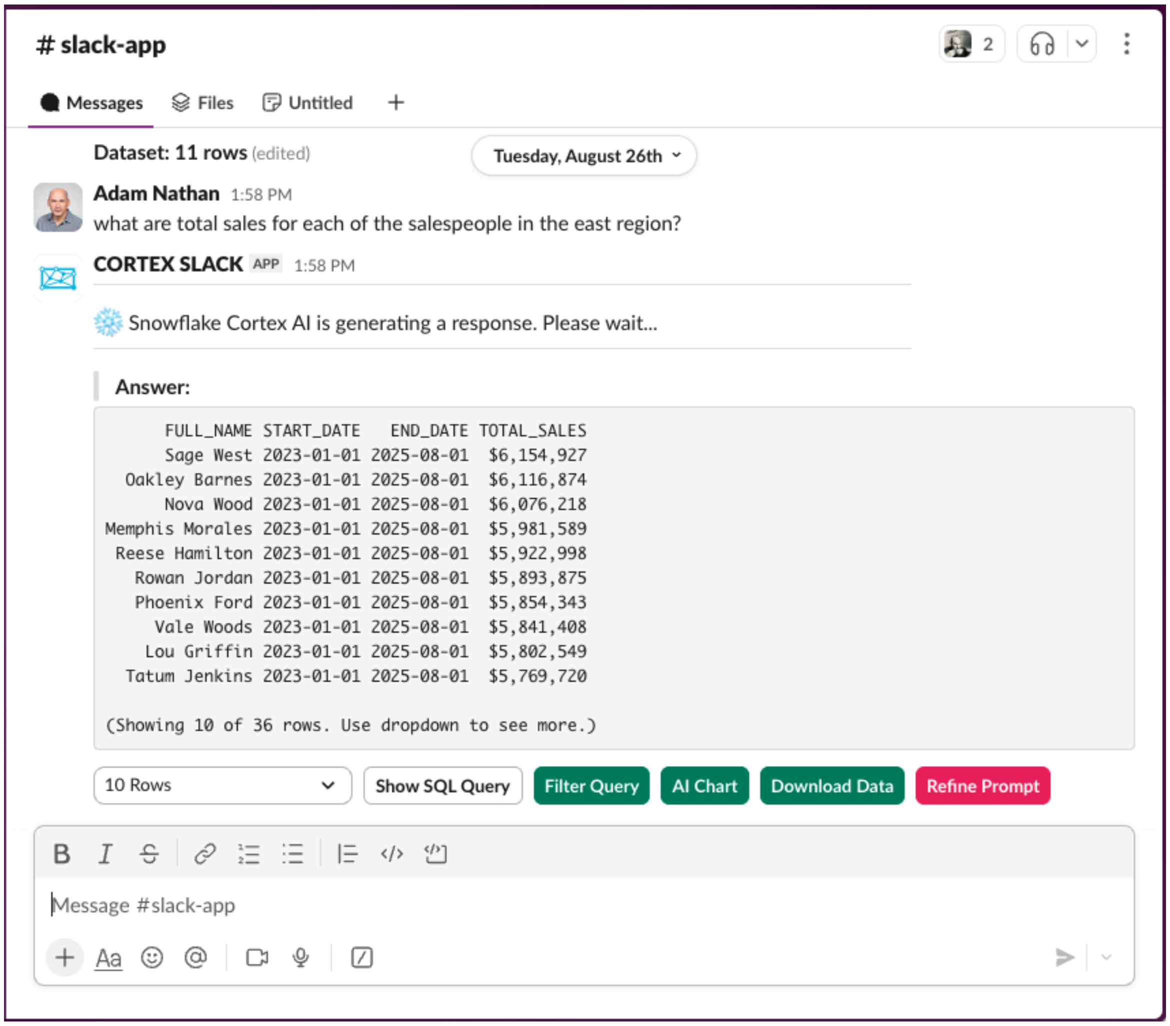
Quickstart for a Cortex Slack Integration



Dataset: 11 rows (edited)

Adam Nathan 1:58 PM
what are total sales for each of the salespeople in the east region?

CORTEX SLACK APP 1:58 PM

Snowflake Cortex AI is generating a response. Please wait...

Answer:

```
      FULL_NAME START_DATE   END_DATE TOTAL_SALES
      Sage West 2023-01-01 2025-08-01  $6,154,927
  Oakley Barnes 2023-01-01 2025-08-01  $6,116,874
      Nova Wood 2023-01-01 2025-08-01  $6,076,218
Memphis Morales 2023-01-01 2025-08-01  $5,981,589
 Reese Hamilton 2023-01-01 2025-08-01  $5,922,998
   Rowan Jordan 2023-01-01 2025-08-01  $5,893,875
   Phoenix Ford 2023-01-01 2025-08-01  $5,854,343
     Vale Woods 2023-01-01 2025-08-01  $5,841,408
     Lou Griffin 2023-01-01 2025-08-01  $5,802,549
  Tatum Jenkins 2023-01-01 2025-08-01  $5,769,720

(Showing 10 of 36 rows. Use dropdown to see more.)
```

v2.0

INTRODUCTION

A Demonstration Model for Conversational Analytics in Cortex

The Cortex Slack for Sales application is a comprehensive demonstration model that showcases how Slack can serve as an intelligent, desktop and mobile-accessible front-end for Snowflake's Cortex services. This solution addresses a fundamental business challenge across organizations: providing salespeople and their managers with intuitive access to quota performance, compensation details, and team insights that are typically locked away in complex systems or static reports.

The Business Problem: Addressing Universal Sales Analytics Challenges

Sales organizations consistently struggle with providing real-time access to performance data. Salespeople need immediate answers to questions like "How am I tracking against quota this month?" or "What's my commission structure for different deal types?" while managers require insights into team performance, territory trends, and compensation analysis. Traditional BI tools often require specialized training and aren't accessible on mobile devices where field sales teams spend most of their time.

This application demonstrates how conversational AI can transform these common pain points into natural interactions. Instead of navigating complex dashboards or waiting for IT support, users simply ask questions in Slack: "Show me my Q4 performance" or "Which of my reps are struggling with quota attainment?" The mobile-first approach ensures these insights are available anywhere, addressing the reality that modern sales teams work remotely and need data access beyond the office.

Core Architecture & Integration

The application uses the Cortex Agent API:

- **Cortex Search:** Retrieves insights from unstructured documents (e.g., a fictitious “Sales Guide 2025.pdf” in this demo) for policy and process questions.
- **Cortex Analyst:** Converts natural language to SQL queries for structured data analysis with automatic entitlement-based filtering.
- **Intelligent Cortex Agent API:** AI automatically chooses the appropriate Cortex service based on query context.

The Slack interface provides native mobile access to enterprise analytics, enabling sales teams to access insights, generate charts, and download data from anywhere. This mobile capability transforms how field sales teams interact with their performance data and business intelligence.

Every query is automatically filtered based on organizational hierarchy using recursive CTEs. The CRO sees all data, VP Sales sees regional data, managers see team data, and sales reps access only personal metrics. While the current implementation uses hardcoded user assignments for demonstration, it showcases the data model foundations for enterprise-grade row-level security.

A Novel Query Refinement System

The application introduces a new approach to improving user query quality through automated prompt analysis. When users submit questions, a background process evaluates the query's logical specificity and potential for better results using a custom `REFINE_QUERY` stored procedure built into this solution. This has the net effect of mitigating what might look like poor responses from inexact questions.

This system provides modal Proactive Refinement Suggestions:

- **Real-time Evaluation:** The stored procedure analyzes user prompts against the semantic model to identify gaps and oversights.
- **Smart Notifications:** Displays green checkmarks for well-formed queries or red refinement buttons for queries that could be improved.
- **Interactive Enhancement:** A modal-based refinement interface with AI-generated suggestions trains users to ask better analytical questions.
- **Educational Value:** Users see both their original query and suggested improvements, building analytical literacy over time.

This refinement system represents a paradigm shift from traditional BI tools that simply return results to an intelligent system that proactively helps users improve their analytical thinking and question formulation guiding them towards deterministic responses.

Advanced Visualization & Data Interaction

The system uses Claude-4-Sonnet to automatically select and configure optimal Plotly visualizations based on data characteristics and user intent.

Interactive Data Exploration features include:

- **Dynamic Filtering:** Complex filtering modals enable date ranges, categorical selections, numeric thresholds, and sorting without re-running queries.
- **Pandas-based Processing:** Fast in-memory filtering operations on cached results.
- **Multi-format Export:** CSV downloads with proper formatting and flexible display options.
- **Transparent Analytics:** Users can view generated SQL queries for learning and verification.

The Demonstration Data: A Synthetic Sales Org and Transactions

The application operates on a 100+ employee sales hierarchy with 5 organizational levels, realistic compensation structures, territory assignments, and monthly performance tracking. The semantic model includes a collection of product categories with authentic pricing and deal structures over a two-and-a-half-year period, providing a rich foundation for meaningful analytics demonstrations.

Rich metadata definitions with business synonyms, verified query patterns, and custom AI instructions enable natural language interactions like "Who are the top sales managers this quarter?" or "Show me Data Warehouse product trends."

Strategic Value & Future-Ready Foundation

This model successfully showcases conversational analytics workflows, AI service orchestration, security framework concepts, and user experience patterns that make data analysis accessible to non-technical users. The mobile accessibility through Slack represents a significant advancement in field-accessible business intelligence.

While architecturally sound, the application requires hardening for enterprise deployment including identity management integration, scalability optimization, comprehensive error handling, data governance controls, and production deployment architecture.

The application serves as an excellent reference implementation for organizations evaluating conversational analytics, planning Cortex adoption, or designing modern analytics architectures. The Query Refinement System alone represents an innovation in business intelligence, moving beyond simple query execution to active user education and analytical skill development - addressing the widespread challenge of data literacy in sales organizations.

This hands-on lab introduces Snowflake's Cortex Agent, demonstrating its value in streamlining data analysis. You'll explore Cortex Search for intelligent document review and Cortex Analyst for extracting insights from structured data. The lab focuses on integrating pre-written Python code, but requires setting up Snowflake and Slack trials, configuring a Slack app, and building out a Python environment.

Through this "final assembly" process and a focus on the underlying technology, you'll gain practical experience in how Cortex Agent combines these powerful functionalities, showcasing its ability to efficiently retrieve and synthesize information from diverse data sources, ultimately accelerating your data-driven decision-making.

SOLUTION ARCHITECTURE: CORTEX SLACK FOR SALES

System Overview

The Cortex Slack for Sales application demonstrates a conversational analytics architecture that transforms Slack into an intelligent data interface. The system orchestrates multiple Snowflake Cortex services through a Python-based “middle-tier”, enabling natural language data exploration with foundational security patterns.

Core Components

1. Slack Interface Layer

- Slack Bolt Framework: Handles real-time message processing, interactive components, and mobile-accessible UI
- Socket Mode Connection: Maintains persistent connection for instant response delivery
- Rich Block Kit UI: Supports tables, charts, modals, and file uploads within Slack conversations

2. Cortex Agent Orchestration

- Dual Agent Framework: Automatically routes queries between Cortex Search (unstructured documents) and Cortex Analyst (structured data)
- Cortex Agent Tool Selection: AI dynamically determines optimal service based on query context and semantic model coverage

3. Security & Access Control

- Hierarchical Row-Level Security: Recursive CTEs automatically filter all queries based on organizational hierarchy
- Service Account Authentication: Single Snowflake service account with JWT authentication handles all database connections
- Entitlement-Based Filtering: User email from Slack is passed through to query entitlement tables for access control

4. AI-Enhanced Analytics Pipeline

- Query Refinement Engine: Custom stored procedure analyzes user prompts against semantic models to suggest improvements

- Intelligent Visualization: Claude-4-Sonnet automatically selects optimal Plotly charts based on data characteristics. [1]
- Interactive Data Exploration: Pandas-based in-memory filtering enables complex data manipulation without re-querying

Data Architecture

Semantic Layer:

- Snowflake Semantic Model: YAML-based model defines business context, synonyms, and relationships for a ~100 employee sales organization. The model is supported by underlying SQL Views.
- Verified Query Patterns: Pre-validated analytical queries for common business scenarios

Security Model:

- Demonstration Entitlements: Hardcoded user assignments showcase 5-level hierarchy (CRO → VP → Regional Manager → Sales Manager → Sales Rep)
- Recursive Access Control: Each user sees only data within their organizational scope
- Email-Based Authorization: Slack user email addresses map to entitlement table records for access decisions

Technical Implementation

Application Stack:

- Python 3.9+: Core application runtime with Slack Bolt, Pandas, and Plotly
- Snowflake Integration: Native connector with Snowpark for advanced operations
- Modular Architecture: Separate modules for charting (charter.py), filtering (data_filter_modal.py), and Cortex integration (cortex_chat.py)

Key Workflows:

1. Query Processing: User message → Service selection → Cortex API → Response parsing → Security filtering → UI rendering
2. Chart Generation: Data analysis → AI visualization selection → Plotly configuration → Image rendering → Slack upload

[1] Special thanks to Branden Ciranni. Bugs from modifications to his code are this developer's.

3. Interactive Filtering: Modal creation → User input → Pandas operations → Result display → Cache management

Scalability & Production Considerations

Current Limitations:

- In-memory caching suitable for demonstration scale (~100 users)
- Single-instance deployment without load balancing or high availability
- Hardcoded user management requiring manual entitlement configuration
- Hierarchical relationships among sales levels require further updates
- The strength of Query Refinement is a function of the model. Experiment to find a balance between token cost and quality of results. This demonstration uses openai-gpt-5.

Production Readiness Requirements:

- Automated User Provisioning: Integration with HR systems to populate entitlement tables based on organizational changes
- Distributed Caching: Redis or database-backed session management
- Container Orchestration: Kubernetes deployment with auto-scaling capabilities
- Monitoring & Observability: Comprehensive logging, metrics, and alerting systems

This architecture serves as a reference implementation for organizations evaluating conversational analytics patterns, demonstrating how AI can democratize data access while maintaining enterprise security standards.

This solution is seen as a logical compliment to Snowflake Intelligence. The two serve similar functions but focus on different workflow needs, e.g. quick answers directly within Slack and mobile versus more in-depth analytical inquiry that is made possible through Snowflake Intelligence.

LAB PREREQUISITES

Here are the key skills and resource requirements a developer should possess prior to beginning this hands-on lab:

Skills

- **Command Line / Terminal Proficiency:** Ability to open and navigate a command line or terminal, and execute basic commands (e.g., cd, git clone, python, pip).
- **Python Fundamentals:** Some familiarity with Python syntax, the importing of libraries, understanding how to execute Python scripts, and the concept of virtual environments. Participants should already have **Python installed** on their machine. They will not be required to code.
- **SQL:** A basic understanding of SQL queries and the difference between structured and unstructured data.
- **Familiarity with LLMs and Generative AI:** A conceptual understanding of what Large Language Models (LLMs) are and how generative AI works.
- **Text Editor Proficiency:** Ability to open, edit, and save plain text files (like `.env` files).
- Slack: the user should have a basic familiarity with Slack to understand its basic function and have the ability to create a channel.
- **Git-hub integration**. You should be able to create a repository on your local machine

Tools

- A computer with python3 installed
- This lab requires the use of two trial licenses for free software: a Snowflake Trial and a Slack Trial. They are free and straightforward to acquire and configure. Instructions are included here.
- This guide is developed for Mac. The majority of the instructions here will apply to other environments as well, but you may need to look at slight variations in code to execute, e.g. the creation and activation of Python environments

BUILDING THE APPLICATION

The development of this lab contains the following steps:

1. **Get Your Snowflake Trial Account**
    a. Acquire a trial Snowflake account suitable for this lab.
2. **Clone a GitHub repository**
    a. Access and clone a GitHub repository to download the lab's application files
3. **Configure Python**
    a. Configure a Python virtual environment with required libraries
4. **Setup Key-Pair Authentication**
    a. Configure secured access with Snowflake
5. **Configuring Snowflake**
    a. Build out the database and schema objects to hold project files
6. **Load Semantic View**
    a. Upload a completed .yaml file, the semantic layer to describe the retail data.
7. **Load PDF Content & Create Cortex Search Service**
    a. Upload the unstructured PDFs that will be used for our Cortex search service
    b. Configure Cortex Search to parse, chunk and load the unstructured PDF data
8. **Build, Secure and Integrate Your Slack Application**
    a. Leverage a pre-configured manifest .json file to simplify Slack app creation.
9. **Testing Your Application End-to-End**
    a. Validate the end-to-end connectivity and functionality of the application
10. **Clean Up**
    a. If desired, the scripts to drop the lab elements

Get Your Snowflake Trial Account

⚠️ It is mandatory to use a Snowflake trial account for this lab. Not all Snowflake cloud providers and regions have the full set of LLM models and functionality available for the Cortex features you'll be using here. To ensure correct functioning of the solution, sign up for a Snowflake Trial below.

Additionally, a trial account lets you act as an ACCOUNTADMIN for your dedicated environment during the lab, simplifying setup with no security implications for your organization's production data.

The Snowflake trial comes with $400 of free credits and no further obligations besides providing an email address for activation.

Obtaining a Snowflake Trial

1. Navigate to the Snowflake trial signup page: https://signup.snowflake.com/
2. Fill in straightforward personal and company details.
3. ⚠️ On the next screen, for the configuration options, be sure to select EXACTLY:
    - **Region:** United States
    - **Cloud Provider:** Amazon Web Services (AWS)
    - **Edition:** Enterprise
4. Finally, click the **"Get started"** button.
5. Validate the trial through the email you will receive.
6. Log in to verify connectivity

Clone a GitHub Repository

Clone Repository and Set Up Python Environment

Next, you'll set up your local Python environment and clone the lab code from GitHub.

1. **Create a Project Directory:** Create a folder
2. **Clone the Repository:**
    a. On your machine, open a terminal or command prompt.
    b. Navigate to a designated folder where you want to store the lab files.
    c. Then, clone the lab repository using the following command:

None

```
git clone https://github.com/sfc-gh-anathan/cortex-slack-for-sales.git
```

3. **Validate** you have a new folder named **cortex-slack-for-sales** containing a full suite of project files.

Configure Python

Retrieve Your Account/Server URL

Once your Snowflake trial account is active, you'll need to retrieve your unique **Account/Server URL** from Snowsight, Snowflake's web interface. This URL is essential as it's how your Python bot will connect to your specific Snowflake instance.

1. In Snowsight, locate and click your **username** in the lower-left corner to open the user menu.
2. From the menu, navigate to the **Account** submenu.
3. Click on **View account details** next to your active account.
4. An **Account Details** page will open.
5. Find and make a note of these values
    a. **Your User Name**
    b. **Account Identifier**
        i. **ORGANIZATION_NAME-ACCOUNT_NAME**
    c. **Account/Server URL**
        i. **ORGANIZATION_NAME-ACCOUNT_NAME.snowflakecomputing.com**
    d. **Account Locator**
        i. WUT55555

Update Environment Variables Part I

There are two sets of environmental variables to configure.

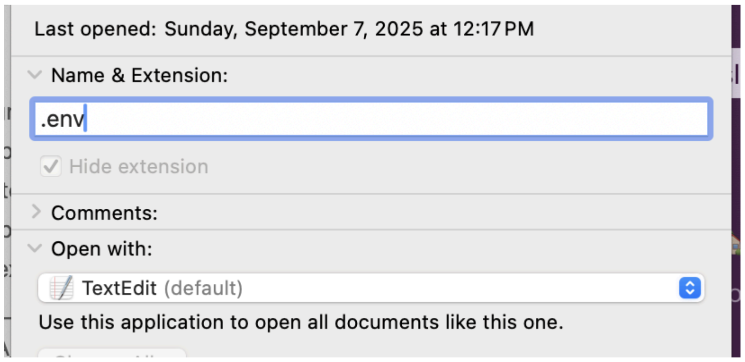
You will configure program variables to access Snowflake in Part I. (ROLE, WAREHOUSE, ACCOUNT…)

In Part II, further down, you will enter the variables to enable connection to Slack. (TOKENS)

To make this simple, you will find a fill-in-the-env.txt file in the repo directory. You will be updating values in that .txt file and then renaming the file and removing the extension.

1. Locate the **fill-in-the-env.txt** file in the root directory.
2. **Rename** this file to .env (There is no file name, only the ".env" extension)
3. Open the .env file in a text editor. You will receive a warning about a file name starting with a dot. Ignore.
    a. Validate that there is no extension (like .txt) that is hidden from view.
    b. To do this: right click on the file and validate that the Name & Extension has ONLY **.env**

4. Right click on the .env file and select “Get Info”) Validate you see the following for Name & Extension. If it is “.env.txt”, then delete the “.txt”


Last opened: Sunday, September 7, 2025 at 12:17 PM

Name & Extension:

.env

☑ Hide extension

Comments:

Open with:

TextEdit (default)

Use this application to open all documents like this one.

5. You will update the following placeholder values in the **.env** file with your specific details:
    a. **Snowflake Account Details:** Update
        - ACCOUNT = **Your Account Identifier**
            - Enter it in the pattern displayed in the **.env** file
        - HOST = **Account/Server URL**
            - Account/Server URL
        - AGENT_ENDPOINT
            - Add **Your Account Locator** before the “.us-east-1”
        - DEMO_USER = the name you created to log in to the Snowflake Trial
        - DEMO_USER_ROLE = ACCOUNTADMIN
        - SLACK TOKENS: Leave them as placeholders for now. We’ll edit later
6. You .env should look like this pattern:

ACCOUNT=**ORGANIZATION_NAME-ACCOUNT_NAME**
HOST=**ORGANIZATION_NAME-ACCOUNT_NAME**.snowflakecomputing.com
AGENT_ENDPOINT=https://**WUB55555**.us-east-1.snowflakecomputing.com/api/v2/cortex/agent:run
DEMO_USER=**YOUR_USER_NAME**
DEMO_USER_ROLE=**ACCOUNTADMIN**

Set Up Python Virtual Environment

1. Open a Terminal
2. Ensure your system has python3 installed before proceeding (see code below)
3. In the **cortex-slack-for-sales** directory, execute the commands below. They will create an isolated Python environment, install all necessary libraries, while preventing conflicts with other Python projects on your machine.
4. *Note: These instructions are for running the hands-on lab on a Mac. The source .venv/bin/activate command might differ slightly for Windows users (e.g., .\.venv\Scripts\activate).*

None

```
# Verify you are running Python3. (You will see a version number if you are.)

python3 --version

# Create a virtual python3 environment named .venv

cd cortex-slack-for-sales

python3 -m venv .venv

# Activate the virtual environment

source .venv/bin/activate

# Verify that you are using the virtual environment's Python

which python3

# you should see: cortex-slack-for-sales/.venv/bin/python3

# Install the required Python libraries

pip install -r requirements.txt
```

Setup Key-Pair Authentication

You will generate the RSA public and private key pair required for securely authenticating your Python application with Snowflake. These keys will be stored directly in the root directory of your project (the `cortex-slack-for-sales` folder).

1. **Generate Private Key:** Open your terminal or command prompt (ensure you are still in your activated virtual environment within the project root directory). Execute the following command to generate your private key (`rsa_key.p8`). This command generates a 2048-bit RSA private key and saves it as `rsa_key.p8`. The `-nocrypt` option is used for simplicity in this lab to avoid password protection on the private key. In a production environment, you would typically encrypt this file with a passphrase.

```
None
openssl genrsa 2048 | openssl pkcs8 -topk8 -inform PEM -out rsa_key.p8 -nocrypt
```

2. **Generate Public Key:** Now, use the private key to generate its corresponding public key **rsa_key.pub**. This command extracts the public key from your newly created private key and saves it in a file named **rsa_key.pub**. You will need the content of this public key file for the next step in Snowflake.

```
None
openssl rsa -in rsa_key.p8 -pubout -out rsa_key.pub
```

3. **Validate file creation:** After executing these commands, you should see **rsa_key.p8** and **rsa_key.pub** files in your project's root directory.

Configuring Snowflake

This section guides you through the essential configurations within your Snowflake trial account to enable the lab's functionalities.

You will set up secure key pair authentication for your Snowflake user, create foundational database objects and stages, and then load the necessary semantic model and PDF documents.

Finally, you will deploy a Cortex Search Service, allowing your application to leverage Snowflake's AI capabilities for both structured data analysis and unstructured document review.

Install Your Public Key into Snowflake

To allow your Python bot to securely connect to Snowflake, you'll associate the public key you generated with your Snowflake user.

1. Open a SQL worksheet in Snowsight
    a. You can find Worksheets in the left hand pane:
        - **Command Line Icon** >> **Project** >> **Worksheets**
2. Open your **rsa_key.pub** file in a plain text editor. This is the one you created earlier.
    a. The file will be in the root of your python project
3. Carefully copy **only the multi-line string of characters** that sits *between* the -----BEGIN PUBLIC KEY----- and -----END PUBLIC KEY----- lines.
4. Paste this content *between the single quotes* in the RSA_PUBLIC_KEY parameter of the ALTER USER statement below.
5. Remember to replace **YOUR_USER_NAME** with your actual Snowflake username in both the **ALTER** statement and the **SELECT** statement (this will match your **DEMO_USER** in .env and is also the one you use to login to Snowflake..
6. Execute the following SQL command in Snowsight to associate the key to a user
7. Run the **SHOW USERS** and **SELECT** statements. You should see **'true'**.

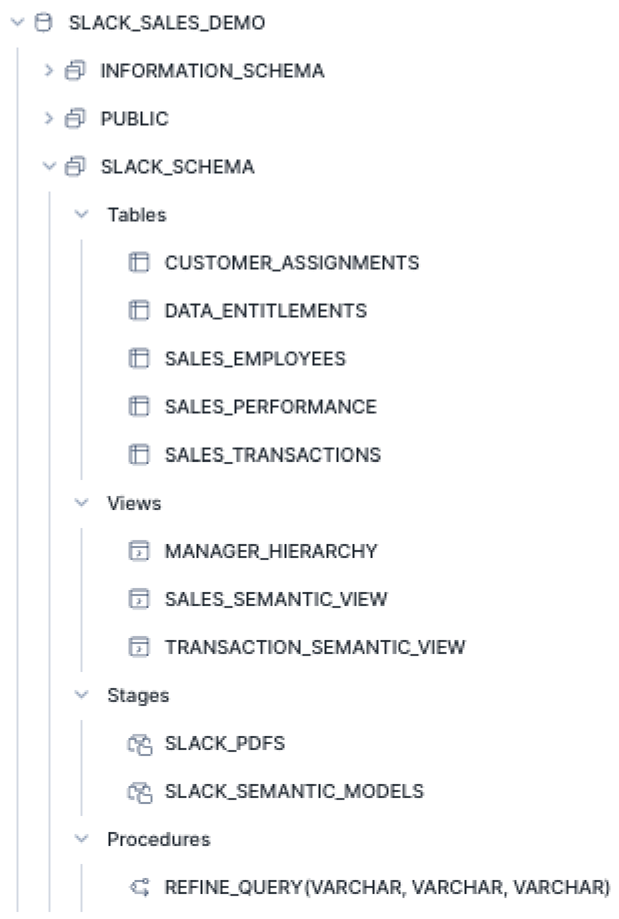
```
None
ALTER USER YOUR_USER_NAME SET RSA_PUBLIC_KEY =
'YOUR_BASE64_PUBLIC_KEY_CONTENT_GOES_HERE';


SHOW USERS; -- this must be executed directly before the SELECT statement below


SELECT "has_rsa_public_key" FROM TABLE(RESULT_SCAN(LAST_QUERY_ID())) where
"name" = 'YOUR_USER_NAME';
```

Create Foundational Objects

Now, you will set up the basic database, schema, warehouses, and stages required for the lab. The provided SQL will create these objects and select them for use.

1. 01-setup.sql
2. 02_sales_hierarchy_setup.sql
3. 03_sample_sales_data.sql
4. 04_sales_semantic_views.sql
5. 05_refine_query_procedure.sql

Visually confirm that you see the following in the database SLACK_SALES_DEMO:



```
SLACK_SALES_DEMO
  INFORMATION_SCHEMA
  PUBLIC
  SLACK_SCHEMA
    Tables
      CUSTOMER_ASSIGNMENTS
      DATA_ENTITLEMENTS
      SALES_EMPLOYEES
      SALES_PERFORMANCE
      SALES_TRANSACTIONS
    Views
      MANAGER_HIERARCHY
      SALES_SEMANTIC_VIEW
      TRANSACTION_SEMANTIC_VIEW
    Stages
      SLACK_PDFS
      SLACK_SEMANTIC_MODELS
    Procedures
      REFINE_QUERY(VARCHAR, VARCHAR, VARCHAR)
```

Familiarize yourself with the tables and views.

Load Semantic View

The semantic model (**sales_semantic_model.yaml**) is crucial for Cortex Analyst to understand your structured data in natural language queries.

- **Upload Semantic Model:**

  1. In Snowsight, navigate to
     - **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA >> Stages** >> **SLACK_SEMANTIC_MODELS**.
  2. Click the **"+ Files"** button (or similar upload icon) to upload files.
  3. Select the **sales_semantic_model.yaml** file from your project folder and **Upload**.
  4. Visually confirm you see the **sales_semantic_model.yaml** file in the stage.

Load PDF Content & Create Cortex Search Service

For Cortex Search to review documents, you'll need to load a sample PDF document into the **SLACK_PDFS** Snowflake stage.

This document will be parsed and chunked by the Snowflake Cortex service. This will first extract the text from the PDFs and then create small chunks of text that are necessary for effective embedding, the underlying process of Cortex Search.

Finally, you'll initiate a Cortex Search Service to enable high speed retrieval of the text within the PDFs.

- **Upload PDF Documents (Using the Stage Interface):**

    1. In Snowsight, navigate to **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA >> Stages >> SLACK_PDFS**.
    2. Click the **"+ Files"** button (or similar upload icon) to upload files.
    3. Navigate to the **Data** folder within the root of your project directory and select any PDF files there. **Upload** them to the stage.
- **Parse and Chunk PDFs:** Now, cut, paste and run all SQL commands in the **06_cortex_search_service.sql** file in the root of the project directory. This SQL will use Cortex functions to load the Cortex Search service with your PDF data.
    1. NOTE: THIS MAY TAKE FROM 2 - 5 MINUTES
- **Validate Creation of Chunked PDFs:**
    1. In Snowsight, navigate to **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA** >> **Tables**. Visually confirm that the **PARSE_PDFS** and **PARSED_PDFS** tables have been created and contain data
    2. Examine the content within each by selecting the table name and then selecting "**Data Preview**" in the header menu for the table.
- **Create Cortex Search Service**
    1. This step deployed the Cortex Search service, allowing your Cortex Agent to perform intelligent searches over your loaded PDF content.
- **Validate**
    - Navigate to **Database Icon >> Databases >> SLACK_DEMO >> SLACK_SCHEMA** >> **Cortex Search Services**.
    - Visually confirm that **INFO_SEARCH** is listed
    - Select **INFO_SEARCH**, then "View Details" at upper right
    - Verify the status is "active" for both Serving and Indexing.
- **Explore**
    - Select "Data Preview" and note how the page content is broken into chunks and the title of the source document is listed along with the file's path. At far right, you'll see an "embedding" (not human readable).
    - Select **Playground** at upper right. Enter "How are commissions calculated?"

Build, Secure and Integrate Your Slack Application

This section guides you through integrating your bot with Slack. You will create a dedicated Slack channel, provision your Slack App using a manifest file with pre-configured settings for application creation, install the application into your workspace, and retrieve the essential API tokens.

Finally, you'll add (“invite”) your bot to the Slack channel you have created and perform a quick check to confirm the integration is successful.

Create Your Slack Trial Account

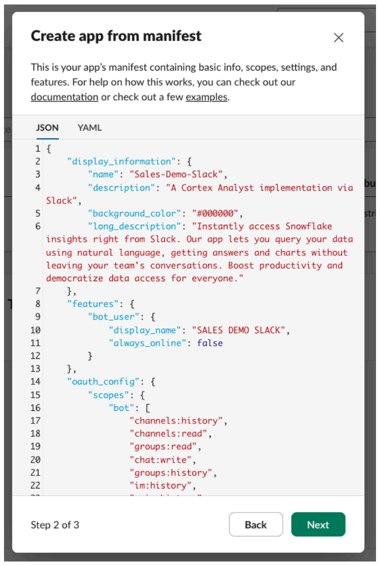
To set up your Slack trial account:

1. Go to **https://slack.com/get-started#/createnew**.
2. Follow the prompts to create a new Free Trial Slack workspace.

Create Your Slack App from a Manifest

You'll create the Slack App using a provided manifest.json file, which pre-configures your application's features and permissions.

1. Go to the Slack API dashboard: https://api.slack.com/apps.
2. Click the **"Create New App"** button.
3. Select **"From a manifest"**.
4. Choose the **workspace** where you want to install this app (your trial workspace). Then click **Next**.
5. Open your **manifest.json** file
6. in the root of your project folder in a text editor. **Copy its entire content** and paste it into the provided JSON editor on the Slack API page.



This is your app's manifest containing basic info, scopes, settings, and features. For help on how this works, you can check out our documentation or check out a few examples.

```json
{
    "display_information": {
        "name": "Sales-Demo-Slack",
        "description": "A Cortex Analyst implementation via Slack",
        "background_color": "#000000",
        "long_description": "Instantly access Snowflake insights right from Slack. Our app lets you query your data using natural language, getting answers and charts without leaving your team's conversations. Boost productivity and democratize data access for everyone."
    },
    "features": {
        "bot_user": {
            "display_name": "SALES DEMO SLACK",
            "always_online": false
        }
    },
    "oauth_config": {
        "scopes": {
            "bot": [
                "channels:history",
                "channels:read",
                "groups:read",
                "chat:write",
                "groups:history",
                "im:history",
```

7. Click **"Next"**, then **"Create"** to finish creating the app.
8. You will navigate to a "**Basic Information**" page for the application itself.

Install the App and Retrieve Tokens

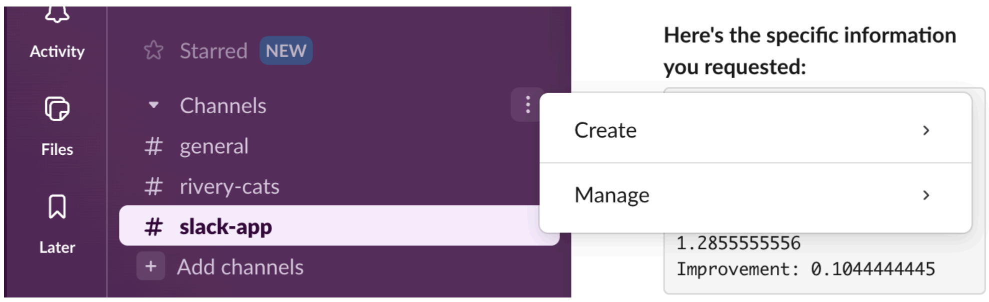
After creating the app, you need to install it to your workspace and obtain the necessary authentication tokens. Here's how to find the two tokens that are critical for your Python middle-tier code to connect to Slack

1. On the app's settings page, navigate to **Install App** >> **Install to Workspace** in the left sidebar.
2. Review the requested permissions and click **"Allow"**.
3. A Bot User OAuth Token will be displayed. Copy it
4. Upon successful installation, you will be redirected to the "OAuth & Permissions" page. Here in **Settings >> Install App**, you'll find your **Bot User OAuth Token** (starts with xoxb-)
5. Open your .env file and paste it after **SLACK_BOT_TOKEN**
6. **In Slack navigate to Settings >> Basic Information >> Display Information**
    a. Upload company-logo.jpg for your App icon
7. In **Settings >> Basic Information >> App-Level Tokens**
    a. Select "**Generate Token and Scopes**"
    b. Call your Token Name "slack-app-token"
    c. Add a "Scope" of **connections:write**
    d. Select **Generate**
8. Copy the Token (will start with 'xapp') and paste into your .**env** file for **SLACK_APP_TOKEN**

Create the Lab Channel

First, create a new public channel in your Slack workspace where your bot will interact.

1. In your Slack workspace, select the "**Home**" icon at the top of the left-hand menu bar
2. Select the dropdown for the **Channels section header**
3. Select "**Create**" and then **"Create a channel"**.



```
1.2855555556
Improvement: 0.1044444445
```

4. Name the channel "<**YOUR COMPANY_NAME> SLACK**" and ensure it is set to **Public**.
5. Click **"Create"**.

You can skip inviting others for now.

Invite the Bot to Your Channel

Even after installation, the bot needs to be explicitly invited into channels where you want it to participate.

1. Go to your newly created “**<COMPANY_NAME> SLACK”** channel in Slack.
2. In the Slack message box, type “**/invite @COMPANY_NAME> Slack”**.
3. Select your bot from the autocomplete list and press Enter.

Testing Your Application End-to-End

Test Slack Connection

This step verifies that your Python environment can successfully connect to your Slack App via Socket Mode and receive/send messages. First we need to add tokens in Python so that it can secretly connect to Slack. These are stored in the .env file. You saw them earlier when setting values in the file.

Activate the Python environment (if it is not already)

Open your terminal and activate the virtual machine

```
# Activate the virtual environment


source .venv/bin/activate
```

Execute the Slack Connection

1. Execute the test-slack-connection.py script. You'll find it in the root of the project directory.

```
None
python test-slack-connection.py
```

2. You should see a debug section and the values for your two tokens
3. This section will be followed by **"⚡ Bolt app is running!"**
4. Success

Validate Connectivity in Slack

1. Make sure you see output in your terminal indicating that the “⚡️ `Bolt` app is running!”
2. Go to your **#slack-bot** channel in Slack.
3. Type “hello” (exact case)
4. Your bot should respond with: "**Hey there @your_username**!"

If your bot responds, your Slack integration is successful, and your environment is ready to proceed with the main lab application!

If you are not successful:

1. Double-check your .env tokens for accuracy
2. Validate that the tokens were visible after running test-slack-connection.ppy.
    a. If no, it may be that the .env file cannot be accessed because it has a hidden extension like “.txt” Remove this through the File Info context menu.
3. Make sure you’ve added the application to the channel.
4. Make sure that @xxxx SLACK has been invited to the channel.

Running the Solution

Now that your environment is fully configured, it's time to launch the application and begin exploring its capabilities.

This solution showcases the power of Snowflake's Cortex Agent in combining structured data analysis with unstructured document review. You'll interact with the bot in natural language, asking questions that leverage **Cortex Analyst** to generate insights and potentially various graph types from structured data, or tapping into **Cortex Search** to retrieve answers directly from your loaded PDF documents.

1. **Launch the Application:** Ensure your Python virtual environment is still activated in your terminal (if not, run source .venv/bin/activate). Then, execute the app.py script:

```
None
python app.py
```

2. As above, you should see output in your terminal indicating that the "⚡`Bolt` app is running!"

Explore the Application in your Slack Channel

Here’s a sequence of questions and actions to guide you through an exploratory first pass.

1. Prompt: “How is a salesperson’s quota calculated?”
    a. You should see an answer based on the Sales Guide 2025 pdf, including a citation and a source
2. Prompt: “Stack rank all salespeople in the East Region. Include their territory.”
    a. 36 rows should be returned of which only 10 are displayed.
    b. Set the Row filter for 25 rows.
    c. Set the Filter Query for the Territory of New York
    d. At the bottom of Filter Query select Clear All Filters
3. Prompt: “What are average sales per month in 2024 for each region? Show results by month.”
    a. Select View AI Chart
    b. Select Download Data
4. Prompt: “Which Sales Manager had the lowest % of employees that met their quota?”
    a. Select “Show SQL Query”
5. Prompt: “Who is the best salesperson?”
    a. Note the answer is based on Total Sales, but this is one of many interpretations of the prompt
6. When Refine Prompt appears, enter refinements in text box and resubmit
    a. Make modifications per the feedback.
    b. A “Prompt satisfactory” notification icon should appear

Clean Up

To clean up the trial environment, run 99_**cleanup.sql**.

This script will drop the **SLACK_DB** database and the **SLACK_S** warehouse created for this project.

Key Documents & Resources

Here are the key documents and their specific URLs that would be most helpful for participants during this lab:

1. **The Lab's GitHub Repository:** This is your primary source for all the project code, manifest files, semantic model, and sample data.
    - **URL:** https://github.com/sfc-gh-anathan/cortex-slack-for-sales
2. **Snowflake Cortex Documentation:** For a deeper understanding of Snowflake's built-in AI capabilities, including Cortex Agent, Cortex Search, and Cortex Analyst functions.
    - **URL:** https://docs.snowflake.com/en/developer-guide/cortex/
3. **Slack Bolt for Python Documentation (Getting Started):** This resource is excellent for understanding how the Python bot interacts with the Slack API, especially regarding event handling and basic bot functionality.
    - **URL:** https://tools.slack.dev/bolt-python/getting-started/