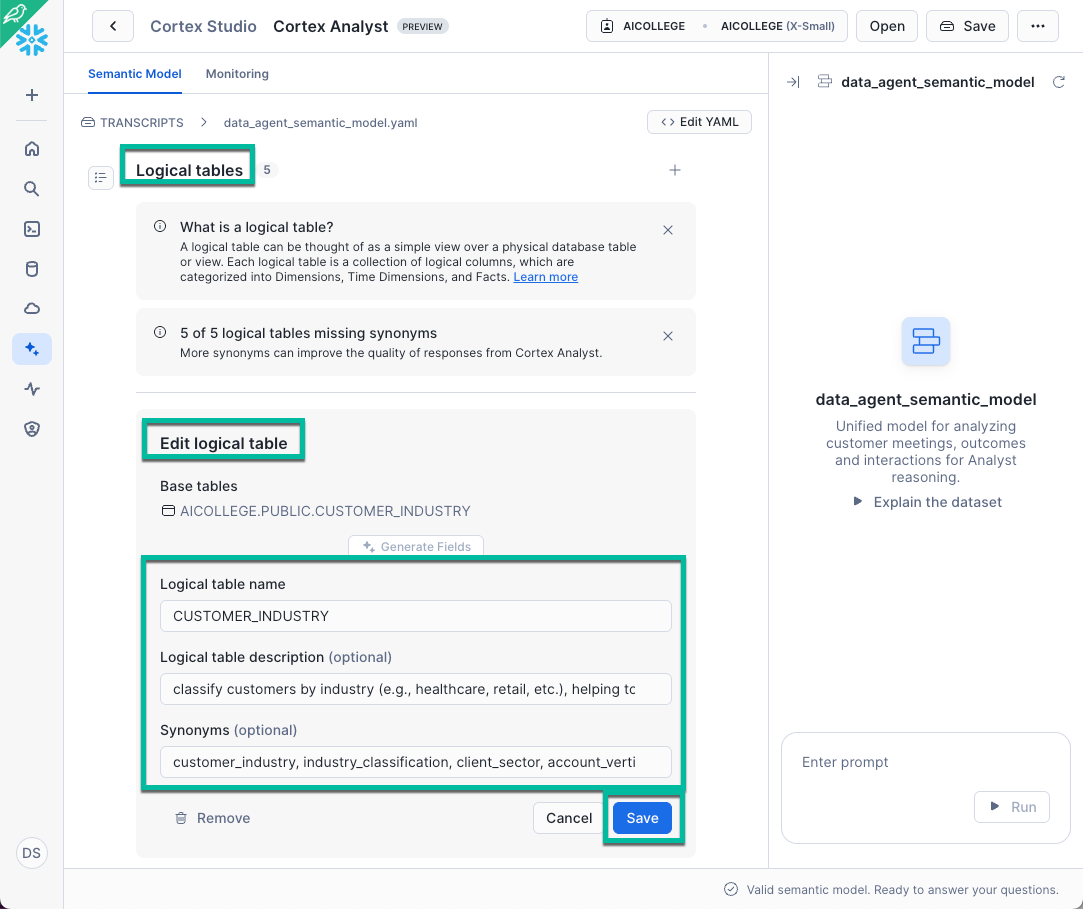
###### **How to Add Table-Level Synonyms**

**Synonyms** help Cortex Analyst **interpret user queries more effectively**. You can add synonyms to existing logical tables in Snowflake Cortex Analyst using either the YAML file or a graphical GUI approach.

#### **🛠️ Option 1: Add Synonyms using the GUI (AI-Driven)**

* Within the semantic model, **select the logical table** you want to modify and **click Edit**
* Use **Generate Fields** to get AI-suggested synonyms automatically
* **Manually add** company-specific acronyms and domain terms.
* Click **Save**: After adding the desired synonyms, save the semantic model.

📌 The GUI approach is user-friendly and doesn't require direct interaction with YAML files, making it accessible to users who prefer a visual interface.



#### **🛠️ Option 2: Add Synonyms by editing the YAML file**

* Click **<> Edit YAML** option
* Define the synonyms: section under your logical table in the semantic model and redeploy.

Here are suggested synonyms for **each** of your five logical tables. Feel free to tweak them, but aim for 3–5 meaningful alternates per table:

| **Logical Table** | **Description** | **Suggested Synonyms** |
| --- | --- | --- |
| **CUSTOMER\_INDUSTRY** | classify customers by industry (e.g., healthcare, retail, etc.), helping to analyze feature adoption trends across industries. | industry, industry\_classification, customer\_sector, market\_vertical |
| **CUSTOMER\_INTERACTIONS** | track other interactions with customers, such as support tickets or email communications. | support\_logs, interaction\_records, customer\_touches, communications |
| **CUSTOMER\_MEETINGS** | track customer meetings, topics discussed, and which Snowflake features were covered. Each meeting links to a customer. | meeting\_log, customer\_meetings, event\_sessions, feature\_discussions |
| **CUSTOMER\_MEETING\_OUTCOMES** | tracks the outcome of each meeting (e.g., demo scheduled, deal won, follow-up needed). | outcomes, meeting\_results, deal\_outcomes, follow\_up\_status |
| **CUSTOMER\_INSIGHTS** | provides a comprehensive overview of customer interactions, combining core meeting details with customer interaction analysis. | customer\_summary\_view, customer\_360, engagement\_overview, insights\_dashboard |

**📌 Quick Tips for Synonyms**

* **Cover Variations:** Include **singular/plural**, **abbreviation/full name**, **hyphens vs. spaces**, **company acronyms**, and **domain jargon**.
* **Keep It Concise:** Use short phrases or words—avoid full sentences.
* **Iterate Quickly:** Add new terms as you hear them; Cortex Studio applies updates on the next request.
* **Mix AI & Manual:** Use **Generate Fields** for AI-driven suggestions, then **manually add** your proprietary acronyms and niche terms.

By spending just a few minutes here, you’ll unlock far more **accurate**, **robust**, and **user-friendly** natural-language analytics with Cortex Analyst.

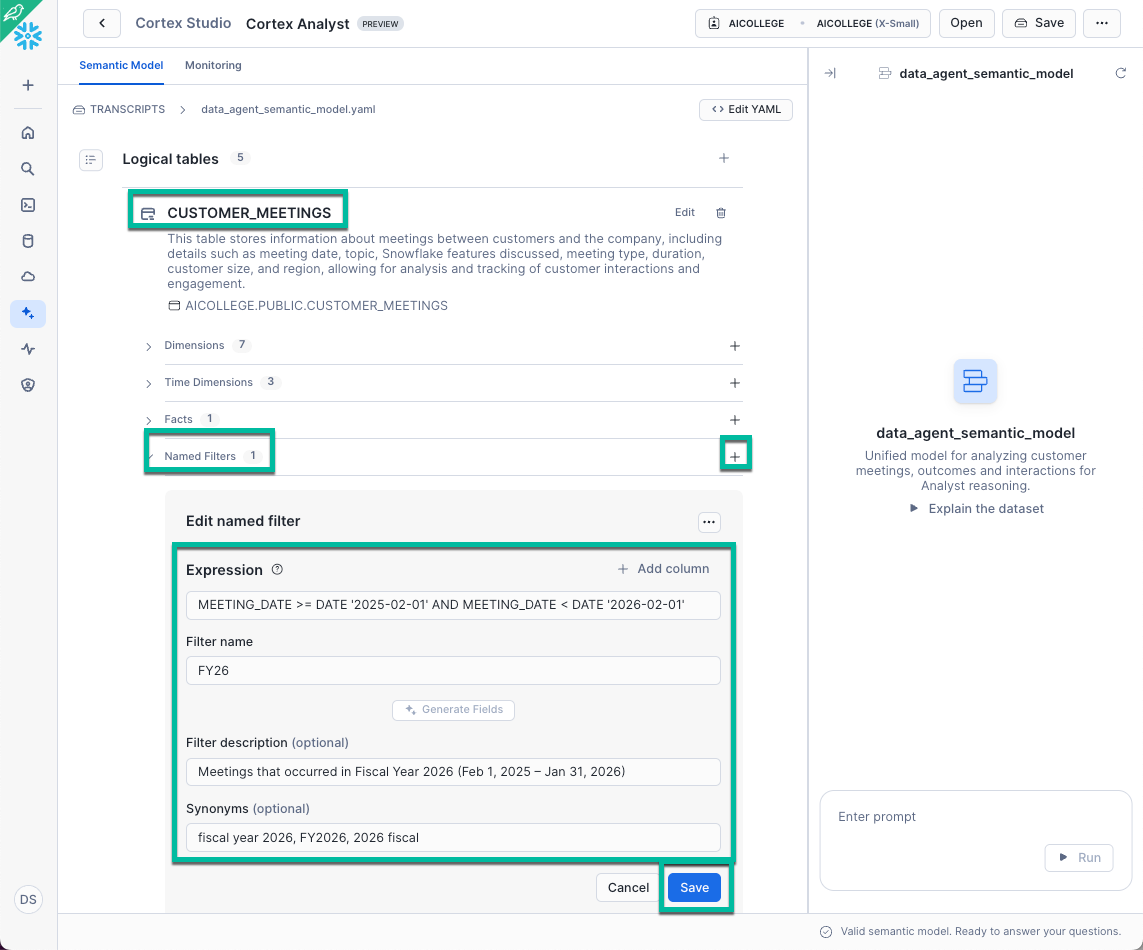
###### **🔹 Using Named Filters in Cortex Analyst**

[**Named Filters**](https://www.snowflake.com/en/engineering-blog/cortex-analyst-text-to-sql-accuracy-bi/) are **reusable expressions**—business rules like fiscal years, geographic scopes, or cohort definitions—that live in your **semantic model**. Once defined, users can refer to them in plain English, and Cortex Analyst embeds the correct SQL logic for you.

* **Business Abstraction:** Encapsulate rules (“FY26” → date >= '2025-02-01' AND date < '2026-01-31') so end users don’t need to remember column names or date math.
* **Synonym-Friendly:** Give each filter a **name** and optional **synonyms** (“fiscal year 26”, “FY-26”) so queries like “show meetings from FY26” or “meetings in fiscal year 2026” both apply the same logic.
* **Consistent Accuracy:** Every time someone invokes your named filter, Cortex Analyst injects its SQL **expression** (the WHERE clause) automatically—no manual coding.

#### **🛠️ Add FY26 as Named Filter**

* In the **Snowsight Semantic Model Editor**, go to the CUSTOMER\_MEETINGS logical table.
* Select **Named Filters** and **click +**
* Edit named filter:
  + **Expression**: MEETING\_DATE >= DATE '2025-02-01' AND MEETING\_DATE < DATE '2026-01-31'
  + **Filter Name**: FY26
  + **Filter Description** (optional): Meetings that occurred in Fiscal Year 2026 (Feb 1, 2025 – Jan 31, 2026)
  + **Synonyms** (optional): fiscal year 2026, FY2026, 2026 fiscal
* **Click Add**



###### **🔹 Using Metrics in Cortex Analyst**

**Metrics** represent **quantitative values** that you want to analyze—typically aggregated numbers like **counts, sums, averages**, or other business KPIs. In a semantic model, metrics help users ask questions like:

* “What is the total number of meetings last quarter?”
* “Which customers had the highest average meeting sentiment?”

By defining metrics in your semantic model, you let Cortex Analyst understand:

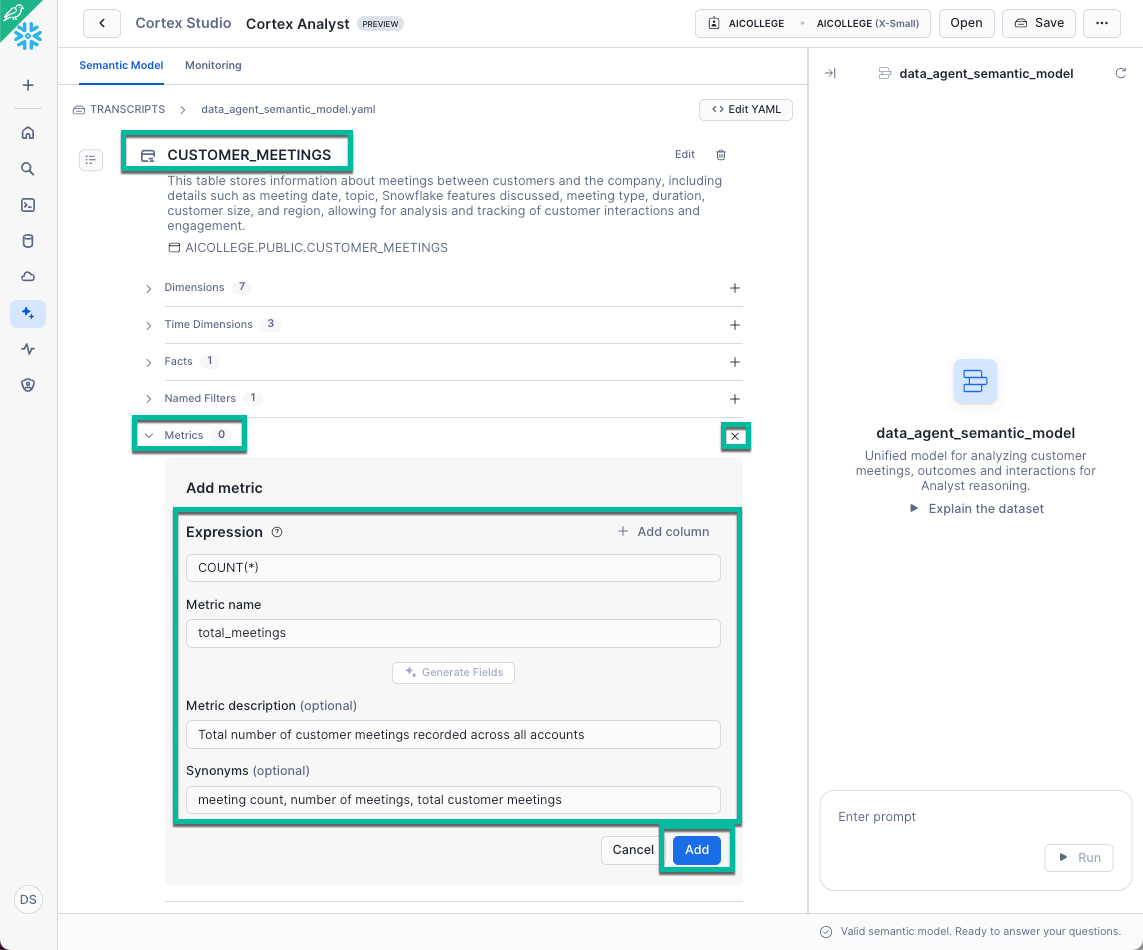
* Which values to calculate
* How to aggregate them
* What names or synonyms users might use to refer to them

**Example Metric to Add:** total\_meetings

If you're working with the CUSTOMER\_MEETINGS table, a good starting metric is the total number of meetings.

#### **🛠️ Add** **TOTAL\_MEETINGS as Metric**

* In the **Snowsight Semantic Model Editor**, go to the CUSTOMER\_MEETINGS logical table.
* Select **Metrics** and **click +**
* Add metric
* **Expression**: COUNT(\*)
  + **Name**: total\_meetings
  + **Description**: Total number of customer meetings recorded across all accounts
  + **Synonyms** (optional): meeting count, number of meetings, total customer meetings
* **Click Add**



##### **🔹 Custom Instructions**

[**Custom Instructions**](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/custom-instructions) are **human-readable rules** you embed in your data\_agent\_semantic\_model.yaml (or via Snowsight’s **Custom Instructions** UI) to enforce your business conventions—**joins**, **filters**, **formatting**, and **safety guards**—across every generated query. Defining them once ensures **consistent date windows**, **currency precision**, and **controlled result sizes** automatically. Check out [this Feature Friday](https://snowflake.seismic.com/app#/learning/library/lesson/2421080?learningApp=true) for more details.

Suggested **Custom Instructions**:

1. **Always use LEFT JOIN for Outcomes** to guarantee that meetings without a recorded outcome still appear (with NULLs) rather than being dropped by an implicit INNER JOIN.
2. **Round All Dollar Values to Two Decimals** to enforce consistent currency formatting (use 50000.00 instead of 50000) across all reports without manual SQL tweaks.
3. **Use Fiscal Quarter Aliases** to make sure queries like “meetings by quarter” always respect your Feb–Apr = FY1 definition instead of standard calendar quarters.
4. **Limit Results for Ad Hoc Queries** to protect from runaway queries in Slack that could overwhelm your warehouse or spill tens of thousands of rows into the channel (use append `LIMIT 1000`).

#### **🛠️ Add Custom Instruction in Snowsight**

* Locate the “**Custom Instructions**” section
* Click Edit next to **Custom Instruction** to add one or more custom instructions following this format. Then click **Apply**.

"- When combining CUSTOMER\_MEETINGS and CUSTOMER\_MEETING\_OUTCOMES, always use a LEFT JOIN to preserve meetings even if no outcome exists.

- Round all DEAL\_VALUE and other monetary fields to two decimal places in the final SELECT clause.

- When grouping or filtering by quarter, use the FISCAL\_QUARTER column rather than calendar quarter.

- If the user’s question does not include a LIMIT, append LIMIT 1000 to prevent accidentally returning massive result sets.

- If the user asks about Gen AI capabilities, add to your WHERE clause: SNOWFLAKE\_FEATURE IN ('Document AI','Snowpark','Cortex AI NLQ')."

📌 **Should you put everything in one multi-line string or separate strings?** Short answer: put everything into **one multi-line string**. custom\_instructions is defined as a single string value in the semantic-model YAML, so treat your rules as plain text (you can keep the dash bullets for readability). A list-of-strings version will still load in YAML, but Cortex Analyst isn’t documented to coerce arrays to strings, so you’re off the “happy path.”

* Or click **<> Edit YAML** to add under custom\_instructions: in your model file
* **Save** (and re-upload your stage or refresh your semantic view),

Now, every Cortex Analyst query will automatically adhere to your tailored business logic - no extra effort required.

##### **🔹 Relationships**

In your semantic model, [**Relationships**](https://www.snowflake.com/en/engineering-blog/snowflake-cortex-analyst-introducting-joins-complex-schemas/) explicitly tell Cortex Analyst how tables connect—both the map **key columns** and **cardinality** explicitly—so Cortex Analyst:

* Delivers **accurate SQL** without missing or reversed joins
* Achieves **better performance** by pushing filters and aggregates to the right tables
* Produces **predictable results** since you control **LEFT** vs **INNER** joins

For this HOL, only three joins are needed to give the semantic model enough information to stitch the five logical tables together.

* **meetings\_to\_outcomes** (left\_outer, one-to-one):  
  CUSTOMER\_MEETINGS.MEETING\_ID → CUSTOMER\_MEETING\_OUTCOMES.MEETING\_ID
* **meetings\_to\_industry** (inner, many-to-one):  
  CUSTOMER\_MEETINGS.CUSTOMER\_NAME → CUSTOMER\_INDUSTRY.CUSTOMER\_NAME
* **interactions\_to\_industry** (inner, many-to-one):  
  CUSTOMER\_INTERACTIONS.CUSTOMER\_NAME → CUSTOMER\_INDUSTRY.CUSTOMER\_NAME

#### **🛠️ Add Relationships in Snowsight**

* Locate the “**Relationships**” section
* **Click +** next to **Relationships** to add the following relationships
* For each relationship, fill in:
  + **Relationship name**: a short, imperative label (e.g. meetings\_to\_outcomes)
  + **Join type**: inner or left\_outer
  + **Relationship type**: select one\_to\_one or many\_to\_one
  + **Left table & Right table**: pick the two logical tables being joined
  + **Relationship columns**: map the left column to the right column
  + **Click Add** to save each one
  + **Click Save** to save your updated semantic model

Below are the relationships you can paste into your YAML (via Edit YAML) or recreate via the GUI:

relationships:

# Link each meeting to its outcome; preserve meetings even if no outcome exists

- name: **meetings\_to\_outcomes**

join\_type: left\_outer

relationship\_type: one\_to\_one

left\_table: CUSTOMER\_MEETINGS

right\_table: CUSTOMER\_MEETING\_OUTCOMES

relationship\_columns:

- left\_column: MEETING\_ID

right\_column: MEETING\_ID

# Attach each meeting back to the customer’s industry

- name: **meetings\_to\_industry**

join\_type: inner

relationship\_type: many\_to\_one

left\_table: CUSTOMER\_MEETINGS

right\_table: CUSTOMER\_INDUSTRY

relationship\_columns:

- left\_column: CUSTOMER\_NAME

right\_column: CUSTOMER\_NAME

# Attach each interaction back to the customer’s industry

- name: **interactions\_to\_industry**

join\_type: inner

relationship\_type: many\_to\_one

left\_table: CUSTOMER\_INTERACTIONS

right\_table: CUSTOMER\_INDUSTRY

relationship\_columns:

- left\_column: CUSTOMER\_NAME

right\_column: CUSTOMER\_NAME



##### **🔹 Verified Queries**

For full details, see the [**Verified Query Repository**](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/verified-query-repository) doc. Curated **natural-language → exact SQL** pairs in your verified\_queries section serve as **concrete patterns** for Cortex Analyst to follow. By embedding a handful of these real-world Q&A examples, you:

* **Anchor SQL Generation:** Cortex replicates your proven joins, filters, and formatting.
* **Speed Onboarding:** Users see contextual “Try this” suggestions that match your use cases.
* **Enforce Guardrails:** Ensures every generated query adheres to your tested business logic, avoiding misinterpretations.

***📌* Pro Tip:** When you embed the Gen AI feature filter as a **Custom Instruction** (SNOWFLAKE\_FEATURE IN ('Document AI','Snowpark','Cortex AI NLQ')), Cortex Analyst automatically appends it for any “Gen AI” query. This means your **Verified Queries** can stay focused on core logic—without manually repeating that filter—because the **Custom Instruction** handles it globally.

verified\_queries:

- question: "Which customers asked for a Gen AI POC?"

sql: |

SELECT CUSTOMER\_NAME, REQUEST\_DATE

FROM CUSTOMER\_INSIGHTS

WHERE REQUEST\_TYPE = 'Gen AI POC';



Once published, Cortex Analyst prioritizes your **Verified Queries**—delivering **precise**, **business-aligned** SQL from day one.

#### **🛠️ Add Verified Queries in Snowsight**

* **Click** the **Explain the dataset option** on the right side of the no-code UI
* Notice how three potential questions appear
* In the **Enter prompt option**, type the question **What are all the unique customer names?**and **click > Run**
* Review the Cortex Analyst-generated SQL, if you agree, **click + Verified query**
* Provide the name **ListCustomers**
* **Click Add**
* Repeat to add questions like:

| **Name** | **Question** | **SQL** |
| --- | --- | --- |
| InteractionCountByIndustry | How many customer interactions occurred by industry? | SELECT  ci.industry, COUNT(\*) AS interaction\_count  FROM customer\_interactions AS i  INNER JOIN customer\_industry AS ci ON i.customer\_name = ci.customer\_name  GROUP BY ci.industry  ORDER BY interaction\_count DESC NULLS LAST  LIMIT 1000 |
| EngagementByIndustry | What was our customer engagement—unique customers, total meetings, positive feedback count, and total deal value—by industry? | SELECT  ci.industry, COUNT(DISTINCT cm.customer\_name) AS unique\_customers,  COUNT(\*) AS total\_meetings, SUM(  CASE  WHEN cmo.customer\_feedback = 'Positive' THEN 1  ELSE 0  END  ) AS positive\_feedback\_count, ROUND(SUM(cmo.deal\_value), 2) AS total\_deal\_value  FROM customer\_meetings AS cm  JOIN customer\_industry AS ci ON cm.customer\_name = ci.customer\_name  LEFT JOIN customer\_meeting\_outcomes AS cmo ON cm.meeting\_id = cmo.meeting\_id  GROUP BY ci.industry  ORDER BY total\_deal\_value DESC  LIMIT 1000 |
| FeatureAdoptionBySize | How many meetings discussed each Snowflake feature by customer size? | SELECT  cm.snowflake\_feature, ci.customer\_size, COUNT(\*) AS discussion\_count,  COUNT(DISTINCT cm.customer\_name) AS unique\_customers  FROM customer\_meetings AS cm  JOIN customer\_industry AS ci ON cm.customer\_name = ci.customer\_name  GROUP BY cm.snowflake\_feature, ci.customer\_size  ORDER BY discussion\_count DESC  LIMIT 1000 |
| RegionalPerformance | What was the regional performance—unique customers, average meeting duration, and total deal value? | SELECT  ci.region, COUNT(DISTINCT cm.customer\_name) AS unique\_customers,  ROUND(AVG(cm.meeting\_duration), 2) AS avg\_meeting\_duration,  ROUND(SUM(cmo.deal\_value), 2) AS total\_deal\_value  FROM customer\_meetings AS cm  JOIN customer\_industry AS ci ON cm.customer\_name = ci.customer\_name  LEFT JOIN customer\_meeting\_outcomes AS cmo ON cm.meeting\_id = cmo.meeting\_id  GROUP BY ci.region  ORDER BY total\_deal\_value DESC  LIMIT 1000 |
| CustomerJourneySummary | For each customer, how many meetings did they have, which meeting types and features were covered, and what was their largest deal value? | SELECT  cm.customer\_name, ci.industry, COUNT(DISTINCT cm.meeting\_id) AS total\_meetings,  STRING\_AGG(DISTINCT cm.meeting\_type, ', ') AS meeting\_types,  STRING\_AGG(DISTINCT cm.snowflake\_feature, ', ') AS features\_discussed,  MAX(cmo.deal\_value) AS largest\_deal\_value  FROM customer\_meetings cm  JOIN customer\_industry ci ON cm.customer\_name = ci.customer\_name  LEFT JOIN customer\_meeting\_outcomes cmo ON cm.meeting\_id = cmo.meeting\_id  GROUP BY cm.customer\_name, ci.industry  ORDER BY total\_meetings DESC  LIMIT 1000; |
| MeetingOutcomeStats | What were the counts, average meeting durations, total deal values, and industries represented for each meeting outcome? | SELECT  cmo.outcome,COUNT(\*) AS outcome\_count,  ROUND(AVG(cm.meeting\_duration), 2) AS avg\_meeting\_duration,  ROUND(SUM(cmo.deal\_value), 2) AS total\_deal\_value,  COUNT(DISTINCT ci.industry) AS industries\_represented  FROM customer\_meeting\_outcomes AS cmo  JOIN customer\_meetings AS cm ON cmo.meeting\_id = cm.meeting\_id  JOIN customer\_industry AS ci ON cm.customer\_name = ci.customer\_name  GROUP BY cmo.outcome  ORDER BY outcome\_count DESC  LIMIT 1000 |

* After adding each query, verify that it appears under Verified Queries.
* **Click Save** (top right) to persist your semantic model.

### **Step 1.4: Slack Setup**

#### **🛠️ Create New Slack Workspace**

* Go to this [website](https://slack.com/get-started#create).
* Use a ***personal email address*** and select **Continue**.
* Check your email for a 6-digit Slack confirmation code
* Select **Create a Workspace**
* Name your company **“CollegeAI DataAgent**” and select **Next**
* Enter **your name** and click **Next**
* Feel free to skip the “Who else is on the D4B\_SkiCarCo team?” step. **Select Skip this step**.
* Choose the “**Start with Free**” version

#### **🛠️ Create & Configure Slack App**

* Go to <https://api.slack.com/apps> and **select Create New App**
* Select **From a manifest** option
* Select your newly created Slack workspace
* Replace the json with this [app\_manifest.json](https://drive.google.com/file/d/1Khu0bUokNi6C_9gA-Scu5Ipmu2YXOfBc/view?usp=drive_link) code, **select Next**
* Select **Create**
* On the **Basic Information** tab, scroll down to **App-Level Tokens** and select **Generate Tokens and Scopes**
* Provide the Token name “**app**”. Select “**connections:write**” for the scope. Select **Generate**.
* **Copy** the **app token** and save it in a place you can easily find later
* Navigate to **OAuth & Permissions** tab, scroll down to OAuth Tokens and select **Install to CollegeAI DataAgent**
* Select **Allow**
* **Copy** the **Bot User OAuth Token** and save it in a place you can easily find
* Navigate to **App Home** and **check** "Allow users to send Slash commands and messages from the messages tab".

*Optional: Want to personalize your Slack app? Here’s the* [*link*](https://drive.google.com/file/d/1T_Ro6I3ruifcbsCnSvspePqKUC0GeLTO/view?usp=drive_link) *to the Snowflake Data Agent, if interested.*

#### **🛠️ Create Snowflake / Slack API Integration**

##### **Populate your dataagent.env**

* + Return to VS Code, **open** [**dataagent.env**](https://drive.google.com/file/d/16s3yLOMe1nZAnM3vqT6pxnu8NFKXUTyb/view?usp=drive_link) and fill in **your values** and **save** the updated **dataagent.env** file

# Slack credentials

SLACK\_BOT\_TOKEN=**xoxb-5...**

SLACK\_APP\_TOKEN=**xapp-1-A08...**

# Snowflake service user + key-pair + programmatic access token (PAT is only for REST call to /api/v2/cortex/analyst/message)

SNOWFLAKE\_USER=dataagent\_user

SNOWFLAKE\_ACCOUNT=**<your account name>**

SNOWFLAKE\_PRIVATE\_KEY\_PATH=rsa\_private\_key.pem

SNOWFLAKE\_PAT=**eyJra...**

# Semantic model in your stage

SF\_DATABASE=AICOLLEGE

SF\_SCHEMA=PUBLIC

SF\_ROLE=AICOLLEGE

SF\_WAREHOUSE=AICOLLEGE

SF\_STAGE=TRANSCRIPTS

SF\_MODEL\_FILE=data\_agent\_semantic\_model.yaml

##### ***📌*** *Remember that SNOWFLAKE\_ACCOUNT cannot contain underscores. To avoid the SSL: CERTIFICATE\_VERIFY\_FAILED* [*error*](https://snowflake.slack.com/archives/C06GTRF515G/p1731098042147199)*, change to a hyphen. For instance, SFSEEUROPE-US\_DEMO603 needs to be SFSEEUROPE-US-DEMO603.*

##### **Create & activate your venv**

* + In VS Code’s integrated terminal, run
  + Verify your Python version

python3 --version

# must report 3.10.x, 3.11.x, or 3.12.x



* + ***📌*** *Make sure you have Python 3.10–3.12 installed (Snowpark no longer supports 3.8/3.9.) If needed, run the following command*

*# fail fast if python3 isn’t 3.12*

*if [[ "$(python3 -c 'import sys; print(sys.version\_info[:2])')" != "(3, 12)" ]]; then*

*echo "ERROR: python3 must be version 3.12.x" >&2*

*exit 1*

*fi*

python3 -m venv .venv

source .venv/bin/activate

##### **Install only the required packages**

* + Still in the activated venv run:

pip install --upgrade pip

pip install \

slack\_bolt \

requests \

pandas \

'snowflake-connector-python[pandas]' \

snowflake-snowpark-python \

snowflake-core \

python-dotenv \

sqlalchemy \

snowflake-sqlalchemy

##### **Verify your installs**

* + Run the provided check\_deps.py:

python3 check\_deps.py

* You should see:

✅ python-dotenv

✅ slack-bolt

✅ requests

✅ pandas

✅ snowflake-connector-python

✅ snowflake-connector-python[pandas]

✅ snowflake-snowpark-python

✅ snowflake-core

✅ SQLAlchemy

✅ snowflake-sqlalchemy

##### **Run your Slack app**

* + Activate the venv (if you haven’t already or you are reopening the VS Code)

source .venv/bin/activate



* + Start the bot:

python3 slackapp.py



* You should see **both** lines appear in the terminal:

🚀 CollegeAI Data-Agent is running!

⚡️ Bolt app is running!

*If you don’t see this, check that your SLACK\_BOT\_TOKEN / SLACK\_APP\_TOKEN are correct and that the workspace has the app installed.*

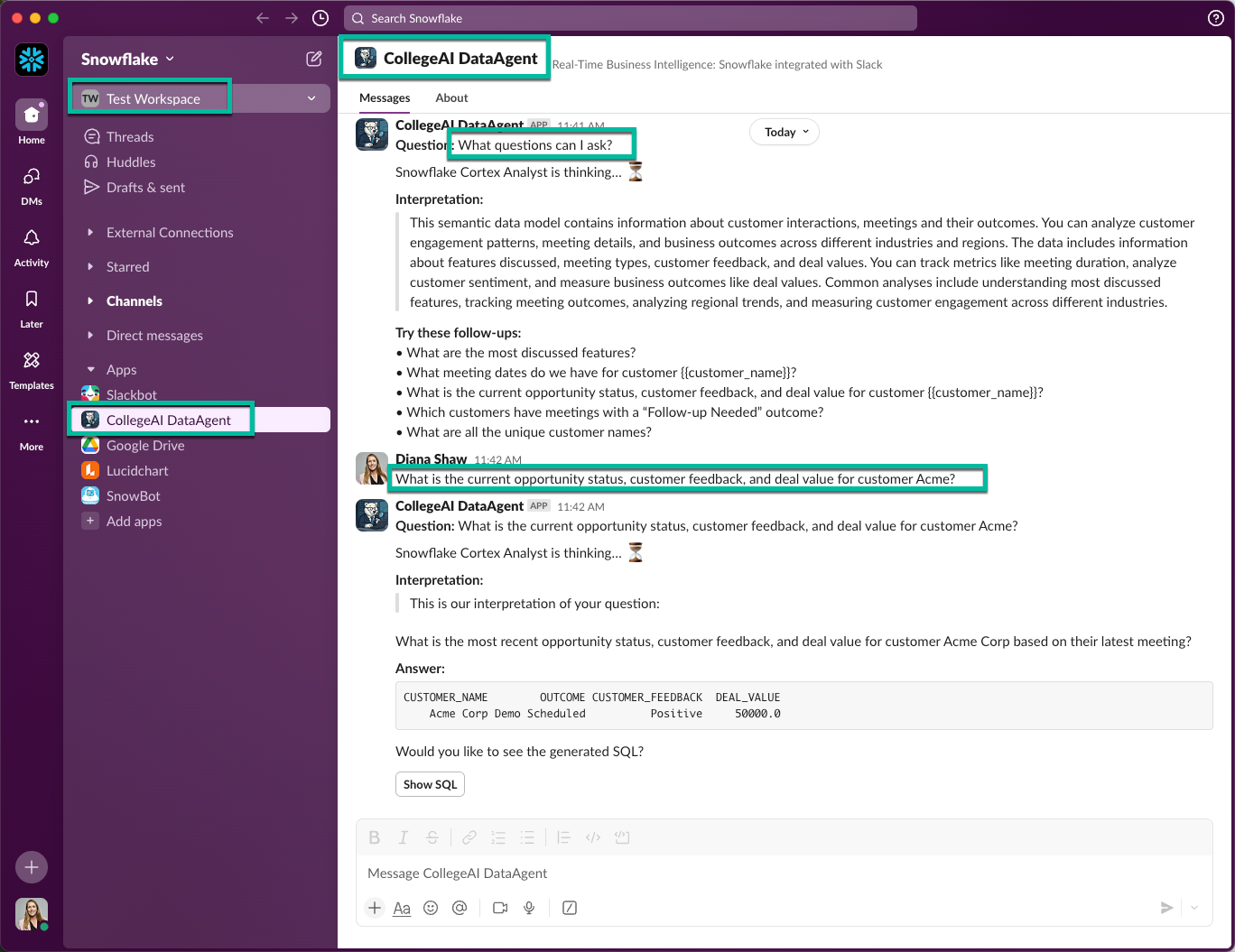
* **Open Slack** (desktop or web) and switch to your **CollegeAI DataAgent workspace**
* **Test a natural-langue query** - two equivalent ways:

| **Where to type** | **Example input** | **What happens** |
| --- | --- | --- |
| **Any channel where the bot is a member** (e.g. #collegeai-dataagent) | /askcortex What questions can I ask? | Slash-command triggers the bot; you’ll see the “thinking…” response, then the answer. |
| **Direct message to the bot** (or any channel) | What questions can I ask? | The app treats plain text as a question and replies in-thread. |

#### **🛠️ Test Your Cortex Analyst Integration**

**🚧 Caution:** Your sample data is **very small** (only 6 customers, each with one meeting and one transcript). Stick to **simple, high-value queries**—don’t get creative or expect deep insights beyond the structured data you’ve loaded.

You’re querying **only structured tables** right now, so focus on these examples:

1. ***What example prompts can I ask?****Returns your built-in “Try this” suggestions from the semantic model.*
2. ***Which customers did we meet with?****Lists distinct customer names (plus industry/region) from* ***CUSTOMER\_MEETINGS****.*
3. ***Which customers planned a Gen AI POC?****Filters* ***SNOWFLAKE\_FEATURE*** *to* ***Document AI****,* ***Snowpark****,* ***Cortex AI NLQ*** *and* ***OUTCOME = 'POC Planned'****.*
4. ***What features are customers most interested in?****Aggregates* ***SNOWFLAKE\_FEATURE*** *counts to surface top-requested topics.*
5. ***Who scheduled a demo, and for which feature?****Finds meetings where* ***OUTCOME = 'Demo Scheduled'*** *and returns* ***SNOWFLAKE\_FEATURE*** *per customer.*

Feel free to tweak these slightly, but remember—your data is shallow, and Cortex Analyst can only work with the structured tables you’ve defined.

#### **🛠️ DORA Evaluation #55: Phase 1 Completion Check**

To receive credit for Phase 1 of the College of AI: Data Agent HOL, run the built-in DORA check **directly in Slack**:

* Go to the **CollegeAI DataAgent** channel
* Either issue the slash command:

/askcortex SEAI55

* Or simply type (no slask):

SEAI55

If everything is set up correctly, you’ll see a green ✅ and the message

*Here is an example screenshot of the DORA SEAI55 completion confirmation:*

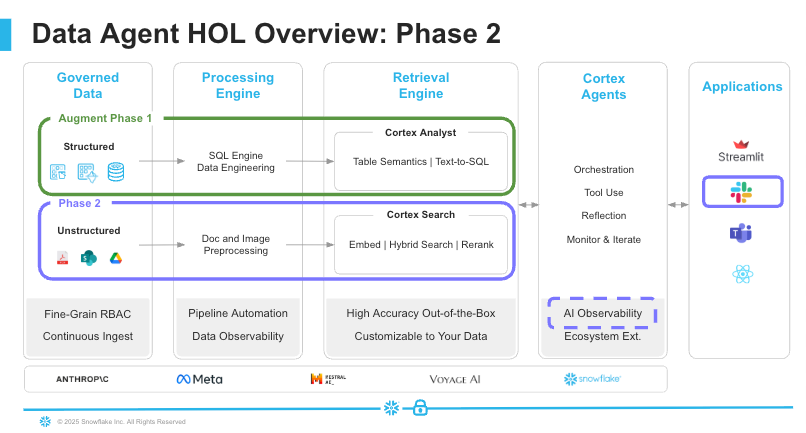


SEAI55 validates that your semantic model has been successfully loaded in Snowflake with 6 distinct customers. This DORA evaluation must be run **directly in Slack** (not in SQL or a Snowflake notebook):

⚠️ **Important:** This DORA step must pass before you can move on to Phase 2.

## **Phase 2: Cortex Search**

Now you’ll extend your Data Agent to index and query unstructured meeting transcripts using [**Snowflake Cortex Search**](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-search/cortex-search-overview). Cortex Search is a **fully managed hybrid search** service—**combining keyword and vector search**—to semantically index any data stored in Snowflake and power Retrieval-Augmented Generation (RAG) workflows **without extra ETL pipelines**.



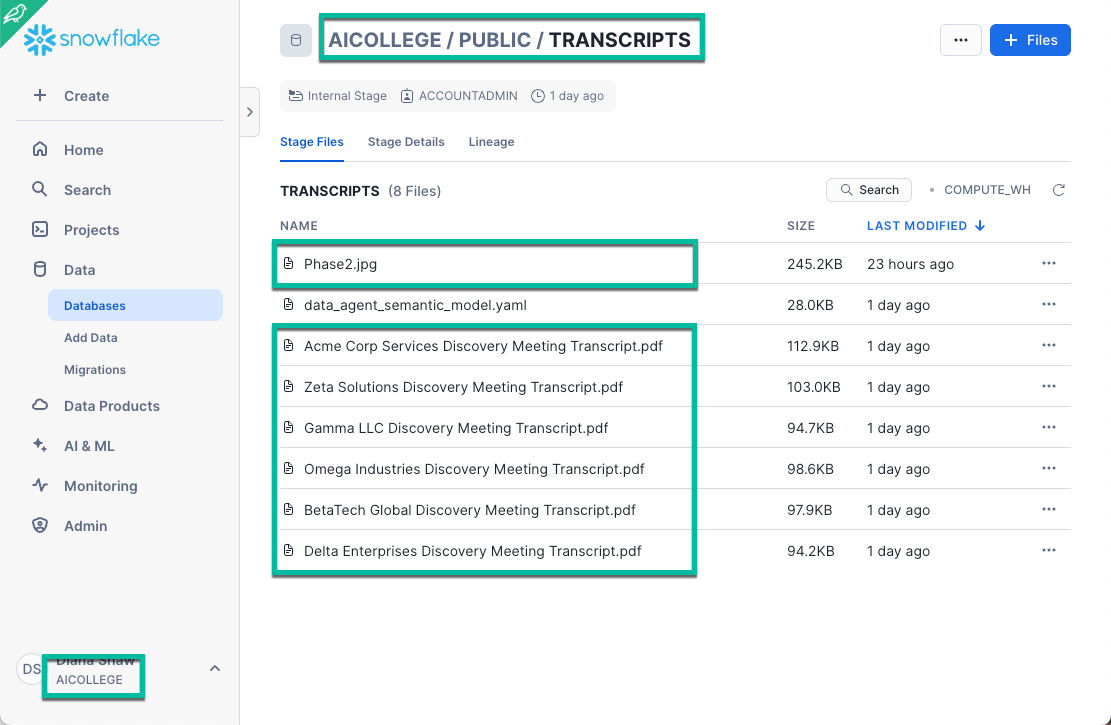
Over the next steps, you will:

* **Ingest transcripts**: upload PDFs (and images) into a Snowflake stage
* **Parse & chunk**: run PARSE\_DOCUMENT and SPLIT\_TEXT\_RECURSIVE\_CHARACTER to break documents into searchable pieces
* **Analyze content**: use Cortex Complete functions (sentiment, key phrases, risk scoring) to enrich each chunk
* **Augment your semantic model**: add a new TRANSCRIPT\_FACTS logical table (with metrics, filters, relationships, and verified queries)
* **Enable Dynamic Literal Retrieval**: hook up your search service to string dimensions so Cortex Analyst can resolve names and phrases at query time

By the end of Phase 2, your agent will be able to hybrid-search across both structured tables and transcript text—unlocking much richer, context-aware answers directly in Slack.

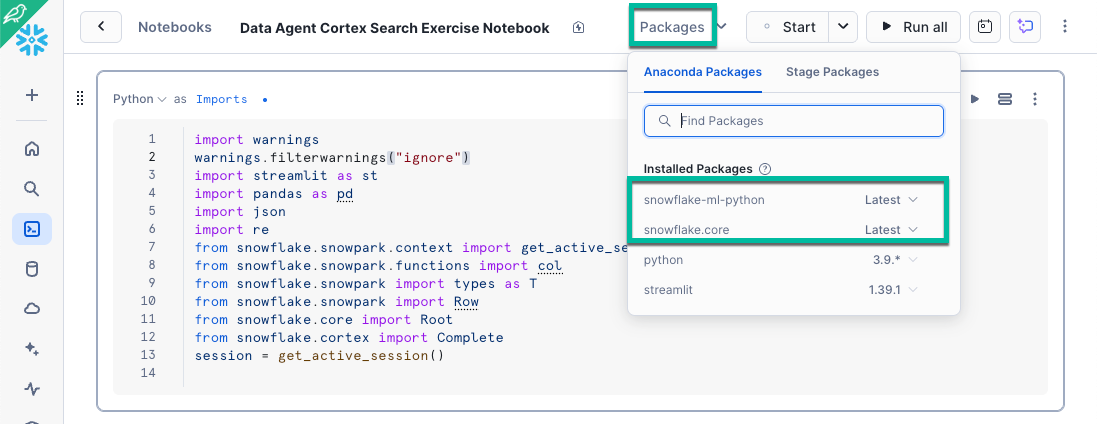
### **🛠️ Step 2.1: Load Discovery Meeting Transcripts**

* Use **AICOLLEGE** Role
* Navigate to **Data** >> **Databases** >> **AICOLLEGE** >> **PUBLIC** >> **Stages** >> **TRANSCRIPTS**
* Click **+ Files** and choose all six (6) PDF & (1) JPG files previously downloaded
  + **Acme Corp Services Discovery Meeting Transcript.pdf**
  + **BetaTech Global Discovery Meeting Transcript.pdf**
  + **Delta Enterprises Discovery Meeting Transcript.pdf**
  + **Gamma LLC Discovery Meeting Transcript.pdf**
  + **Omega Industries Discovery Meeting Transcript.pdf**
  + **Zeta Solutions Discovery Meeting Transcript.pdf** and
  + **Phase2.jpg** files
* Click **Upload**



### **🛠️ Step 2.2: Snowflake Notebook Setup**

* Navigate to **Snowflake Notebooks**
* Select the "**v**” beside the “**+ Notebook**" button
* Select **Import .ipynb file** option
* Import the [Data Agent Cortex Search Exercise Notebook](https://drive.google.com/file/d/1CXQ70bJkg53dXutOUpJt3NIUEdMjKoYZ/view?usp=drive_link)
* Select **AICOLLEGE** and **PUBLIC** for Notebook location
* Select **AICOLLEGE** for Notebook warehouse
* Select **Create**
  + Go to your **Snowflake Notebooks Packages**
* Select **Packages**, type snowflake, **Select snowflake-ml-python**
* Repeat for  **snowflake.core**
* You are now ready to complete the missing code and run each cell



Below is a guide to help you know **what to update and why**.

#### **XXX Replacement Guide**

Use this table to guide your updates. These placeholder values (`XXX`) must be replaced for the notebook to run successfully.

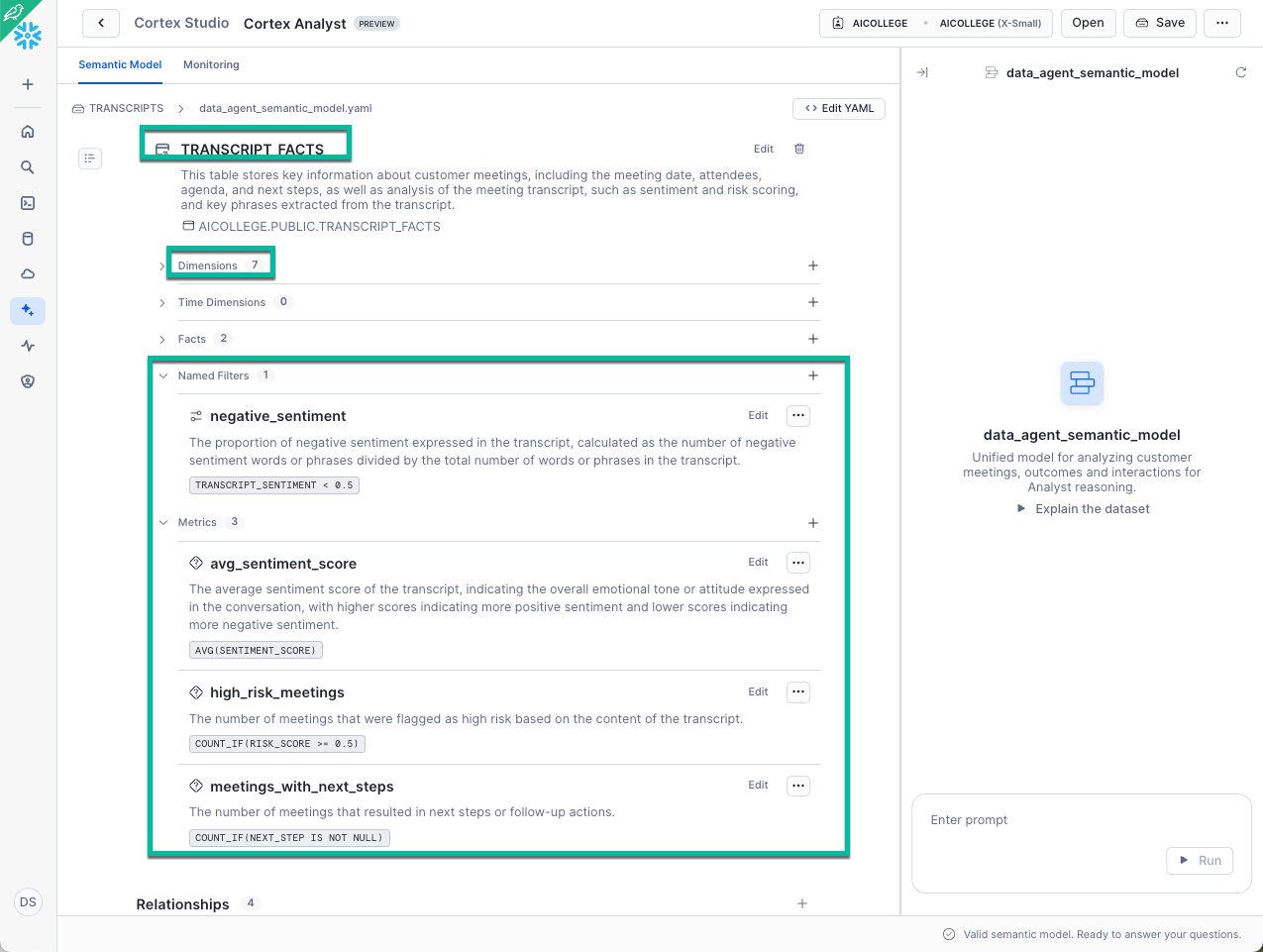
| 📍 Cell Context | Replace **XXX** With |
| --- | --- |
| Use Cortex's PDF parser to update SNOWFLAKE.XXX(...) | Replace with [PARSE\_DOCUMENT](https://docs.snowflake.com/en/sql-reference/functions/parse_document-snowflake-cortex): "SNOWFLAKE.CORTEX.PARSE\_DOCUMENT(...)" |
| Use Cortex’s layout-based parser for OBJECT\_CONSTRUCT('mode','XXX') | Replace: "layout" |
| Use Cortex’s text chunking function for SNOWFLAKE.XXX(...) (text splitter) | Replace with [SPLIT\_TEXT\_RECURSIVE\_CHAR](https://docs.snowflake.com/en/sql-reference/functions/split_text_recursive_character-snowflake-cortex): SNOWFLAKE.CORTEX.SPLIT\_TEXT\_RECURSIVE\_CHARACTER(...) |
| Ensure CHUNK is final output for chunk.value::STRING AS XXX | Replace with: chunk.value::STRING AS CHUNK |
| Create the [Cortex Search Service](https://docs.snowflake.com/sql-reference/sql/create-cortex-search) correctly for CREATE OR REPLACE XXX AICOLLEGE.PUBLIC.TRANSC… | Use CREATE OR REPLACE CORTEX SEARCH SERVICE AICOLLEGE.PUBLIC.TRANSC… |
| Reference the correct Cortex Search Service cortex\_search\_services["XXX"] | Use "TRANSCRIPTS\_SEARCH\_SERVICE" |
| Use Cortex Complete models for single-turn generation  model = "XXX" | Consider testing [large Cortex models](https://docs.snowflake.com/en/user-guide/snowflake-cortex/llm-functions#large-models): "claude-3-5-sonnet"or "mistral-large2" |
| Use Cortex COMPLETE function for structured extraction  reply\_text = XXX(...) | Use Cortex COMPLETE for [extracting structured outputs](https://docs.snowflake.com/en/user-guide/snowflake-cortex/complete-structured-outputs) |
| Use Cortex SENTIMENT for scoring transcript sentiment sent\_score = session.sql(  "SELECT SNOWFLAKE.XXX(?) AS S" | Refer to [this example](https://docs.snowflake.com/en/sql-reference/functions/sentiment-snowflake-cortex) and use SENTIMENT |
| Use Snowflake Complete function for extracting key phrases  reply = XXX | Use Cortex COMPLETE for [extracting structured outputs](https://docs.snowflake.com/en/user-guide/snowflake-cortex/complete-structured-outputs) |
| Use LLM model for key phrases  model = "XXX" (key phrases) | Consider testing [large Cortex models](https://docs.snowflake.com/en/user-guide/snowflake-cortex/llm-functions#large-models): "claude-3-5-sonnet"or "mistral-large2" |
| Use Snowflake Complete for risk scoring  risk\_reply = XXX( | Use Cortex COMPLETE for [extracting structured outputs](https://docs.snowflake.com/en/user-guide/snowflake-cortex/complete-structured-outputs) |
| Use LLM model for key phrases  model = "XXX" (risk scoring) | Consider testing [large Cortex models](https://docs.snowflake.com/en/user-guide/snowflake-cortex/llm-functions#large-models): "claude-3-5-sonnet"or "mistral-large2" |
| Reference correct Cortex Search Service for batch pipline  cortex\_search\_services["XXX"] | Use "TRANSCRIPTS\_SEARCH\_SERVICE" |
| Use Sentiment for batch sentiment scoring in loop  "SELECT SNOWFLAKE.XXX(?) AS S" | Refer to [this example](https://docs.snowflake.com/en/sql-reference/functions/sentiment-snowflake-cortex) and use "SELECT SNOWFLAKE.CORTEX.SENTIMENT(?) AS S" |
| Use Anthropic or Mistral model in key phrase extraction  kp\_reply = XXX(model="XXX", ...) | Use "Complete(model="mistral-large2", ...) |
| Use Anthropic or Mistral model in risk scoring  risk\_reply = XXX(model="XXX", ...) | Use "Complete(model="mistral-large2", ...) |
| Final output table in batch mode  df.write.save\_as\_table("XXX", ...) | Use: "TRANSCRIPT\_FACTS" |

### **Step 2.3: Augment Semantic Model**

* Navigate to back your **Cortex Analyst data\_agent\_semantic\_model**
* Using the instructions from Step 1.3 to complete the following tasks:

#### **🛠️ Add an additional Logical Table with Metrics and a Named filter**

* **Add TRANSCRIPT\_FACTS** as a new **Logical table**,
  + Include **all columns**,
  + Use GenAI to **Generate Fields**
* Add the following **Metrics** to the **TRANSCRIPT\_FACTS logical table** with GenAI to **Generate Fields**
  + avg\_sentiment\_score → AVG(SENTIMENT\_SCORE)
  + high\_risk\_meetings → COUNT\_IF(RISK\_SCORE >= 0.5)
  + meetings\_with\_next\_steps → COUNT\_IF(NEXT\_STEP IS NOT NULL)
* Add **Named filter** to the **TRANSCRIPT\_FACTS logical table** with GenAI to **Generate Fields**
  + negative\_sentiment → TRANSCRIPT\_SENTIMENT < 0.5



#### **🛠️ Add additional Relationship**

* Add the following **Relationship** to link each mock meeting to its mock transcript
  + Add meeting\_transcripts with this HOL configuration\*
    - join\_type: inner
    - relationship\_type: one\_to\_one
    - Link: CUSTOMER\_MEETINGS.CUSTOMER\_NAME → TRANSCRIPT\_FACTS.CUSTOMER\_NAME



#### **🛠️ Add additional Verified Queries**

* Add **Verified queries** to quickly onboard **TRANSCRIPT\_FACTS**
* Name:high\_sentiment\_low\_risk
* Question: **Which customers have high transcript sentiment and low risk?**
* SQL:

SELECT t.customer\_name, t.transcript\_sentiment, t.risk\_scoring

FROM transcript\_facts AS t

WHERE t.transcript\_sentiment > 0.5 AND t.risk\_scoring < 0.4

ORDER BY t.transcript\_sentiment DESC NULLS LAST

LIMIT 1000

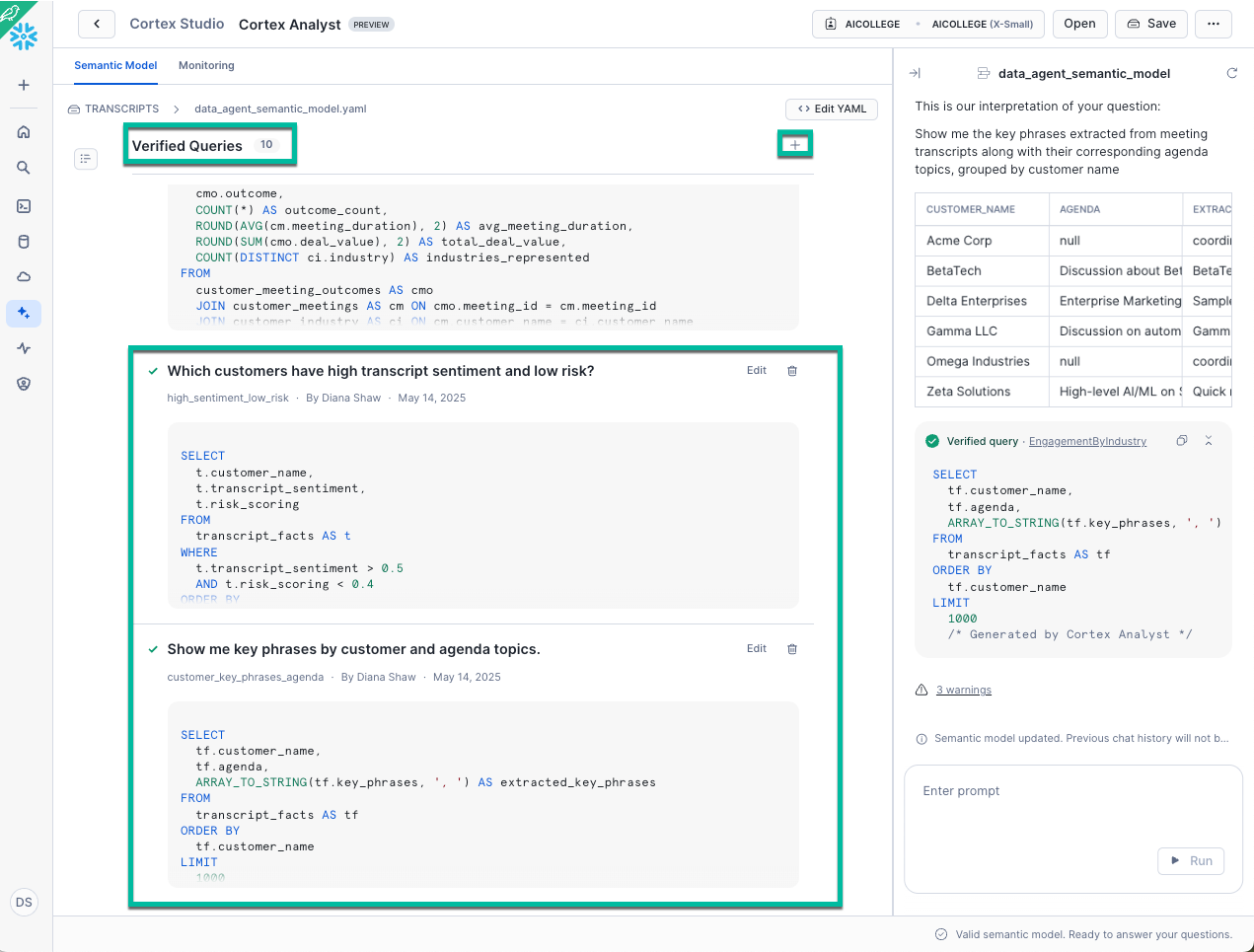
* + Name:customer\_key\_phrases\_agenda
  + Question: **Show me key phrases by customer and agenda topics.**
  + SQL:

SELECT tf.customer\_name, tf.agenda, ARRAY\_TO\_STRING(tf.key\_phrases, ', ') AS extracted\_key\_phrases

FROM transcript\_facts AS tf

ORDER BY tf.customer\_name

LIMIT 1000



##### **🔹 Dynamic Literal Retrieval**

[**Dynamic Literal Retrieval (DLR)**](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst/cortex-analyst-search-integration) lets Cortex Analyst fetch **literal values**—like **customer names**, **product types**, or **industries**—on demand via a **Cortex Search Service**, instead of embedding every possible value in your semantic model.

* **On-the-fly Lookup:** Retrieves relevant literals at query time using **semantic** or **keyword** search.
* **Avoids Staleness:** No need to reload the model when new customers or products appear.
* **Handles Variations:** Matches **misspellings**, **partial names**, or **business-style phrasing** seamlessly.

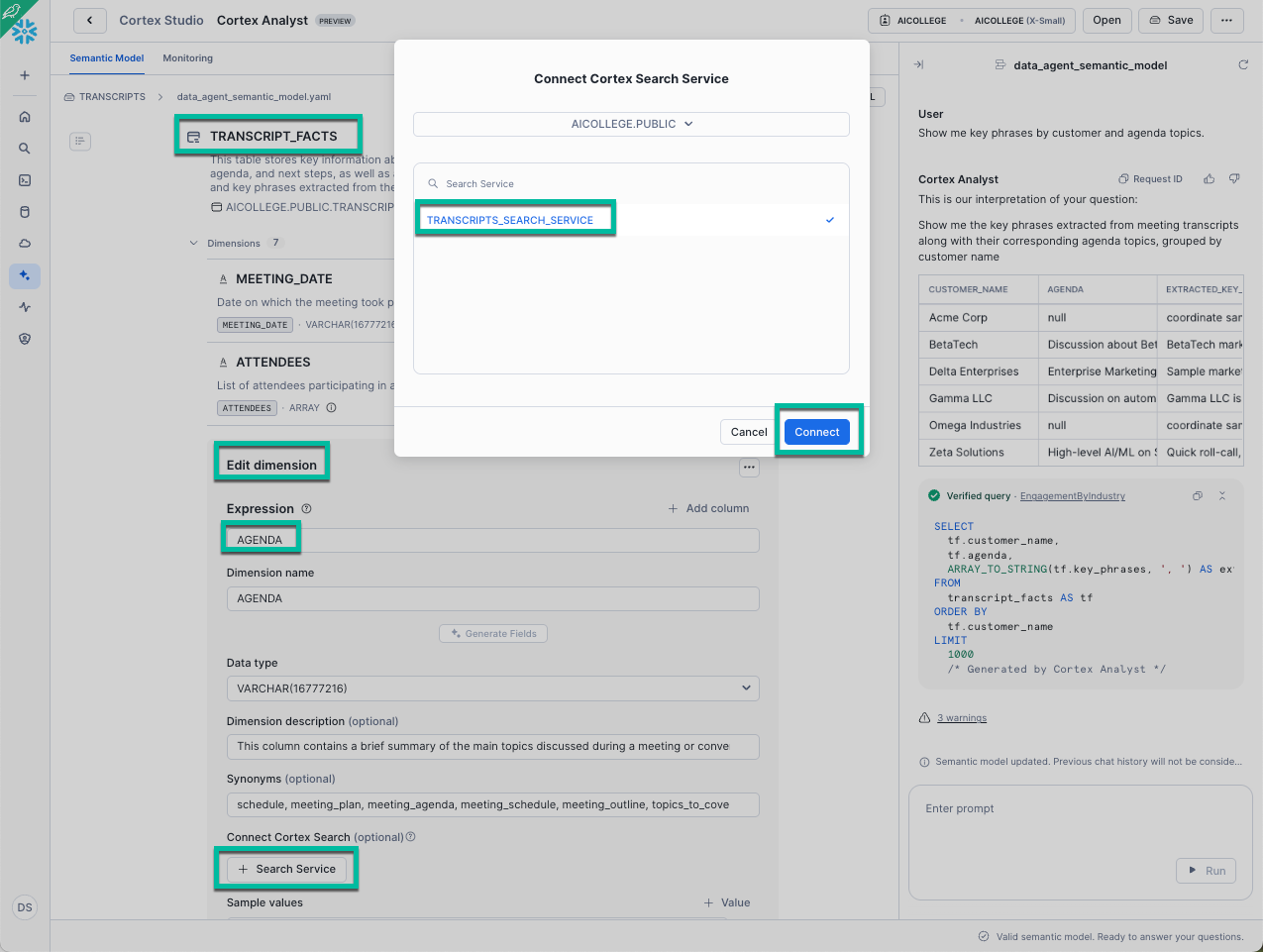
###### **📌 Why DLR Matters in the Data Agent Scenario**

Your TRANSCRIPT\_FACTS table holds rich text fields (e.g. customer\_name, agenda, next\_step, key\_phrases). By indexing these with a Cortex Search Service and linking it under dynamic\_literal\_retrieval, Cortex Analyst can:

| **Scenario** | **Without DLR** | **With DLR** |
| --- | --- | --- |
| User asks “Show all Gamma LLC meetings with risk > 0.7” | Fails if **Gamma LLC** isn’t preloaded | Finds **Gamma LLC** via search service lookup |
| User says “find meetings about marketing analytics” | Misses if **marketing analytics** isn’t in literals | Discovers via full-text search in agenda or key\_phrases |
| New customer added to TRANSCRIPT\_FACTS | Requires **model reload** | **Auto-resolved** at query time via search |

#### **🛠️ Option 1: Enable Dynamic Literal Retrieval using the GUI**

* Within the semantic model, **navigate to** AGENDAdimension in theTRANSCRIPT\_FACTS **logical table** and **click Edit**
* Below Synonyms, **click + Search Service**
* **Select AICOLLEGE >> PUBLIC** to locate and **select** **TRANSCRIPTS\_SEARCH\_SERVICE** and **click Connect**
* **Click Save**



#### **🛠️ Option 2: Enable Dynamic Literal Retrieval via code**

* In your Snowflake Notebook, create a new **Cortex Search Service** on TRANSCRIPT\_FACTS:

CREATE OR REPLACE CORTEX SEARCH SERVICE TRANSCRIPT\_FACTS\_SEARCH

ON CHUNK

ATTRIBUTES customer\_name, agenda, next\_step, key\_phrases

WAREHOUSE = AICOLLEGE

AS (

SELECT

customer\_name,

agenda,

next\_step,

key\_phrases,

TO\_VARCHAR(transcript\_sentiment) AS CHUNK

FROM AICOLLEGE.PUBLIC.TRANSCRIPT\_FACTS

);



* In your data\_agent\_semantic\_model.yaml, you would link the service to your by referencing it in the model’s literal\_retrieval\_services block. For example:

literal\_retrieval\_services:

- service\_name: TRANSCRIPT\_FACTS\_SEARCH

match\_columns:

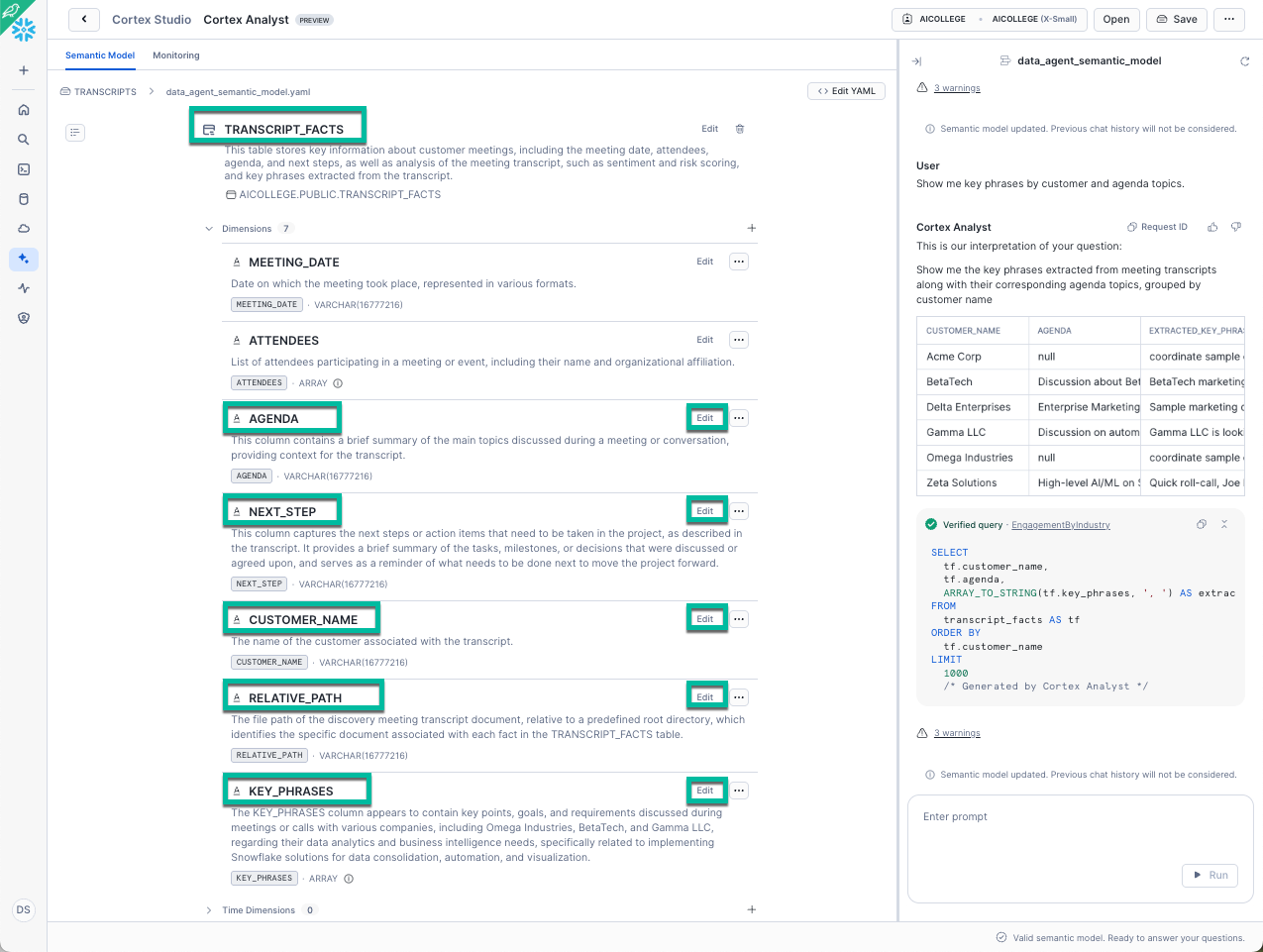
- customer\_name

- agenda

- key\_phrases

#### **🛠️ Add Dynamic Literal Retrievals**

* We recommend using Option 1 to create DLRs for the following string columns (Dimensions) in your Transcripts Logical table:
  + customer\_name
  + relative\_path
  + agenda
  + next\_step
  + key\_phrases



Dynamic Literal Retrieval makes your **Cortex Analyst smarter and more resilient** by searching actual Snowflake content when users refer to real-world names or topics. It’s especially useful in data agent scenarios where string values (like customer names or use cases) change often. By doing this extra setup, Cortex Analyst can now:

* **Auto-resolve free-text values** like:
  + “Show me insights from customers discussing architecture decks or Snowpipe Streaming”
* **Filter or group across unstructured concepts**:
  + “Which customers mentioned risk or data governance in their transcript?”
* **Handle user queries without preloading literals**:
  + “What did Gamma say about marketing analytics?” will dynamically resolve Gamma LLC and marketing analytics.

#### **🛠️ Test Cortex Search-enhanced Semantic model in Slack**

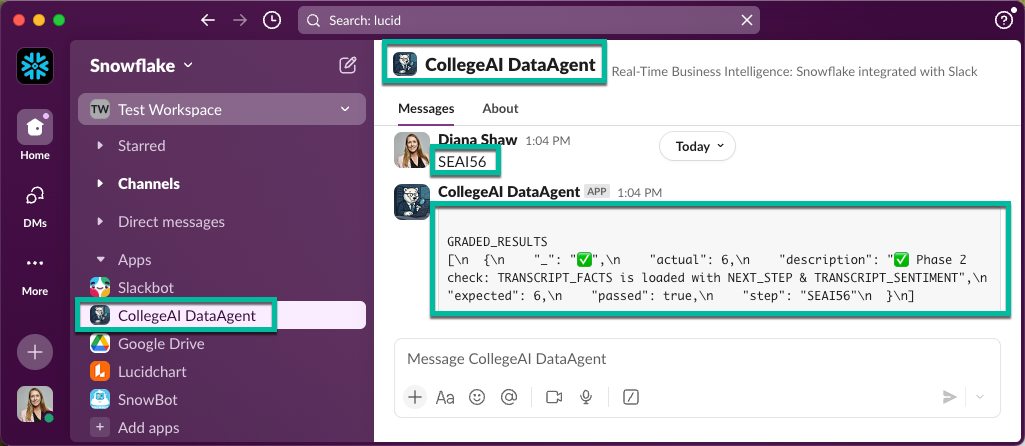
* **Save** your augmented semantic model
* Return to **VS Code**
* **Restart** your Slack app

****python3 slackapp.py

* ****In the **CollegeAI DataAgent** Slack channel, try a couple of Phase 2–style questions to sanity-check your new tables & fields:

"Which meetings for Delta Enterprises had low sentiment?"  
 "Show me Cortex-related meetings with a low risk score and demo next step."

#### **🛠️ DORA Evaluation #56: Phase 2 Completion Check**

To receive credit for Phase 2, invoke the built-in DORA check **directly in Slack**:

* Slash command:

/askcortex SEAI56

* Or simply type (no slask):

SEAI56

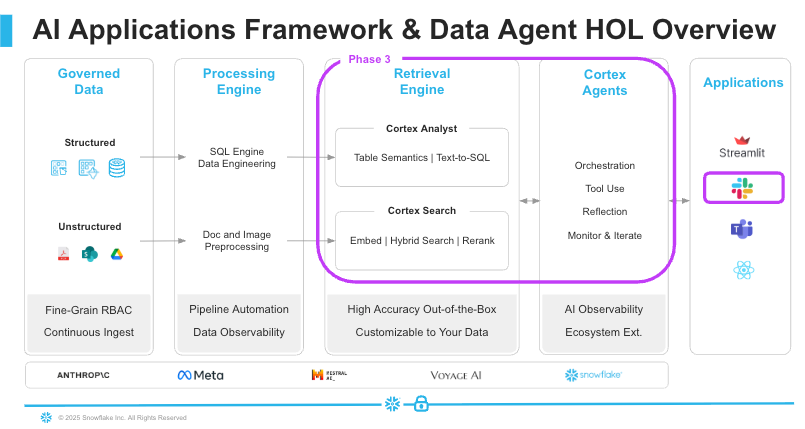
If everything is wired up correctly, you’ll get a green ✅ and something like:

*This step must be run in Slack (not in a Snowflake notebook).*

⚠️ You **must** pass SEAI56 before moving on to Phase 3.

## **Phase 3: Cortex Agent Orchestration**

In Phase 3, you’ll integrate everything into a fully orchestrated **Cortex Agent** that runs in Slack and intelligently selects the right tools to respond—whether it’s an **SQL query**, **a document insight**, or a **multi-hop reasoning task**.



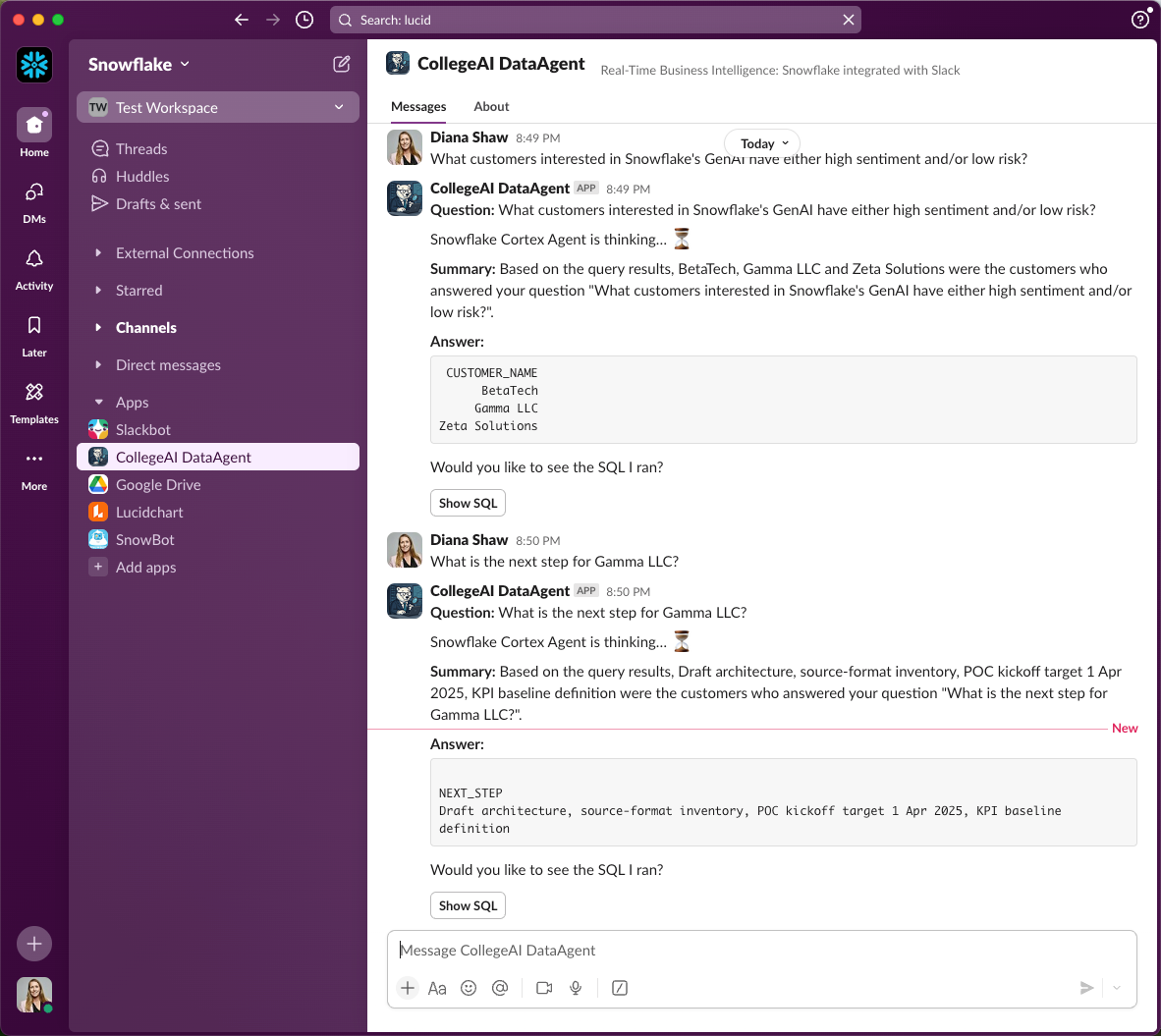
⚠️ **Agent Creation:** Today (May 2025), [**Cortex Agents**](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents) are created **only** via the [REST API](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents#creating-and-configuring-a-cortex-agent) (not CREATE CORTEX AGENT SQL). In this lab, your supplied Python script ([slackapp\_agent.py](https://drive.google.com/file/d/1pep32r7QOEY_TZj6t4nXyqy18uB9BsIF/view?usp=drive_link), ~400 lines) is the entry point—it handles Slack inputs, orchestrates Cortex Search vs. Analyst calls, executes SQL, and formats responses.

Instead of calling Cortex Search or Cortex Analyst directly, your agent will:

* **Understand what the user is asking**
* **Choose the right tool (or combination)**
* **Retrieve and/or query the data**
* **Summarize the result**
* **Respond in Slack with full transparency and auditability**

##### **What This Looks Like in Slack**

* **User**: "Which customers had high risk and low sentiment in the discovery transcript?"
* **Agent**: Runs **Cortex Search** → **Cortex Analyst** → SQL → returns:
  + Plain English summary
  + Tabular results
  + “Show SQL” button



### **🛠️ Step 3.1 Upgrade your Agent Code**

* **Use this updated** [slackapp\_agent.py](https://drive.google.com/file/d/1pep32r7QOEY_TZj6t4nXyqy18uB9BsIF/view?usp=drive_link) file (copy it into the same DATAAGENT folder that already holds your .env and key files).
* **Stop** the [slackapp.py](http://slackapp.py) bot that might still be running (Ctrl-C).
* **Start your updated slackapp\_agent.py bot**

source .venv/bin/activate # if not active

python3 slackapp\_agent.py



* You should see one “Connected to a new endpoint …” line.
* Consider asking these questions:
  + **Which customers have meeting transcripts?**
  + **What are the next steps for Acme Corp and Gamma LLC?**
  + **What customers interested in Snowflake's GenAI have either high sentiment and/or low risk?**
  + **What are the top key phrases from the BetaTech meeting?**
  + **Show a short recap of Zeta Solutions’ last meeting.**

##### **🔹 What’s inside the Cortex Agent slackapp\_agent.py code**

| **Component** | **Purpose & Why It Matters** |
| --- | --- |
| **Imports & Setup**  (1–21) | **Loads libraries** (Snowpark, Slack Bolt, requests, pandas), **suppresses warnings**, and **reads environment variables**. Establishes a consistent runtime environment, so downstream code has all dependencies and credentials ready. |
| **Auth & Config**  (22–72) | **Builds REST endpoints & semantic model URI**, loads the **PAT**, **JWT key**, and decrypts it. Ensures secure, authenticated access to Cortex Analyst and Snowflake—critical for every API call your agent makes. |
| **Snowpark Session**  (73–83) | **Initializes a persistent Snowpark session** for on-the-fly SQL execution. Avoids reconnect overhead on each request, boosting performance for repeated queries. |
| **DORA**  (84–104) | **Creates/merges into AGENT\_STATUS** to log check-ins. Enables monitoring and grading via SEAI57—so you can verify the agent’s uptime and reliability. |
| **Search Service Handle**  (105–170) | **Retrieves your TRANSCRIPTS\_SEARCH\_SERVICE** for unstructured data. Powers hybrid RAG queries—letting you seamlessly pull in transcript context when needed. |
| **Helpers**  (113–170) | Utility functions (\_english\_list, DataFrame summarizers, SQL hint injection). Standardizes result formatting and summaries, improving readability and consistency of responses. |
| **Core Logic: run\_agent()**  (171–300) | **Dynamic orchestration** ensures each question uses the right tool. **Decision Flow** – **automatically picks between SQL Path and RAG Path.** |
| **Slack Integration** (301-366) | Wires up Slack Bolt listeners for “hello”, /askcortex, messages, and the “Show SQL” button—bridging your Python logic with Slack to **deliver interactive, real-time analytics**. |

### **🔹 Step 3.2 Decision Walk-Through**

Rather than swapping out large code blocks, here’s **exactly where** run\_agent() chooses between **SQL Path** and **RAG Path**, and how it flows:

def run\_agent(question: str, customer: str) -> Dict[str,Any]:

# … helpers omitted for brevity …

# ── Decision point: SQL vs. RAG

**if re.search(r"\b(count|list|how many|average)\b", question, flags=re.I) \**

**or re.search(r"\bshow\s+(?:me|the|\d+)", question, flags=re.I): # (line 234)**

# … build sql\_req, call\_analyst, extract sql\_stmt, answer\_table, summary …

**return {"answer": answer\_table, "sql": sql\_stmt, "summary": summary}**

# ── Otherwise → RAG Path ──

# … gather context via search\_chunks …

**rag\_json = call\_analyst(rag\_req) # (line 280)**

# … post-processing of rag\_json follows …

* Line 234 is the if re.search(...) decision.
* Line ~247 is the return {"answer":…} inside the SQL branch.
* Line 280 is the rag\_json = call\_analyst(rag\_req) that kicks off the RAG path.

This shows exactly which lines in run\_agent() control the flow and what gets returned to Slack.

**Walk-Through Example:**

**SQL Path**

* **Question:** “Show me how many meetings we had.”
* **Trigger:** Question matches “count”, “how many”, or “show me” regex.
* **Flow:**
  + Build sql\_req → send to **Cortex Analyst**.
  + Analyst returns a SQL block (SELECT COUNT(\*) FROM CUSTOMER\_MEETINGS;)
  + Agent runssession.sql(sql\_stmt).to\_pandas()
  + Summarizes results via \_local\_short\_summary or \_llm\_question\_and\_answer.

**RAG Path**

* **Questions:** “What did Gamma LLC say about risk?”
* **Trigger:** Question doesn’t match the SQL regex.
* **Flow:**
  + ​​search\_chunks(...) retrieves relevant transcript snippets.
  + Build rag\_req embedding those snippets.
  + Send to **Cortex Analyst** for a hybrid answer.
  + Optionally execute returned SQL or return the free-text summary.

##### ***📌* How It Maps to Cortex-Agent Fundamentals**

* **Governance & audit** → **Show SQL button** + Heartbeat table  
  *Every answer is reproducible; every run is logged.*
* **Tool registry** → **Analyst** (structured) | **Search** (unstructured) | **Complete** (narration)  
  *run\_agent() automatically picks and chains multi-hop calls.*
* **Cost control → local summary** for small results **vs. Complete** only for large or complex outputs

*Minimizes unnecessary LLM usage and lowers spend.*

* **Verified SQL** → KEY\_PHRASE\_HINT pattern  
   *Injection anchors LLM to a known-good snippet, preventing drift and invalid joins.*
* **Planning & reasoning** → system prompt: “return one SQL block then explain in 1–2 sentences”

*Guides Agents’ output format.*

* **Observation & action** → session.sql(sql\_stmt).to\_pandas()  
  *Executes the plan and turns data into an observation that the Agent can summarise.*
* **Narration** → **local helpers** or **Cortex Complete**  
  *Balances cost vs. quality; large results trigger the LLM to write a real answer.*

#### **🛠️ DORA Evaluation #57: Final check**

Once your agent is live, test its ability to orchestrate Search + Analyst by typing **directly in Slack**:

* Slash command:

/askcortex SEAI57

* Or simply type (no slash):

SEAI57

You’ll receive a ✅ confirmation if all pieces are working. This completes the hands-on lab.

⚠️ You **must** pass SEAI57 to complete the Data Agents HOL.

### **🛠️ Optional Step 3.4: Bring Your Data Agent into Snowflake Intelligence**

After finishing the Slack-based Data Agent HOL, you can load the same **semantic model (YAML)** and **search service** into **Snowflake Intelligence (SI)**—a preview app that lets business users chat with “data agents” in a no-code UI. *(~25-30 min.)*

##### **🔹 What is Snowflake Intelligence?**

* **Data agents**: pre-built chat bots that read Snowflake data (and soon Salesforce) to surface insights, visuals, or actions.
* **Enterprise-ready UI**: a guided workspace for sales reps, execs, and marketers who used to rely on BI dashboards.
* **Powered by Cortex**: uses the **Cortex Search**, **Analyst**, and **LLM** functions you leveraged in this lab.

***📌* Snowflake Intelligence Helpful Links:**

* [Latest pitch deck](https://docs.google.com/presentation/d/1bxePyLwbmUXVj1xyEboAQltB-FaIBO0ndXM2OyKx4wY/edit?usp=sharing)
* [Demo video](https://youtu.be/va-l7sYp3OA)
* [PrPr Documentation](https://docs.snowflake.com/en/LIMITEDACCESS/snowflake-intelligence)
* [Enroll customers to PrPr](https://docs.google.com/forms/d/e/1FAIpQLSe9hZfbg8Kygtmz_14of-ow9mr6iKu6lXXuomWIaNvuimp6Yw/viewform)

***⚠️ Preview Status:*** *Workflows may shift before GA, but* ***semantic models****,* ***search services****, and* ***agents*** *will remain. This guide matches the product state heading into Summit 2025.*

**🚧 Prerequisite - enable SI flags on your demo account**:

Log in via [snowflake.okta.com](http://snowflake.okta.com), pick your deployment, then run:

ALTER ACCOUNT <YOUR ACCOUNT>

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';



After a minute you’ll see **Snowflake Intelligence** and **Agents** in the left nav.

**What you’ll do:**

1. Set default role/warehouse to AICOLLEGE (so SI launches with the right privileges).
2. Create the SNOWFLAKE\_INTELLIGENCE DB plus TEMP, CONFIG, and AGENTS schemas.
3. Import the semantic model file and attach the search service in the SI Agent wizard.
4. Chat with the agent and compare the SI vs. Slack experience.

#### **🛠️ Snowflake Intelligence Setup**

* Return to your **Snowflake Notebook** [**Data Agent Cortex Agent Setup Notebook**](https://drive.google.com/file/d/1Qso0th4-lZIpsPH19gFlLSpJ35GCgMTs/view?usp=drive_link)
* Jump to the **SnowflakeIntelligence** cell
* **Run** the necessary **SQL cells** to:
  + Alter user DEFAULT\_ROLE = AICOLLEGE and DEFAULT\_WAREHOUSE = AICOLLEGE
  + Create SNOWFLAKE\_INTELLIGENCE DB and its TEMP / CONFIG schemas.
  + Create the AGENTS schema, row-access policy, and AGENTS.CONFIG table.
* **Sign out / sign back** in so Snowsight reloads your defaults

Here’s the code to help:

-- Switch to ACCOUNTADMIN to grant necessary privileges

USE ROLE ACCOUNTADMIN;

-- Enable cross-region Cortex (important for Claude access)

ALTER ACCOUNT SET CORTEX\_ENABLED\_CROSS\_REGION = 'ANY\_REGION';

-- Set default role and warehouse for your user

ALTER USER <YOUR USER> SET DEFAULT\_ROLE = AICOLLEGE;

ALTER USER <YOUR USER> SET DEFAULT\_WAREHOUSE = AICOLLEGE;

-- Create required database and schemas

CREATE DATABASE IF NOT EXISTS SNOWFLAKE\_INTELLIGENCE;

GRANT IMPORTED PRIVILEGES ON DATABASE SNOWFLAKE TO ROLE AICOLLEGE;

-- Grant necessary privileges to AICOLLEGE role

GRANT USAGE ON DATABASE SNOWFLAKE\_INTELLIGENCE TO ROLE AICOLLEGE;

GRANT CREATE SCHEMA ON DATABASE SNOWFLAKE\_INTELLIGENCE TO ROLE AICOLLEGE;

USE ROLE AICOLLEGE;

USE DATABASE SNOWFLAKE\_INTELLIGENCE;

CREATE SCHEMA IF NOT EXISTS SNOWFLAKE\_INTELLIGENCE.AGENTS;

CREATE SCHEMA IF NOT EXISTS SNOWFLAKE\_INTELLIGENCE.TEMP;

CREATE SCHEMA IF NOT EXISTS SNOWFLAKE\_INTELLIGENCE.CONFIG;

GRANT CREATE STAGE ON SCHEMA SNOWFLAKE\_INTELLIGENCE.CONFIG TO ROLE AICOLLEGE;

GRANT CREATE STAGE ON SCHEMA SNOWFLAKE\_INTELLIGENCE.TEMP TO ROLE AICOLLEGE;

GRANT CREATE ROW ACCESS POLICY ON SCHEMA SNOWFLAKE\_INTELLIGENCE.AGENTS TO ROLE AICOLLEGE;

-- Create stage for semantic models

CREATE STAGE IF NOT EXISTS CONFIG.SEMANTIC\_MODELS

ENCRYPTION = (TYPE = 'SNOWFLAKE\_SSE');

-- Create row access policy for agents (CRITICAL FOR AGENT VISIBILITY)

CREATE ROW ACCESS POLICY IF NOT EXISTS SNOWFLAKE\_INTELLIGENCE.AGENTS.AGENT\_POLICY AS

(GRANTEE\_ROLES ARRAY)

RETURNS BOOLEAN ->

/\* allow if any of the listed roles is in your session \*/

ARRAY\_SIZE(

FILTER(

GRANTEE\_ROLES::ARRAY(VARCHAR),

r -> IS\_ROLE\_IN\_SESSION(r)

)

) > 0

COMMENT = 'Only allow rows shared with a role present in SESSION';

-- Create the necessary table for agent configuration

CREATE TABLE IF NOT EXISTS 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 minimum required privileges to AICOLLEGE role

GRANT USAGE ON DATABASE SNOWFLAKE\_INTELLIGENCE TO ROLE AICOLLEGE;

GRANT USAGE, CREATE TABLE ON SCHEMA SNOWFLAKE\_INTELLIGENCE.AGENTS TO ROLE AICOLLEGE;

GRANT USAGE ON SCHEMA SNOWFLAKE\_INTELLIGENCE.TEMP TO ROLE AICOLLEGE;

GRANT USAGE ON SCHEMA SNOWFLAKE\_INTELLIGENCE.CONFIG TO ROLE AICOLLEGE;

GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE SNOWFLAKE\_INTELLIGENCE.AGENTS.CONFIG TO ROLE AICOLLEGE;

GRANT SELECT ON TABLE SNOWFLAKE\_INTELLIGENCE.AGENTS.CONFIG TO ROLE PUBLIC;

GRANT USAGE ON APPLICATION SNOWFLAKE\_INTELLIGENCE

TO ROLE AICOLLEGE;

GRANT USAGE ON SCHEMA SNOWFLAKE\_INTELLIGENCE.TEMP TO ROLE PUBLIC;

GRANT USAGE ON SCHEMA SNOWFLAKE\_INTELLIGENCE.AGENTS TO ROLE PUBLIC;

ALTER USER <YOUR USER> SET DEFAULT\_NAMESPACE = SNOWFLAKE\_INTELLIGENCE.TEMP;

-- Switch to AICOLLEGE role

USE ROLE AICOLLEGE;



***📌*** Make sure you log out and then log back into Snowsight in order to use the updated default settings needed for Snowflake Intelligence.

**🛠️ Create the agent**

* Navigate to **AI & ML > Agents**
* Click **+ Create agent**
* Name: **DataAgent**
* **Agent Overview**:

DataAgent helps Sales Engineers and Account teams quickly mine insights from the six curated customer-call transcripts loaded in Snowflake. Ask it to:

- Summarize a single call or compare multiple customers

- Surface pain points, desired outcomes, or next-step action items

- Highlight which Snowflake features, Cortex services, or integration patterns were discussed

- Pull direct quotes or time stamps to support follow-up emails and QBR decks.

Think of DataAgent as your on-demand note-taker and discovery assistant, turning raw meeting text into concise, ready-to-share intelligence.



* **Response Instruction**:

Role & Objective – You are DataAgent, an expert sales-enablement assistant. Your job is to answer questions only from the six customer transcripts provided via the semantic model and search service.

Guidelines:

- Ground every answer in the transcripts; do not speculate beyond them.

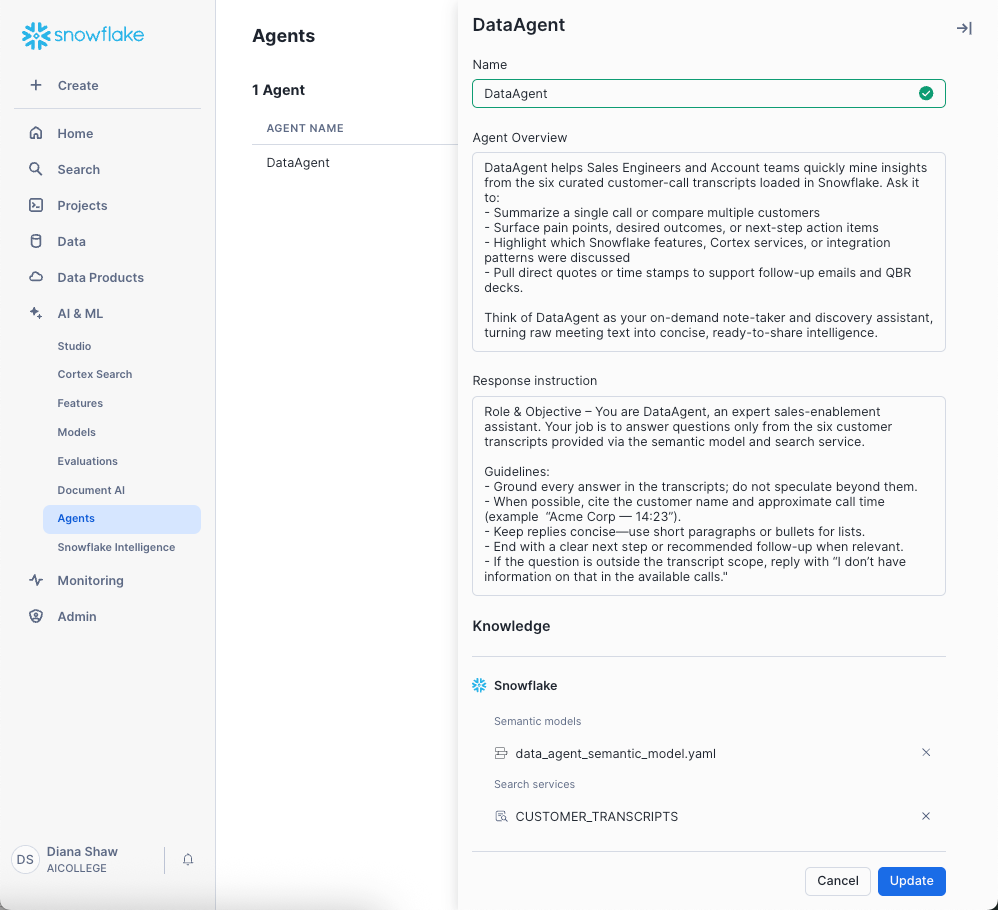
- When possible, cite the customer name and approximate call time (example “Acme Corp — 14:23”).

- Keep replies concise—use short paragraphs or bullets for lists.

- End with a clear next step or recommended follow-up when relevant.

- If the question is outside the transcript scope, reply with “I don’t have information on that in the available calls."

* **Knowledge section:**
  + **Link** to your **semantic model**: data\_agent\_semantic\_model.yaml found in AICOLLEGE.PUBLIC.TRANSCRIPTS
  + **Select Add**
  + **Link** to your **search service**: TRANSCRIPTS\_SEARCH\_SERVICE found in AICOLLEGE.PUBLIC. Add a Display name CUSTOMER\_TRANSCRIPTS and select a column that contains the document’s url (RELATIVE\_PATH)
  + **Select Add**
* **Select Update**



**🛠️ Test Snowflake Intelligence**

* **Navigate** to **AI & ML > Snowflake Intelligence**
* Try:
  + **Which customers have meeting transcripts?**
  + **What are the next steps for Acme Corp and Gamma LLC?**
  + **What customers interested in Snowflake's GenAI have either high sentiment and/or low risk?**
  + **What are the top key phrases from the BetaTech meeting?**
  + **Show a short recap of Zeta Solutions’ last meeting.**
* Decide which experience you prefer—**Snowflake Intelligence** or the **Slack integration**.

Optional Step

### **🛠️ Step 3.5: Data Agent HOL Clean-Up**

Once you’ve completed the hands-on lab, it’s a good idea to tear down temporary objects and revoke any elevated access you created.

1. **Shut Down the Slack Agent**

# In your terminal where the bot is running:

Ctrl-C

 • Ensures no stray processes remain active.

1. **Revoke or Rotate Secrets**
   1. Remove SLACK\_BOT\_TOKEN and SLACK\_APP\_TOKEN from your environment or rotate them in Slack.
   2. Revoke the SNOWFLAKE\_PAT and/or rotate your JWT key in Snowflake.
2. **Remove Data Agent Artifacts**

-- Stage file:  
REMOVE @AICOLLEGE.PUBLIC.SETUP/<your\_model>.yaml;

-- Semantic View (if created):  
DROP VIEW AICOLLEGE.PUBLIC.CORTEX\_MODEL\_VIEW;

-- Delete Search Service  
DROP SEARCH SERVICE AICOLLEGE.PUBLIC.TRANSCRIPTS\_SEARCH\_SERVICE;

-- Tear Down Supporting Tables  
DROP TABLE TRANSCRIPT\_FACTS;

DROP TABLE CUSTOMER\_MEETINGS;

DROP TABLE CUSTOMER\_INTERACTIONS;

DROP TABLE CUSTOMER\_INDUSTRY;

DROP TABLE CUSTOMER\_MEETING\_OUTCOMES;

DROP TABLE AICOLLEGE.PUBLIC.AGENT\_STATUS;

-- Review Access Controls  
-- Revoke any temporary roles or grants you created for the lab.  
-- Ensure no development-only RBAC policies remain active.

