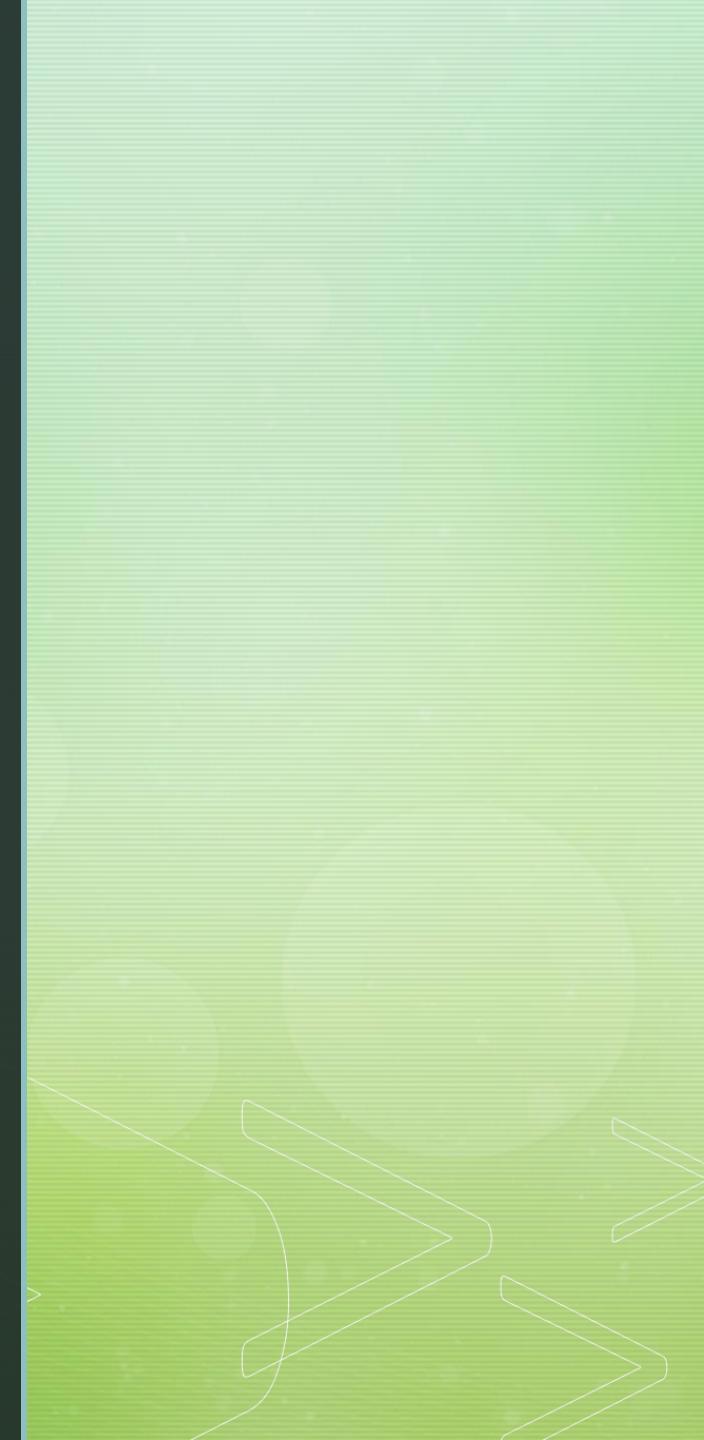


Top 10 Snowflake Integrations with ChatGPT

Design and Implement
Snowflake
Data Applications
with ChatGPT 4





App #1

Configure ChatGPT as a Coding Assistant for Snowflake in VSCode

- Create (Free) Snowflake and ChatGPT Accounts
- Configure Open-Sourced GitHub Project for the Course
- Configure VSCode with Snowflake and ChatGPT Plugins



App #2

Generate Snowflake Sample Databases with ChatGPT from VSCode

- Generate DDL Statements for Database and Tables
- Generate Synthetic Data - in Python and SQL
- Generate Data from Web Scraping
- Generate Table-Based SQL Queries



App #3

Snowflake Metadata Inspector in Natural Language

- Q&A to Chat Interfaces with ChatGPT
- Local Streamlit Web Apps

Interactive Data Analysis with ChatGPT Bot Agent

App #4

- Get Public Datasets from Snowflake Marketplace
- Metadata-Based Query Generator
- ChatGPT Bot for Data Analysis



App #5

Instant Charts with the Advanced Data Analysis Plugin

- EDA with a ChatGPT Plus Subscription
- *Generate Data Analysis Charts from Uploaded File*
- *Generate Data Science and ML Experiments*



App #6

Generate a Usage Monitoring Dashboard for Snowflake Account

- Generate Queries and Related Charts
- As Single and Multi-Page Streamlit Web Apps



App #7

Data Enrichment with an External Integration of ChatGPT

- ChatGPT through External Access Integration
- Data Enrichment with Select Queries
- Streamlit in Snowflake App w/ Select Query
- ChatGPT through External Functions (Obsolete)
- ChatGPT through Container Services (Coming Up)

ChatGPT with LlamaIndex on Personal Documents

App #8

- Library using Retrieval-Augmented Generation (RAG)
- Collect Personal and Custom Content
- Create Vector Store with Vector Embeddings
- Query Indexed Content and Combine w/ ChatGPT



App #9

ChatGPT SQL Generator with LangChain

- Framework to Help Build LLM Apps
- SQL Query Generation from Table Metadata (and Data)
- Jupyter Notebook Experiment
- As Local Streamlit Web App
- Deployed as a Streamlit Community Cloud App



App #10

Snowflake Query Analyzer and Optimizer

- Query Analysis with ChatGPT in VSCode
- As Local Streamlit Web App
- As Streamlit in Snowflake App with a UDF

Top 10 Snowflake Integrations with ChatGPT





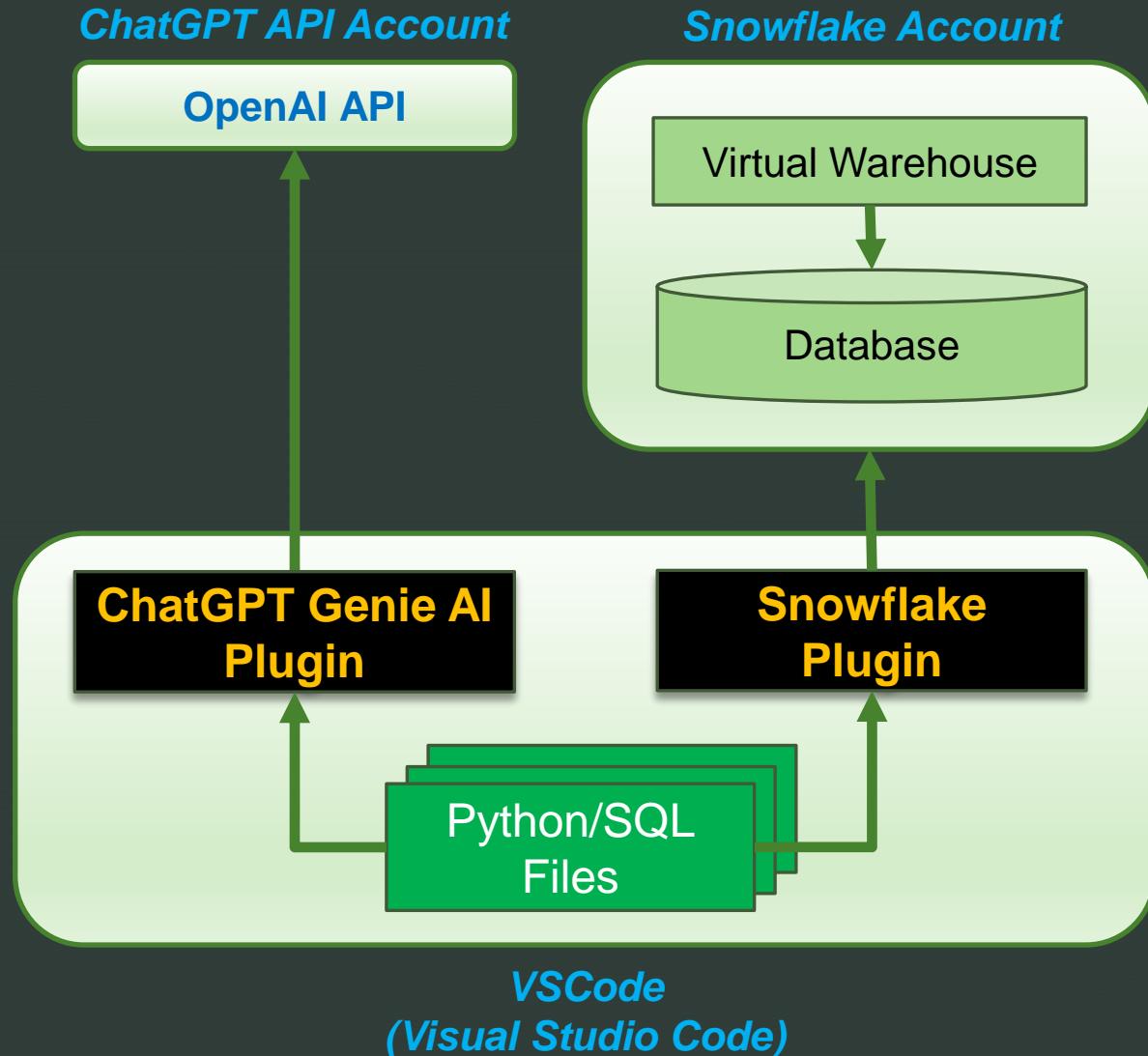
How to Benefit from this Course

- App #: Introduction + Hands-On + Review
- Free Snowflake and ChatGPT Accounts (see exceptions)
- Open-Sourced GitHub Repository w/ all SQL+Python Code
- Optional Final Quiz to Test Your Knowledge + Resources
- Captions + Playback Speed
- Q&A + Money Back Guarantee
- Reviews

App #1: Configure ChatGPT as a Coding Assistant for Snowflake in VSCode

- *Introduction to Snowflake and ChatGPT*
- Configure VSCode for our GitHub Project
- Create a Free Trial Snowflake Account
- Create ChatGPT API and ChatGPT Plus Accounts
- Install Snowflake and ChatGPT VSCode Plugins
- *Review of Snowflake and ChatGPT*

Architecture: VSCode w/ Plugins



Snowflake Free Trial Account

- *signup.snowflake.com*
- max \$400 free credits
- max 30 days
- no CC required
- edition: Standard/**Enterprise**
- provider: **AWS/Azure**
- basic authN: username + password

Snowflake Minimal Knowledge

- Snowsight Web UI - w/ SQL Worksheets
- CSV files - uploaded in tables
- SQL - DDL/DML/DQL
- Python Connector / Snowpark Session
- Streamlit Web Apps - local/deployed (in Streamlit Cloud)
- Streamlit in Snowflake (SiS) Apps - no Native Apps
- Information Schema / Account Usage
- External Function / External Access Integration

ChatGPT Subscriptions

- **ChatGPT API** (*openai.com*) - \$5 free trial/3mo, then PAYG
 - possible rate limiters - in TPM (tokens per minute)
- **ChatGPT Plus** (*chat.openai.com*) - \$20/mo
 - **GPT-4** - with DALL-E, browsing, **analysis**, max 40 msg/3h
 - GPT-3.5 - free
- **Copilot** - ~ChatGPT from Bing Chat ← not used!

ChatGPT Models

- **GPT-4 Turbo** (*gpt-4-1106-preview*) - latest, recommended
- **GPT-4 Turbo with Vision** (*gpt-4-vision-preview*) - w/ DALL-E
- **GPT-4** (*gpt-4*) - better than 3.5
- **GPT-3.5** (*gpt-3.5-turbo*) - for natural language, to generate text/code
- **DALL-E** - to generate and edit images given a natural language prompt
- **TTS (Text-to-Speech)** - to convert text into natural sounding spoken audio
- **Whisper** - to convert audio into text
- **Embeddings** - to convert text into a numerical form
- **Moderation** - to detect whether text may be sensitive or unsafe

Snowflake VSCode Plugin

- save account connection parameters
- object browser
- SQL statement/script execution → query results/history
- Intellisense
- LIST/GET/PUT stage files

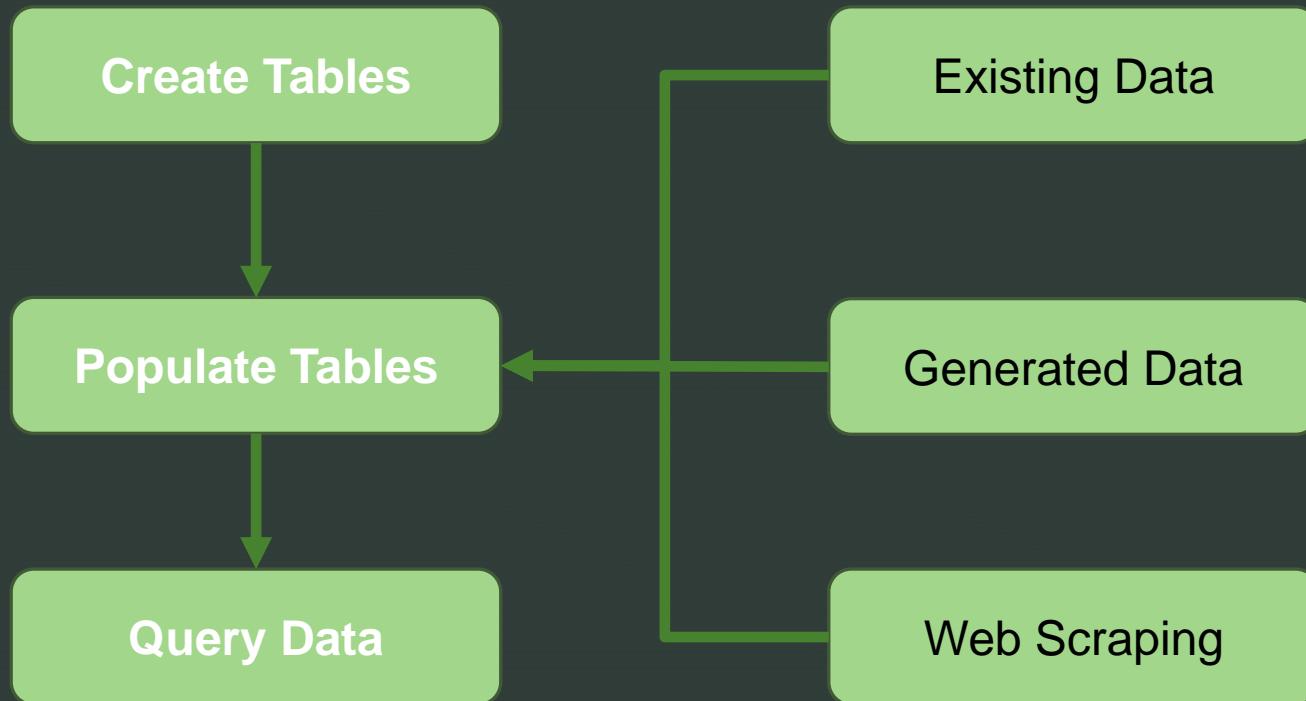
ChatGPT Genie AI VSCode Plugin

- enter OpenAI API key + always select GPT-4 Turbo model
- one-click copy/move generated SQL/Python code
- Genie - Explain / Find bugs / Optimize
- Genie - Add Tests / Add Comments / Complete code
- suggested fix in Problems window (Quick Fix)
- difference highlighting
- generate commit message
- generate Snowflake interview questions, summarize API...

App #2: Generate Snowflake Sample Databases with ChatGPT from VSCode

- *Introduction to Snowflake Sample Data*
- Generate and Run Sample DDL Script
- Generate Fake but Realistic Data in Python
- Generate Synthetic Data in SQL
- Generate and Run Python Code for Web Data Scraping
- Generate SQL Queries for Specific Tables
- *Review of Snowflake Sample Data*

Data Pipeline Workflow



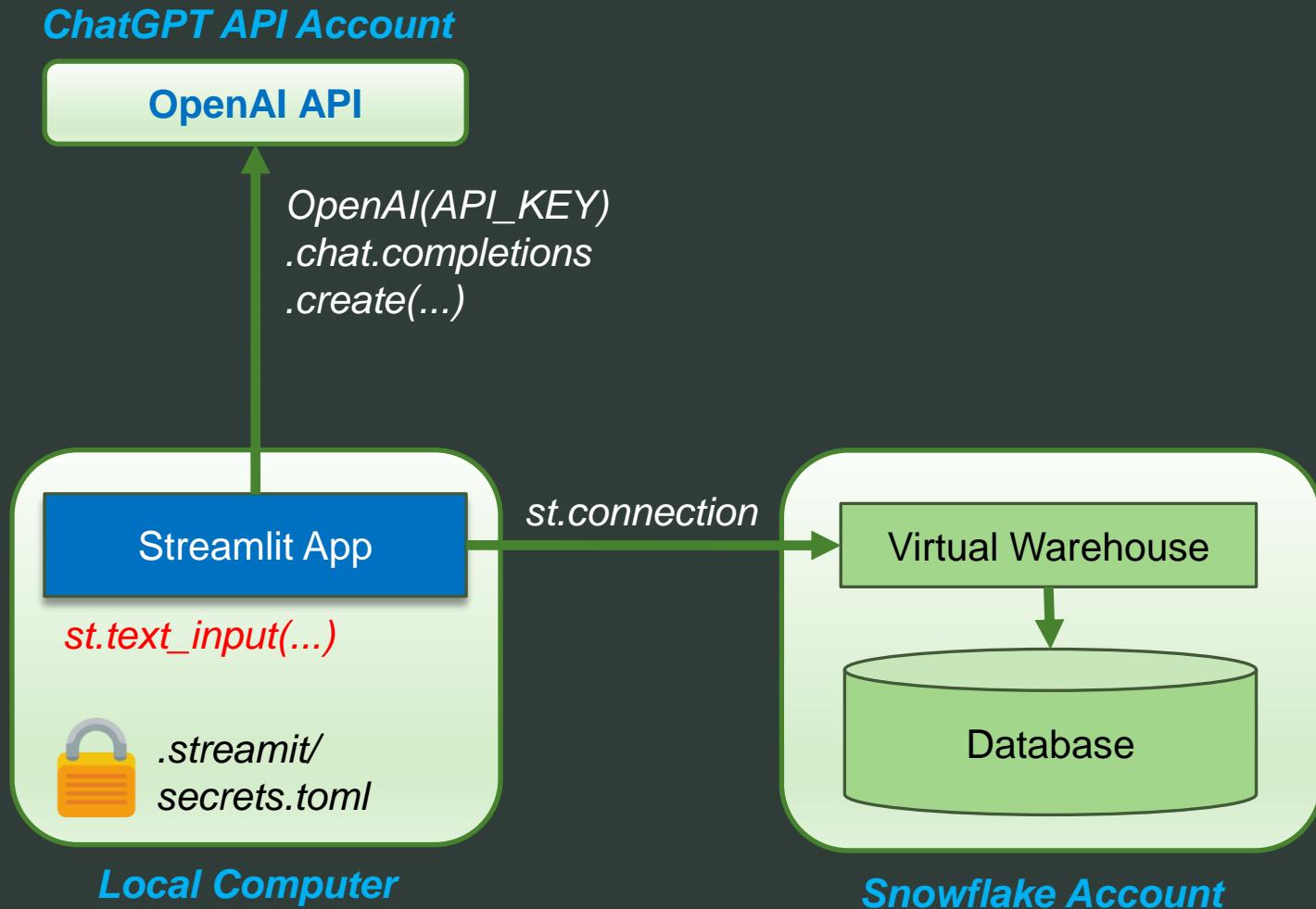
Possible Data Sources in Snowflake

- **Manually Entered Data** - w/ INSERT
- **Uploaded Data Files** - from CSV, Parquet, AVRO...
- **Existing Company Data** - w/ CTAS, INSERT ... AS SELECT ...
- **Public Datasets** - from the Snowflake Marketplace
- **Existing Samples** - shared SNOWFLAKE_SAMPLE_DATA database
- **Enriched Data** - from External Services (ChatGPT as well!)
- **Synthetic Data** - w/ GENERATOR function in Snowflake
- **Realistic Fake Data** - from Faker Python lib
- **Web Scraping Data** - from web pages, Python code

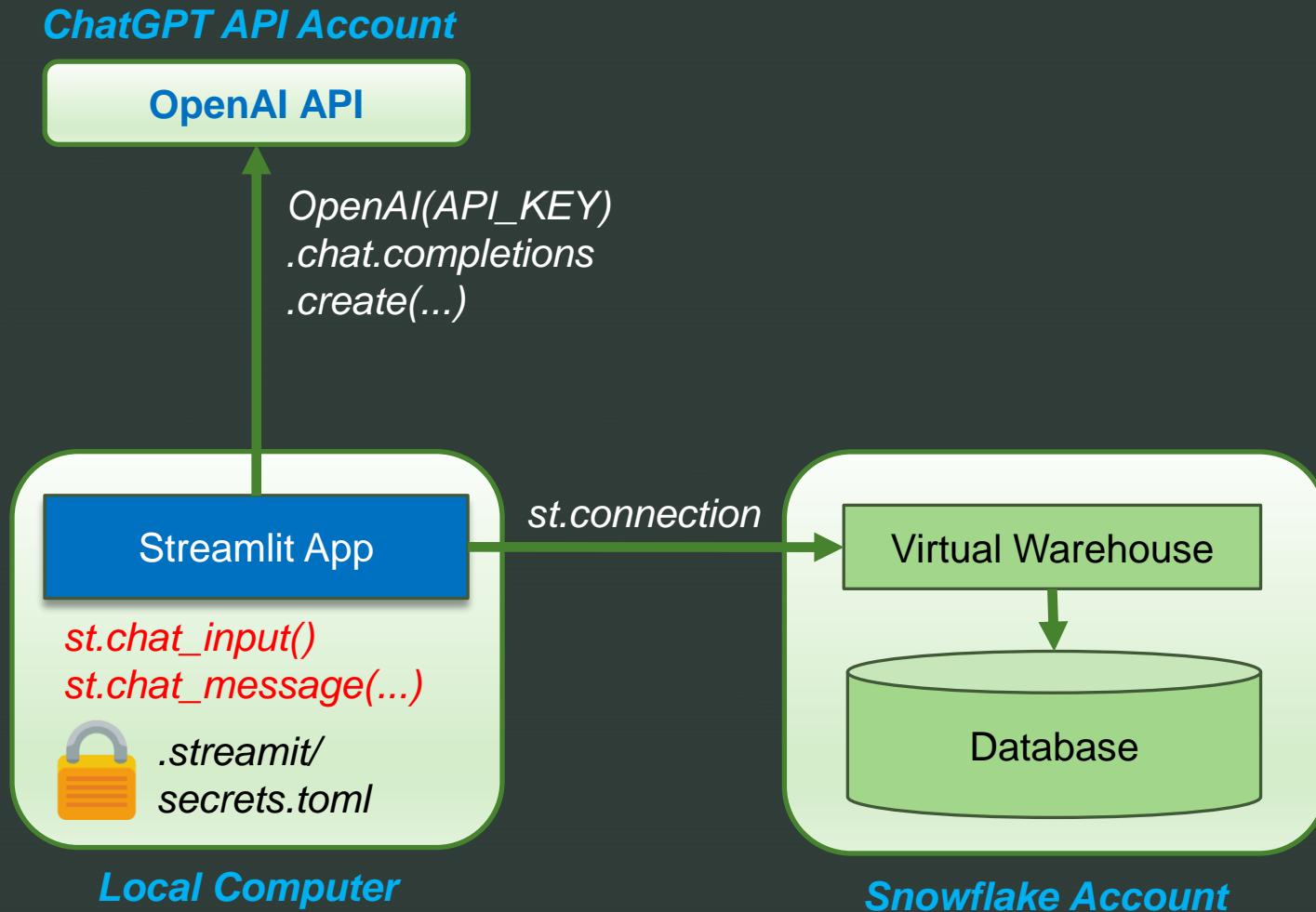
App #3: Snowflake Metadata Inspector in Natural Language

- *Introduction to Snowflake Metadata*
- Create a Simple Q&A Interface with ChatGPT
- Generate and Run Metadata Queries
- Create a Local Streamlit Web App with Tab Control
- Create a Local Streamlit Web App with Chat Controls
- *Review of Snowflake Metadata*

Architecture: Q&A Local Streamlit Web App



Architecture: Chat Local Streamlit Web App



.streamlit/secrets.toml

- exclude from GitHub! ← in .gitignore
 - keep private, only to yourself!
-
- OPENAI_KEY = "<key>" ← ***st.secrets***["OPENAI_KEY"]
 - [connections.**snowflake**] ← ***st.connection***("snowflake")
 - account = "<account>" ← ***st.secrets***[...]["account"]

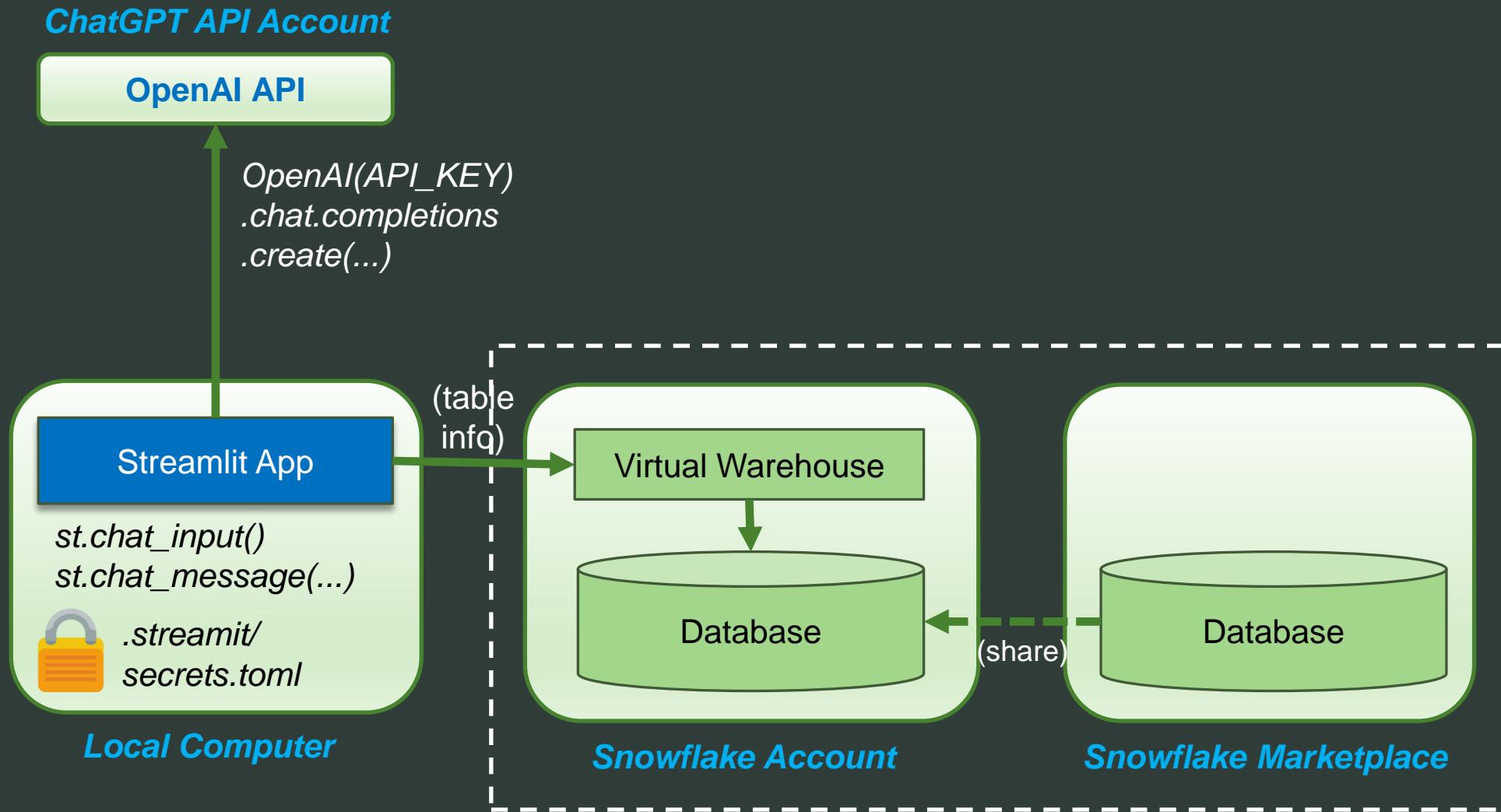
ChatGPT Chat Completions API

- response = OpenAI(key).**chat.completions.create**(...)
- **model** - *gpt-4-1106-preview*, for GPT-4 Turbo
- **messages** - role *system/user/assistant* + content
- **temperature** - 0 for deterministic/repetitive, 1 for random
- **max length** - max tokens to generate (to control cost)

App #4: Interactive Data Analysis with ChatGPT Bot Agent

- *Introduction to Interactive Data Analysis*
- Get Public Datasets from Snowflake Marketplace
- Extract Metadata for Snowflake Objects
- Custom Data Analysis using Natural Language
- Create a Generic ChatGPT Bot
- Create a Data Analysis ChatGPT Bot Inspector
- *Review of Interactive Data Analysis*

Architecture: Marketplace Datasets



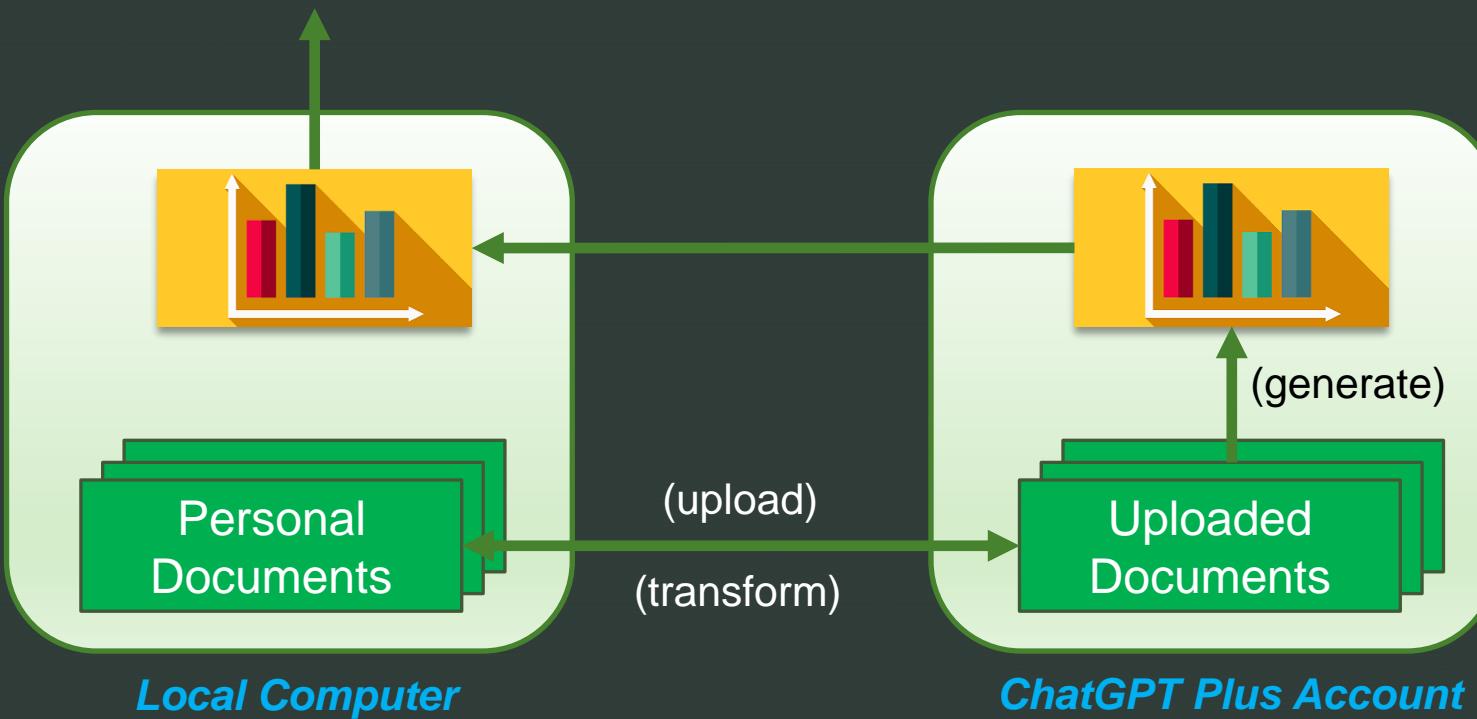
Advanced Prompt Engineering

- extract metadata info about all tables/views + descriptions
- extracted metadata info about all column names + data types
- use <tag> ... </tag> XML-like syntax
- use bullet lists to organize similar data + bold for important
- use ```sql ... ``` markdown for SQL queries
- generate single SQL code snippet
- use "ilike %keyword%" for fuzzy match queries
- limit number of responses (to 10 by default)

App #5: Instant Charts with the Advanced Data Analysis Plugin

- *Introduction to Generated Data Analysis*
- Create Tables with the Regular ChatGPT Subscription
- Exploratory Data Analysis with the ChatGPT Plus Subscription
- Generate Charts and Graphs for Data Analysis
- Generate Cluster Analysis (as a Data Science Experiment)
- *Review of Generated Data Analysis*

Architecture: Advanced Data Analysis



ADA (Advanced Data Analysis) Plugin

- former Code Interpreter
- w/ ChatGPT Plus, ChatGPT Enterprise (confidential)
- max 10 uploaded docs, of max 500MB each
- docs auto-deleted after 3h, max 32k prompts
- file types: PDF, Word, Excel, PowerPoint, CSV, TXT
- for synthesis/transformation/extraction

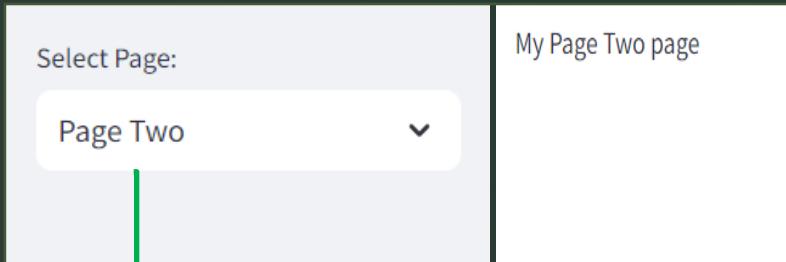
App #6: Generate a Usage Monitoring Dashboard for Snowflake Account

- *Introduction to Usage Monitoring Dashboards*
- Extract Usage Queries and Charts with ChatGPT
- Dashboard as a Streamlit Web App on a Single Page
- Tab-Based Dashboard as a Streamlit Web App
- Dashboard as a Multi-Page Streamlit Web App
- *Review of Usage Monitoring Dashboards*

Snowflake Charts and Dashboards

- cannot automate (no such SQL objects) → manual process
- limited chart support → only 5 chart types
- must refer to third-party online posts
- requires specialized knowledge
- must know Information Schema and Account Usage
- many dashboards for Data Science w/ Streamit apps
- ***ChatGPT can assist in generating queries and charts!***

Streamlit Multi-Page App: w/ SelectBox



```
app.py
```

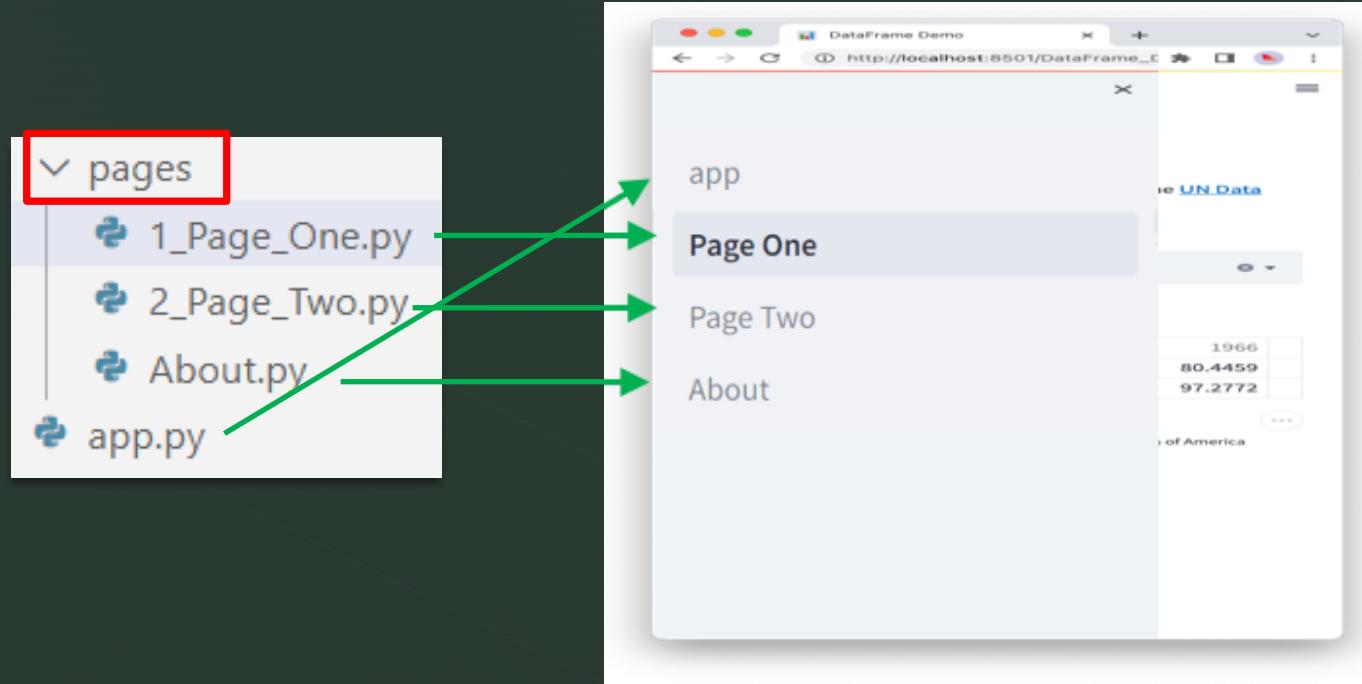
```
import streamlit as st

def main(): st.write("My main page")
def page_one(): st.write("My Page One page")
def page_two(): st.write("My Page Two page")
def about(): st.write("My About page")

funcs = {
    "-": main,
    "Page One": page_one,
    "Page Two": page_two,
    "About": about }
name = st.sidebar.selectbox(
    "Select Page:", funcs.keys())
funcs[name]()
```

The code in the app.py file defines four functions: main, page_one, page_two, and about. It also creates a dictionary called funcs mapping page names to their corresponding functions. The Streamlit application uses a sidebar selectbox to allow users to choose a page. The selected page's function is then called from the funcs dictionary.

Streamlit Multi-Page App: w/ Pages Folder



1_Page_One.py

```
import streamlit as st

st.set_page_config(
    page_title="Plotting Demo",
    page_icon="📈")
```

App #7: Data Enrichment with an External Integration of ChatGPT

- *Introduction to Calling ChatGPT from within Snowflake*
- Call ChatGPT through the OpenAI REST API
- Snowflake Function with External Access Integration
- Create a Simple Streamlit in Snowflake App calling ChatGPT
- Snowflake External Functions for API and Lambda (Obsolete)
- *Review of Calling ChatGPT from within Snowflake*

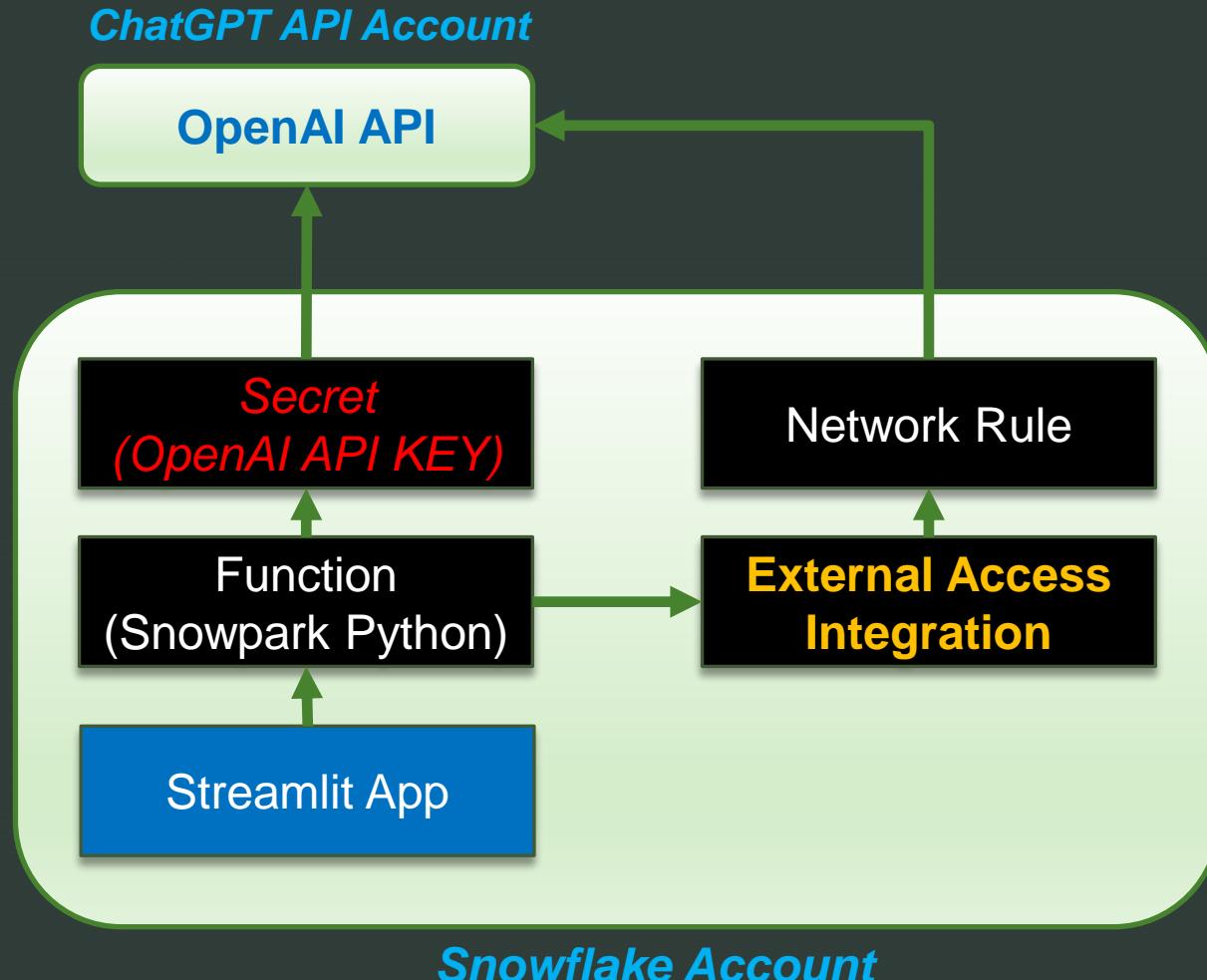
OpenAI REST API

- <https://api.openai.com/v1/chat/completions>
- HTTP POST
- Authorization: Bearer *key* ← HTTP header w/ API key
- { "model": "gpt-4-1106-preview", ← request body (JSON)}
- "messages": [{"role": "user", "content": *prompt*}] }
- request["choices"][0]["message"]["content"] → response

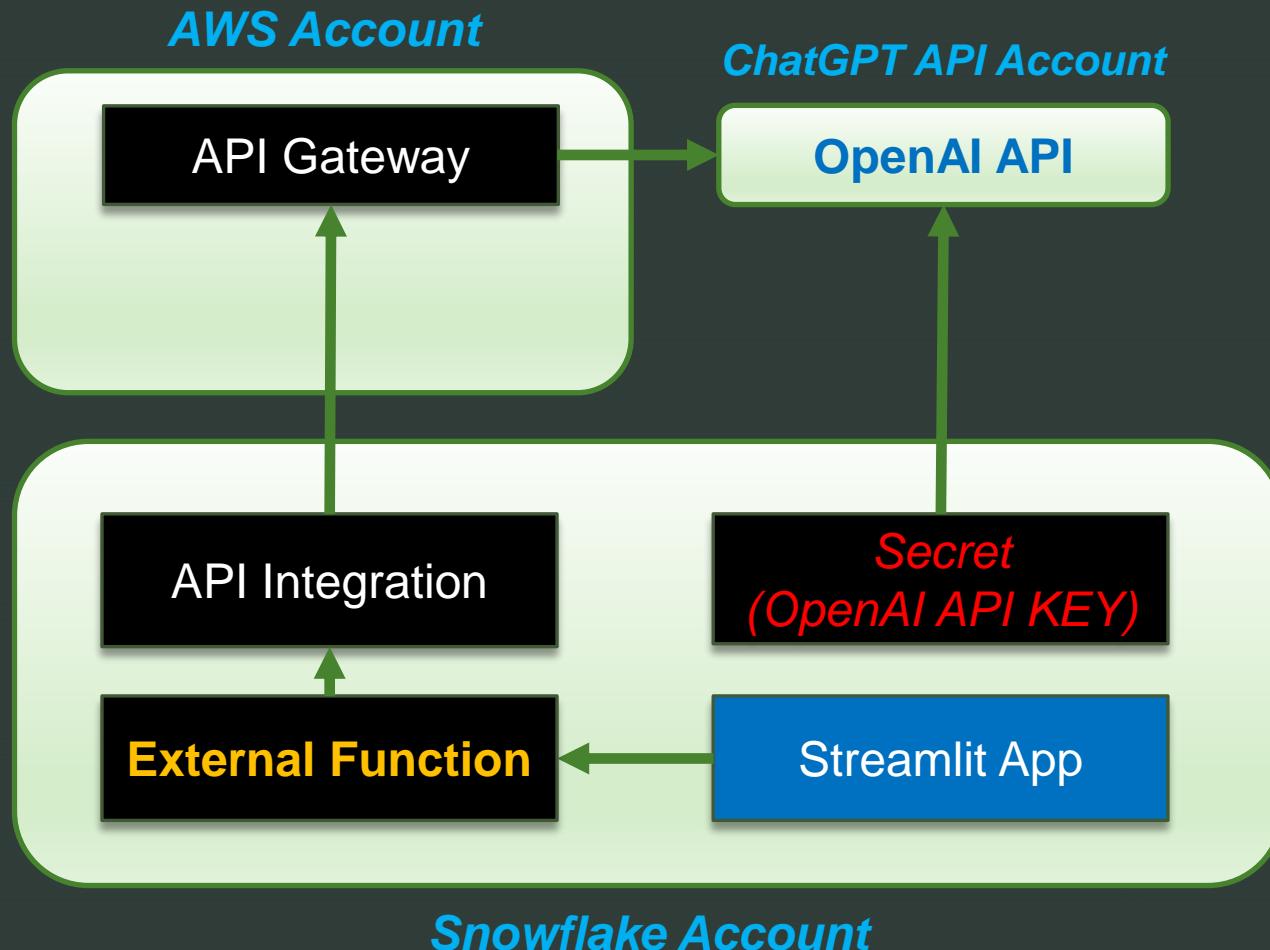
Snowflake to ChatGPT

- (1) Function w/ **External Access Integration**
- (2) External Function w/ API Integration (proxy)
- (3) External Function w/ Lambda (wrapper)
- (4) Snowpark Container Services ← very new!
- `SELECT openai(prompt) → answer`

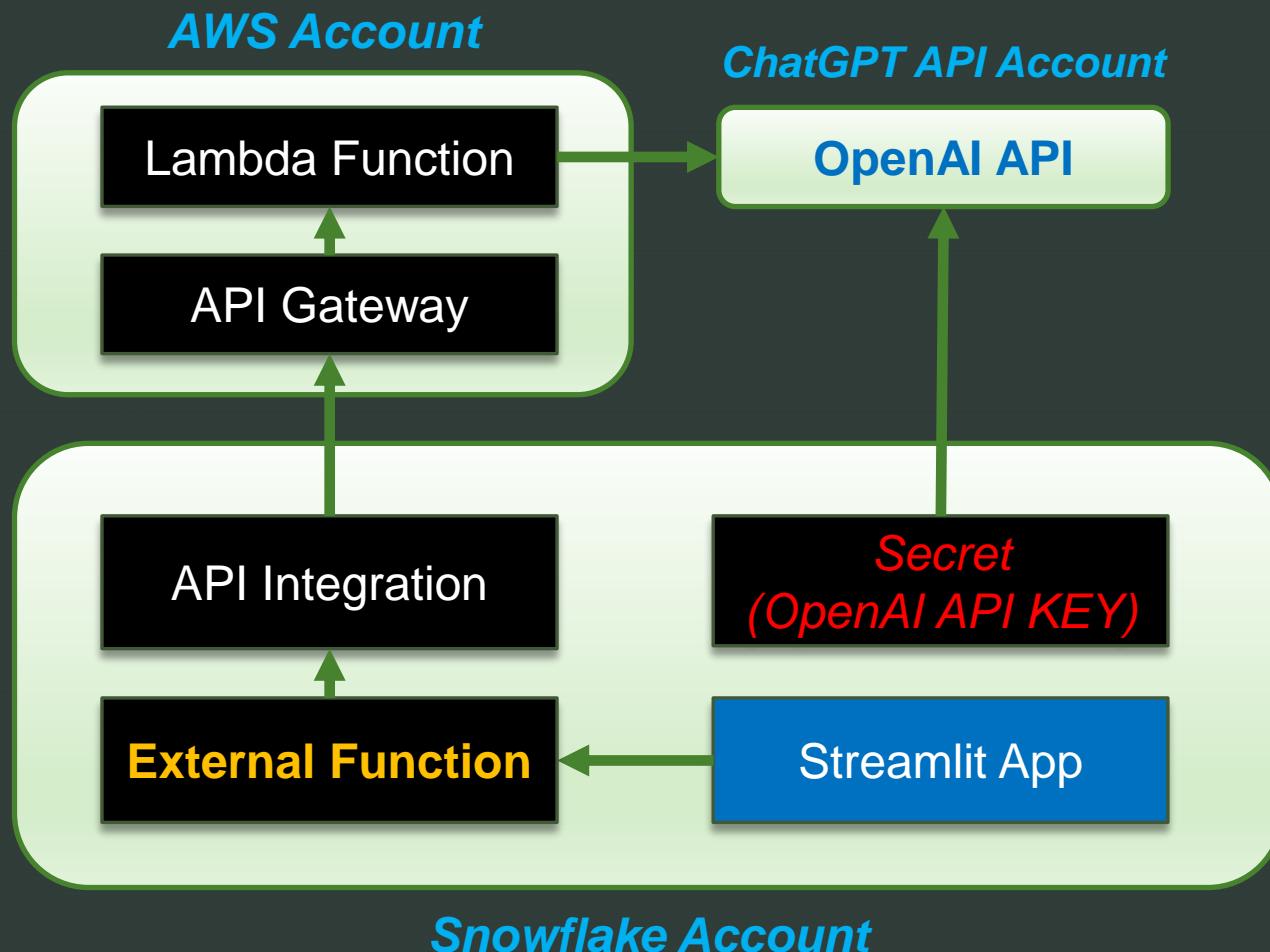
Architecture: External Access Integration



Architecture: Snowflake w/ API Integration



Architecture: Snowflake w/ Lambda



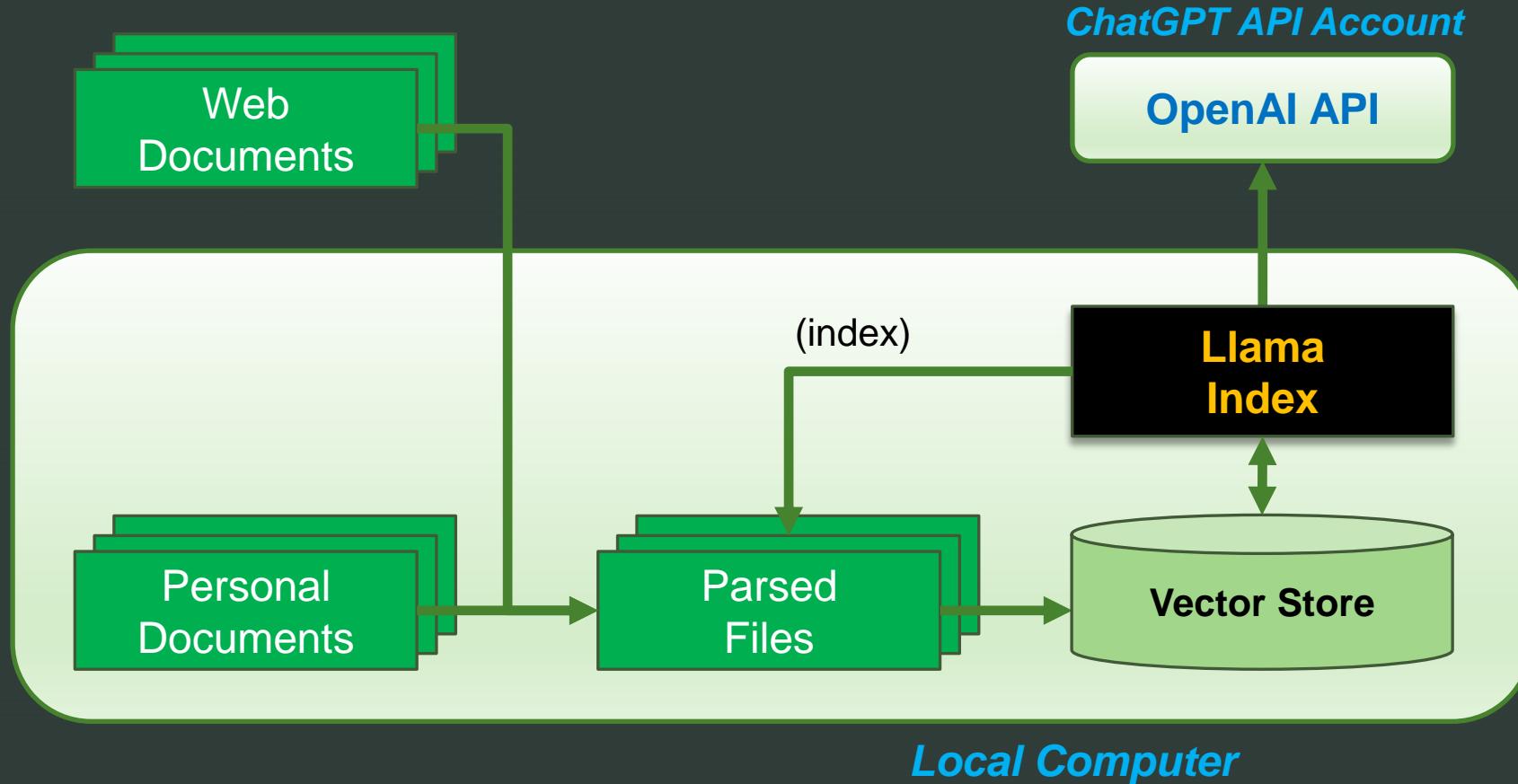
Data Enrichment Possibilities

- Extend existing column data w/ related external info
- Text summarization
- Sentiment analysis → good/bad qualifiers
- Text classification → ex: invoice types
- Text translation → into another language
- Product recommendations

App #8: ChatGPT with LlamaIndex on Personal Documents

- *Introduction to LlamaIndex and RAG*
- Collect Personal and Custom Content
- Create a Knowledge Base with LlamaIndex
- Query ChatGPT with the Indexed Custom Content
- *Review of LlamaIndex and RAG*

Architecture: OpenAI App w/ Llamalndex



LlamalIndex w/ RAG

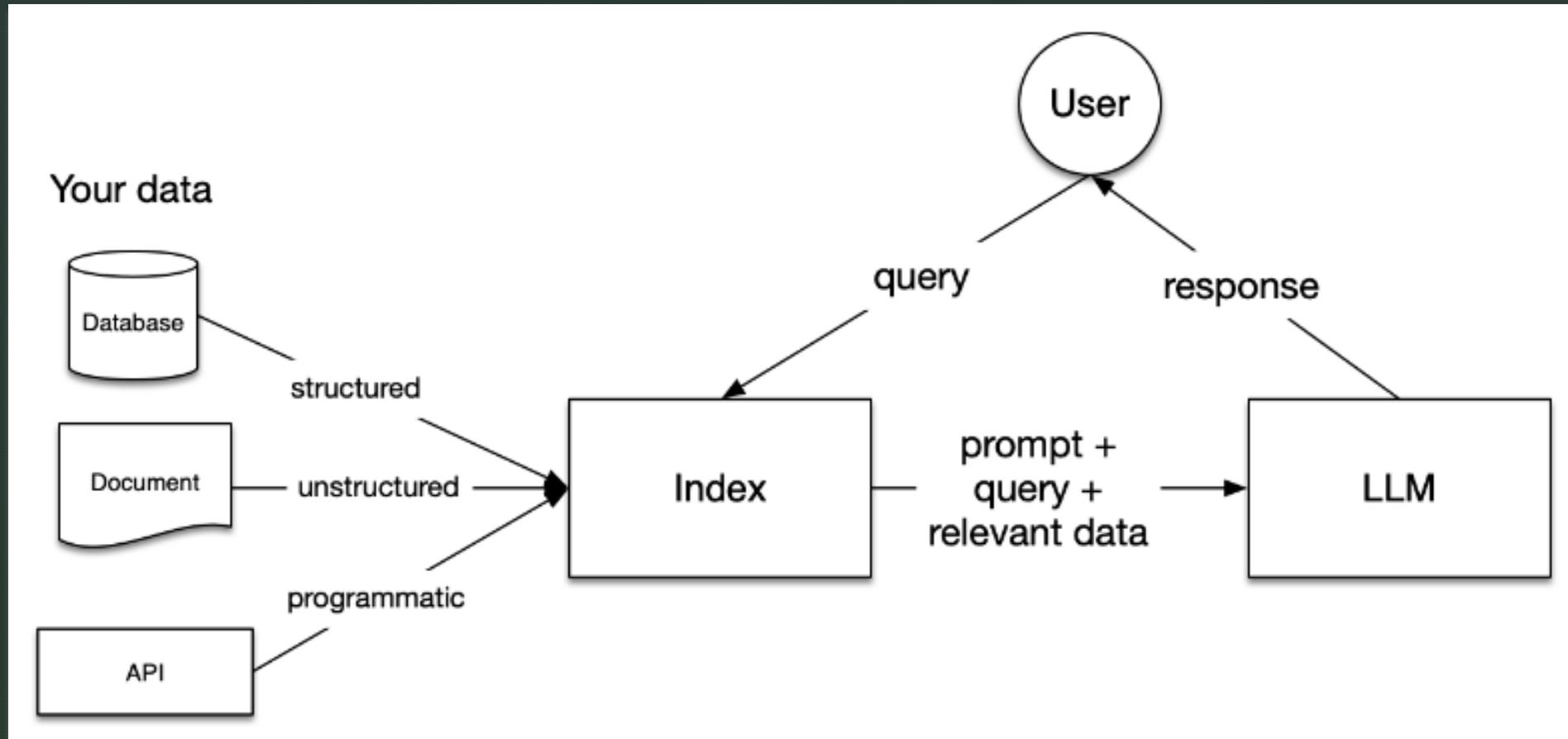
- Retrieves information from your data sources first
- Adds it to your question as context
- Asks the LLM to answer based on the enriched prompt

- There's no training involved → it's cheap
- Data is fetched only when asked → it's always up to date
- Can see the retrieved documents → more trustworthy

RAG (Retrieval Augmented Generation)

- Split content into chunks w/ limited number of words per chunk
- Convert each chunk into ***vector embeddings*** (numbers)
- Store all into a ***vector store*** and build an index for retrieval
- During inference, input prompt is converted to *vector embedding*
- Vector store is searched for higher similarity

RAG Architecture



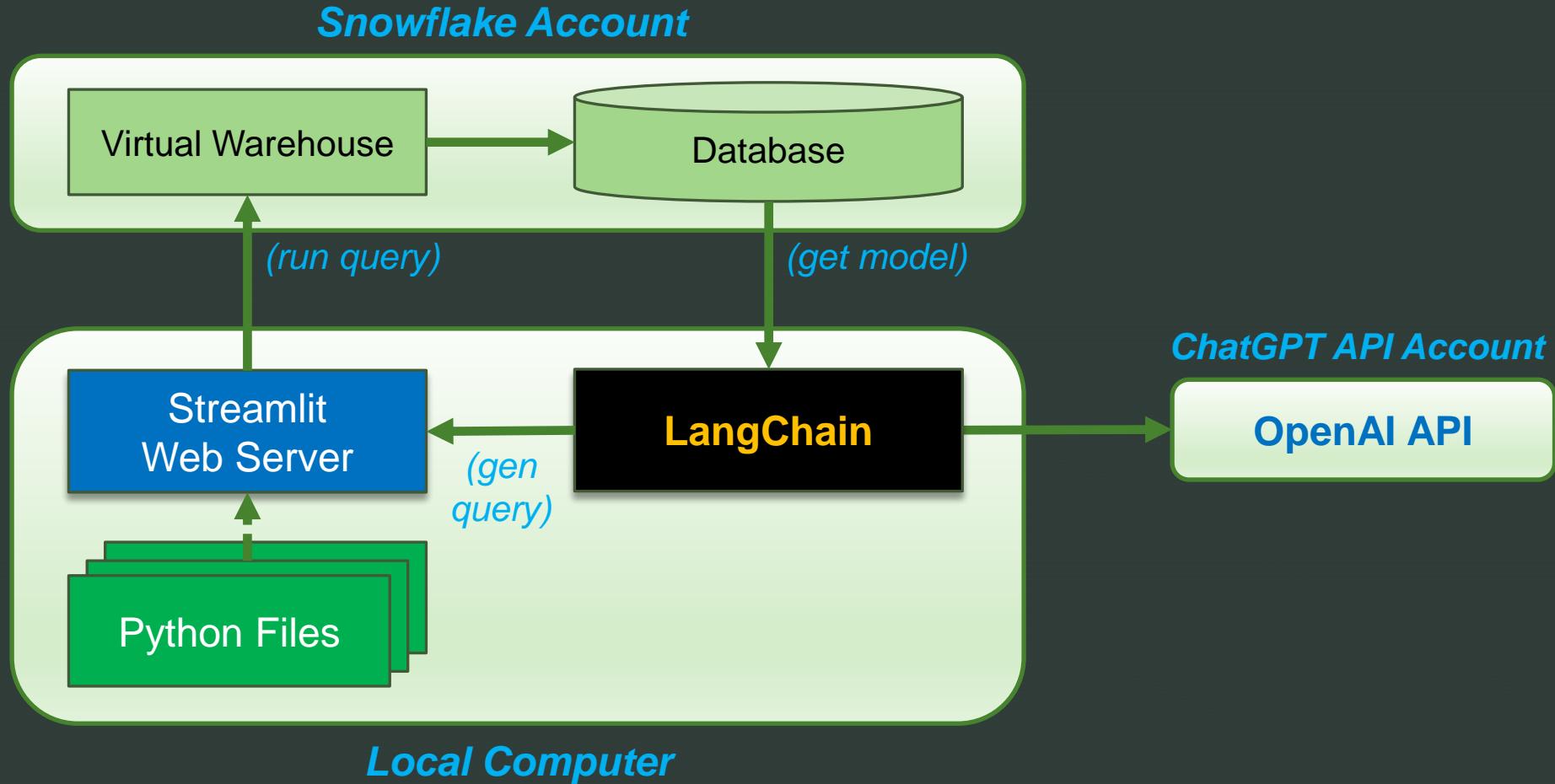
App #9: ChatGPT SQL Generator with LangChain

- *Introduction to LangChain SQL Generation*
- Create & Run a Jupyter Notebook with LangChain
- Implement a SQL Generator as a Local Streamlit Web App
- Deploy the SQL Generator in the Streamlit Community Cloud
- *Review of LangChain SQL Generation*

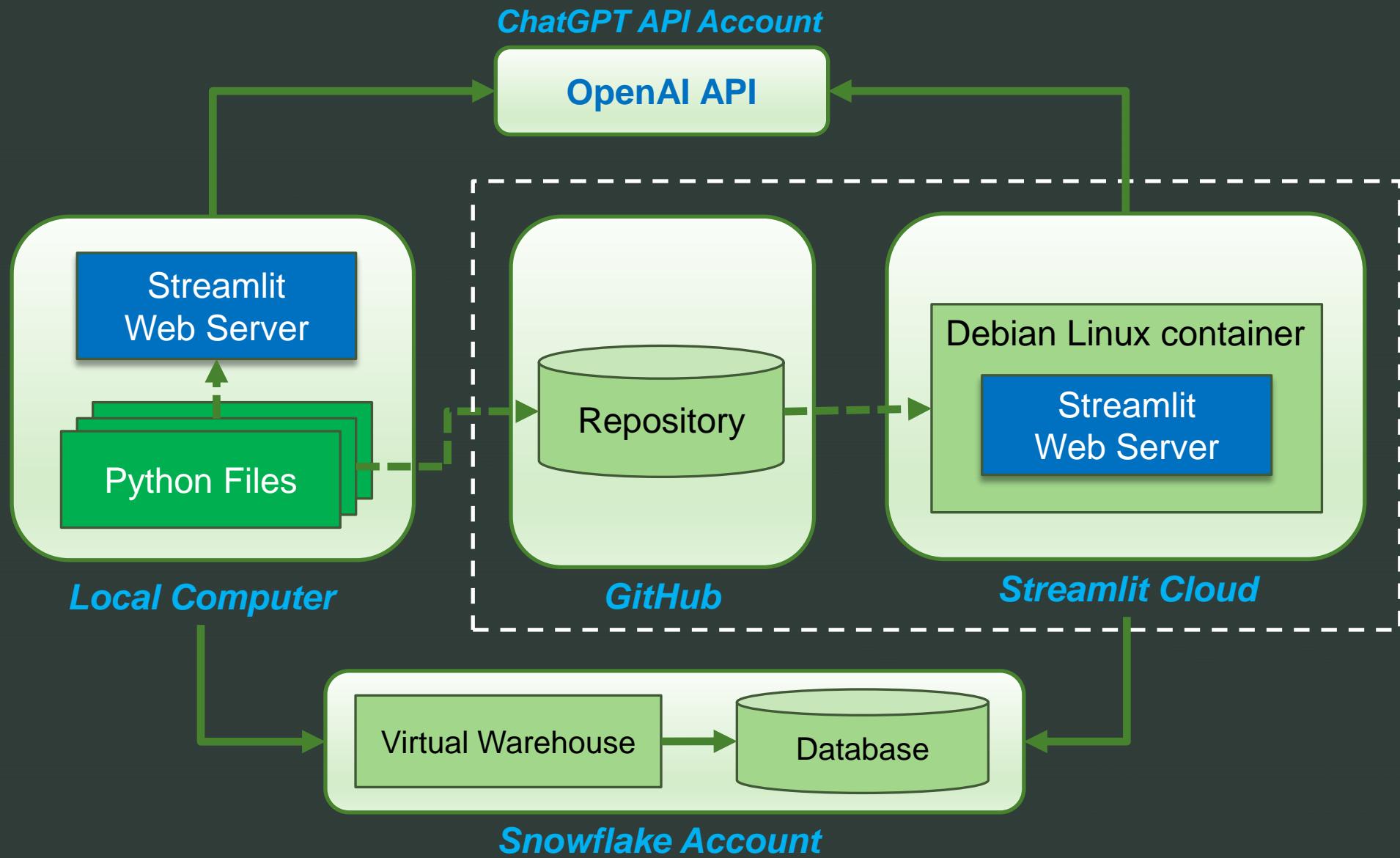
LangChain

- open-source framework to simplify dev of LLM apps
- since Oct 2022, on GitHub
- SQL query generator (from NL questions)
- document analysis and summarization
- chatbots
- code analysis
- RAG (Retrieval-Augmented Generation)

Architecture: Web App w/ LangChain



Architecture: Streamlit Community Cloud





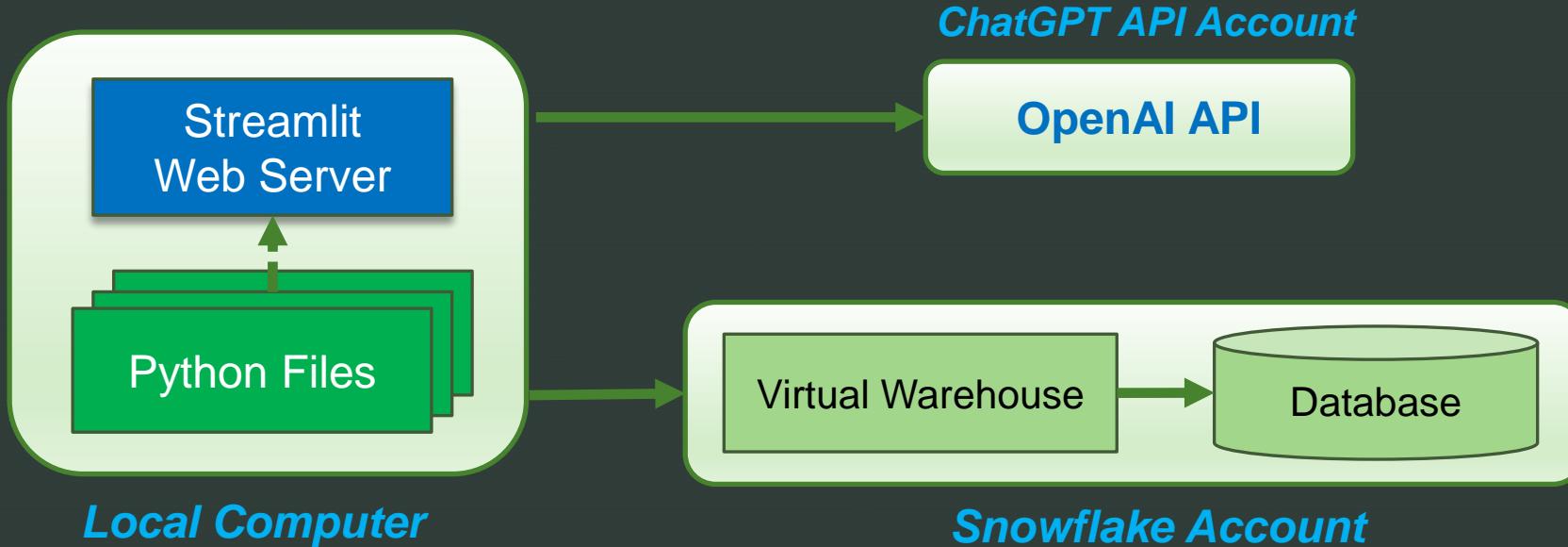
App #10: Snowflake Query Analyzer and Optimizer

- *Introduction to Query Analysis and Performance Optimization*
- Experiment Interactively in VSCode
- Create a Query Analyzer as a Local Streamlit Web App
- Create a Query Analyzer as a Streamlit App calling a UDF
- *Review of Query Analysis and Performance Optimization*

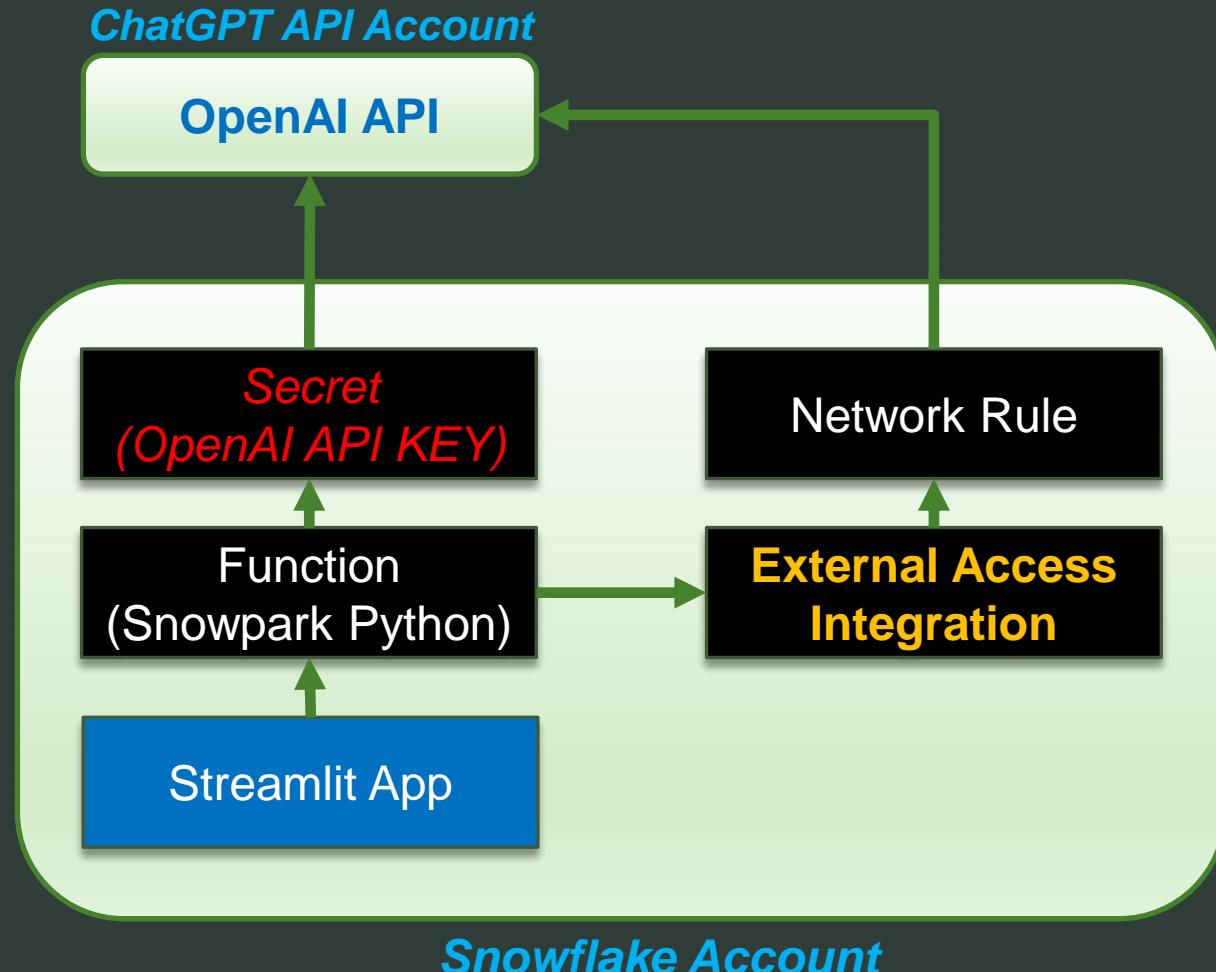
SQL Query-Related Issues

- correct syntax errors → suggest possible fixes
- explain what it does → add as comments
- query execution plan → execute
- performance optimization → clustering keys
- translate in another SQL dialect + similar emulated queries
- encapsulate in a stored procedure

Architecture: Local Streamlit Web App



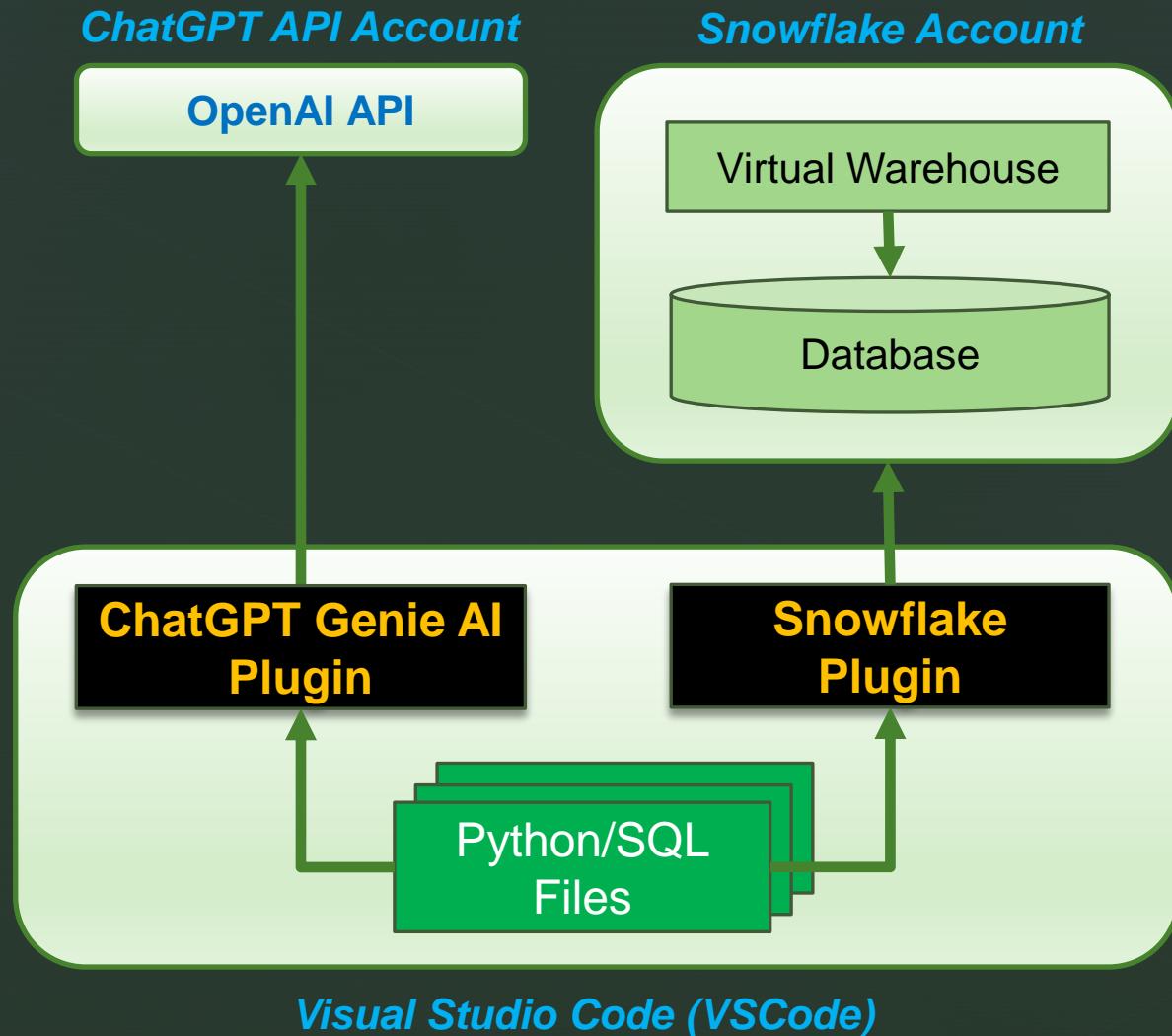
Architecture: Streamlit App w/ EAI



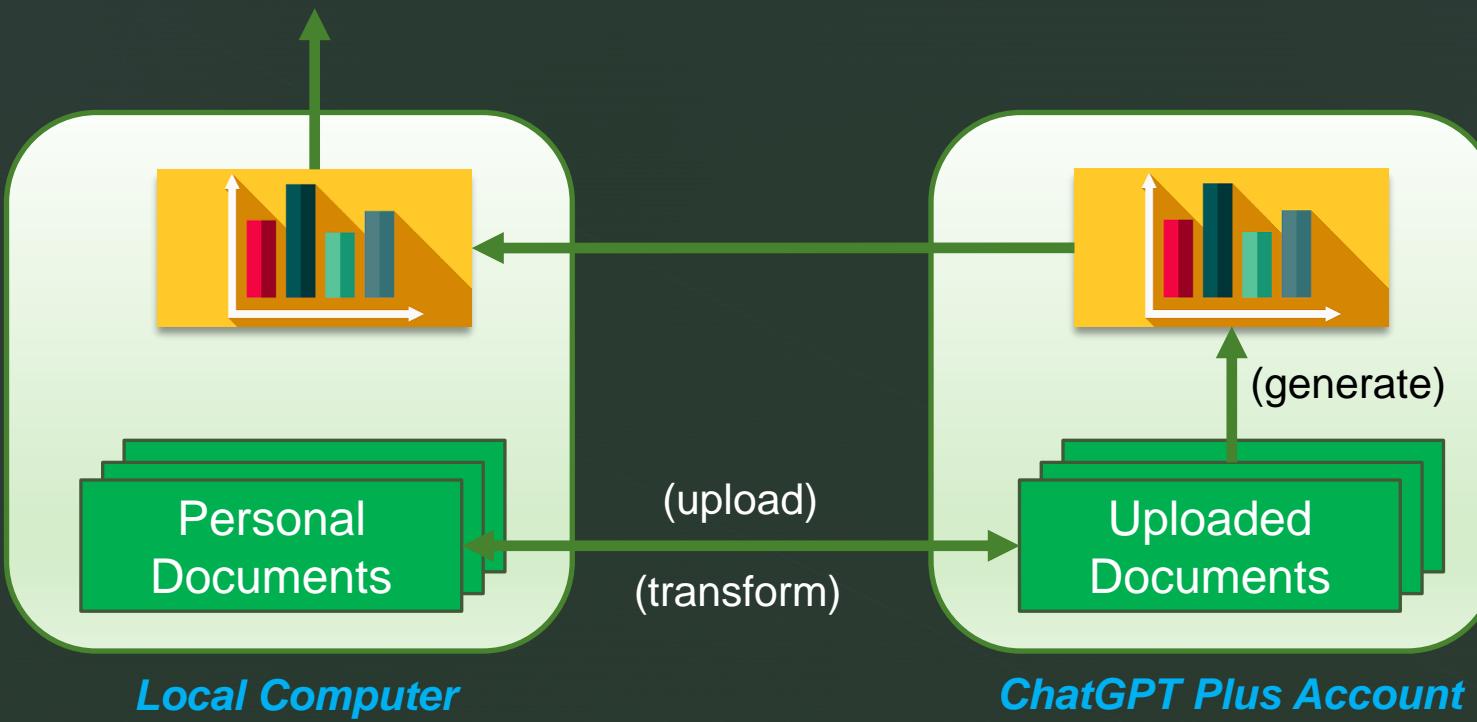
System Architectures

- Connected IDE ← w/ VSCode Plugins
- ChatGPT Plus Playground ← w/ uploaded files
- Local Streamlit Web Apps ← w/ local web server
- Streamlit Apps Deployed in the Community Cloud
- Streamlit in Snowflake Apps ← w/ EAI, EF (obsolete)
- Local/Deployed Web Apps ← w/ LlamaIndex/LangChain

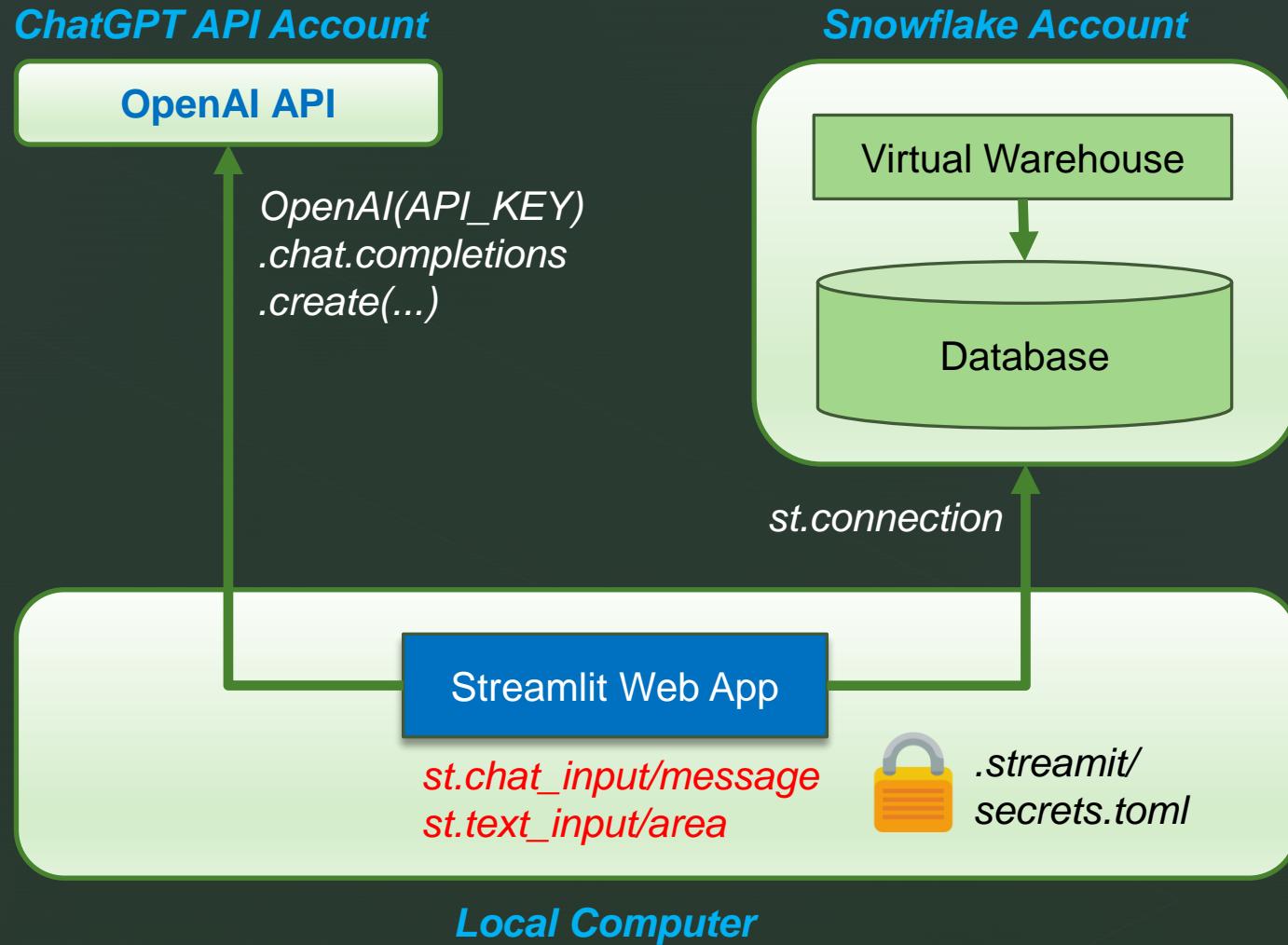
VSCode w/ Plugins



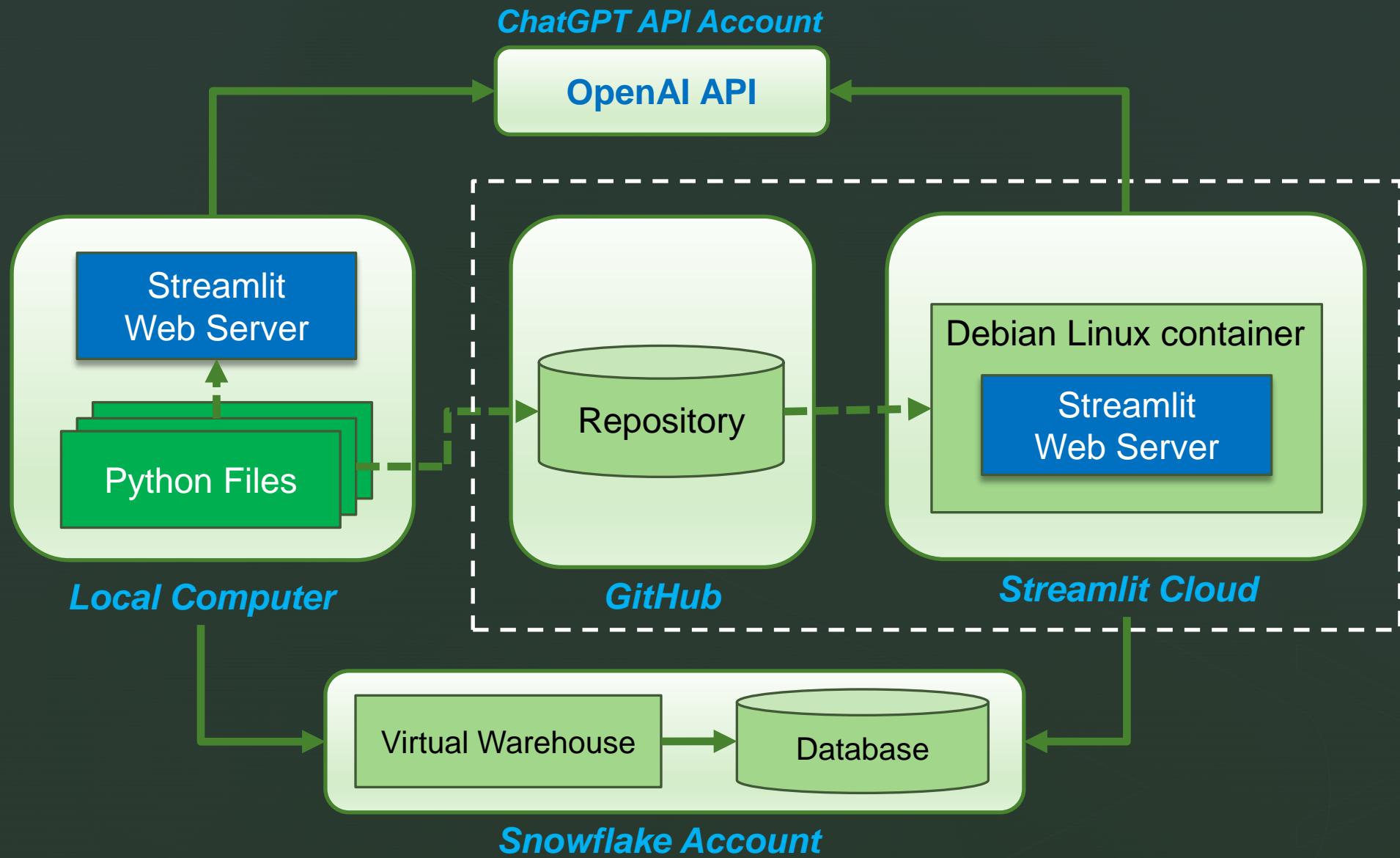
ChatGPT Plus Playground



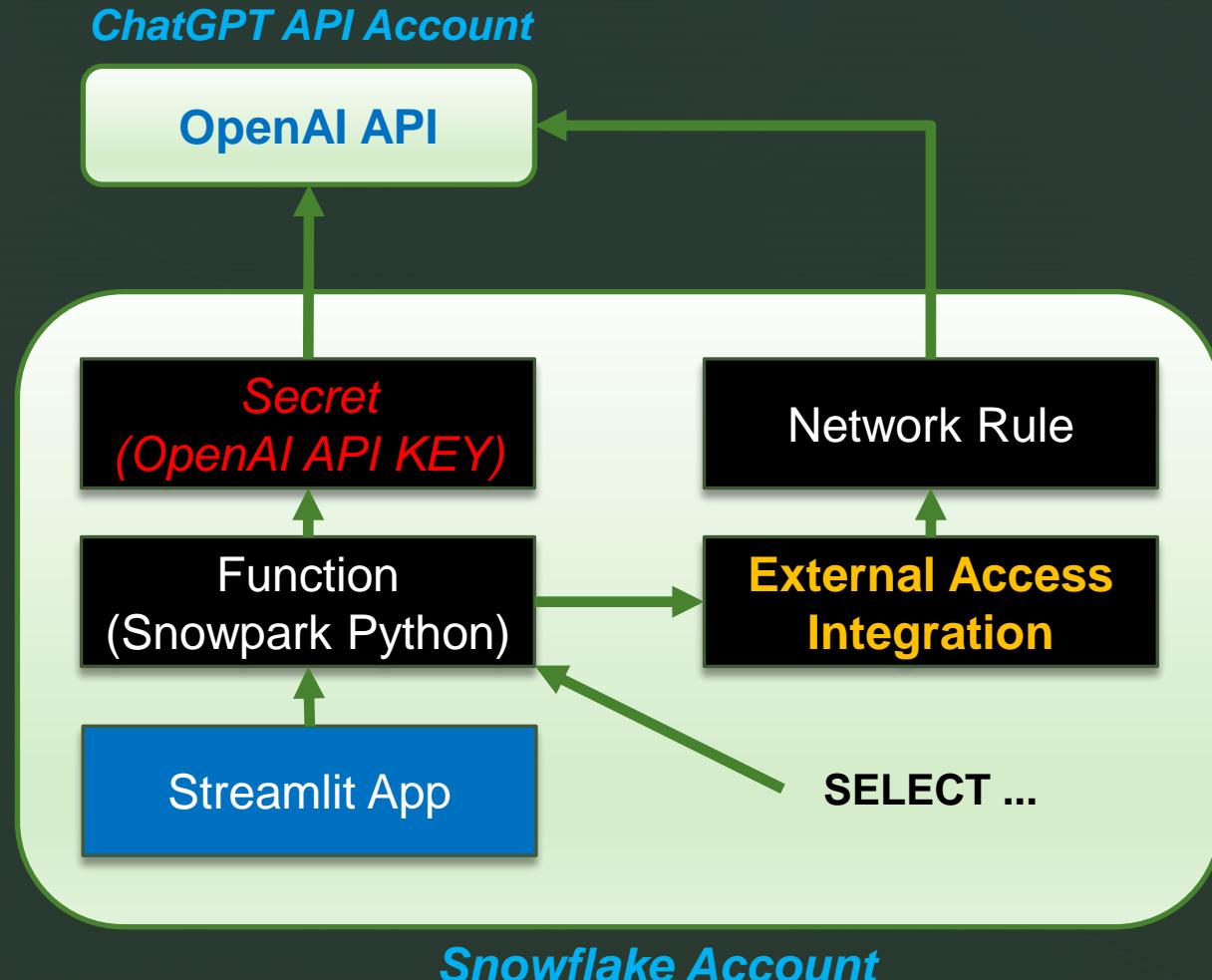
Local Streamlit Web App



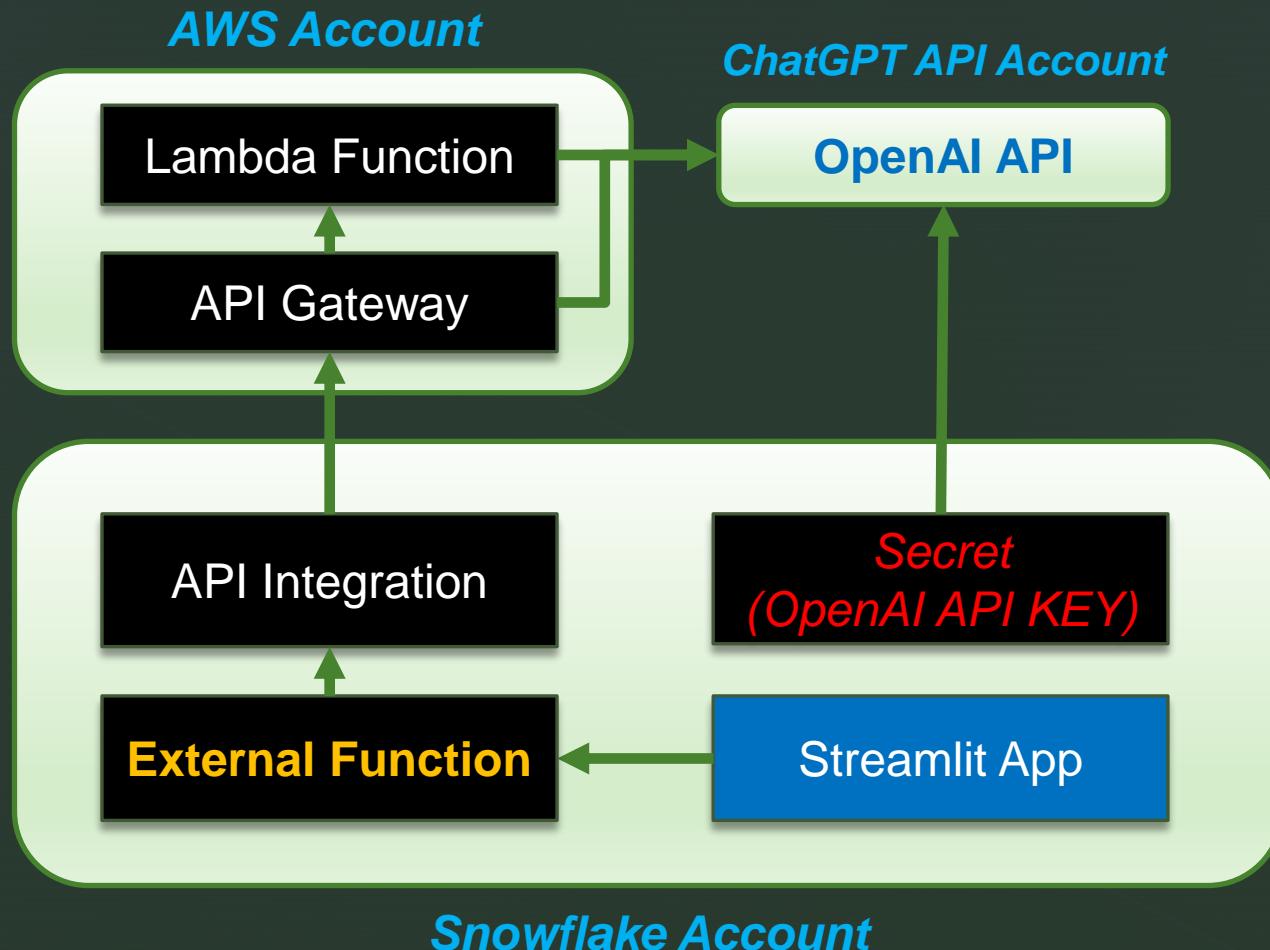
Streamlit Community Cloud



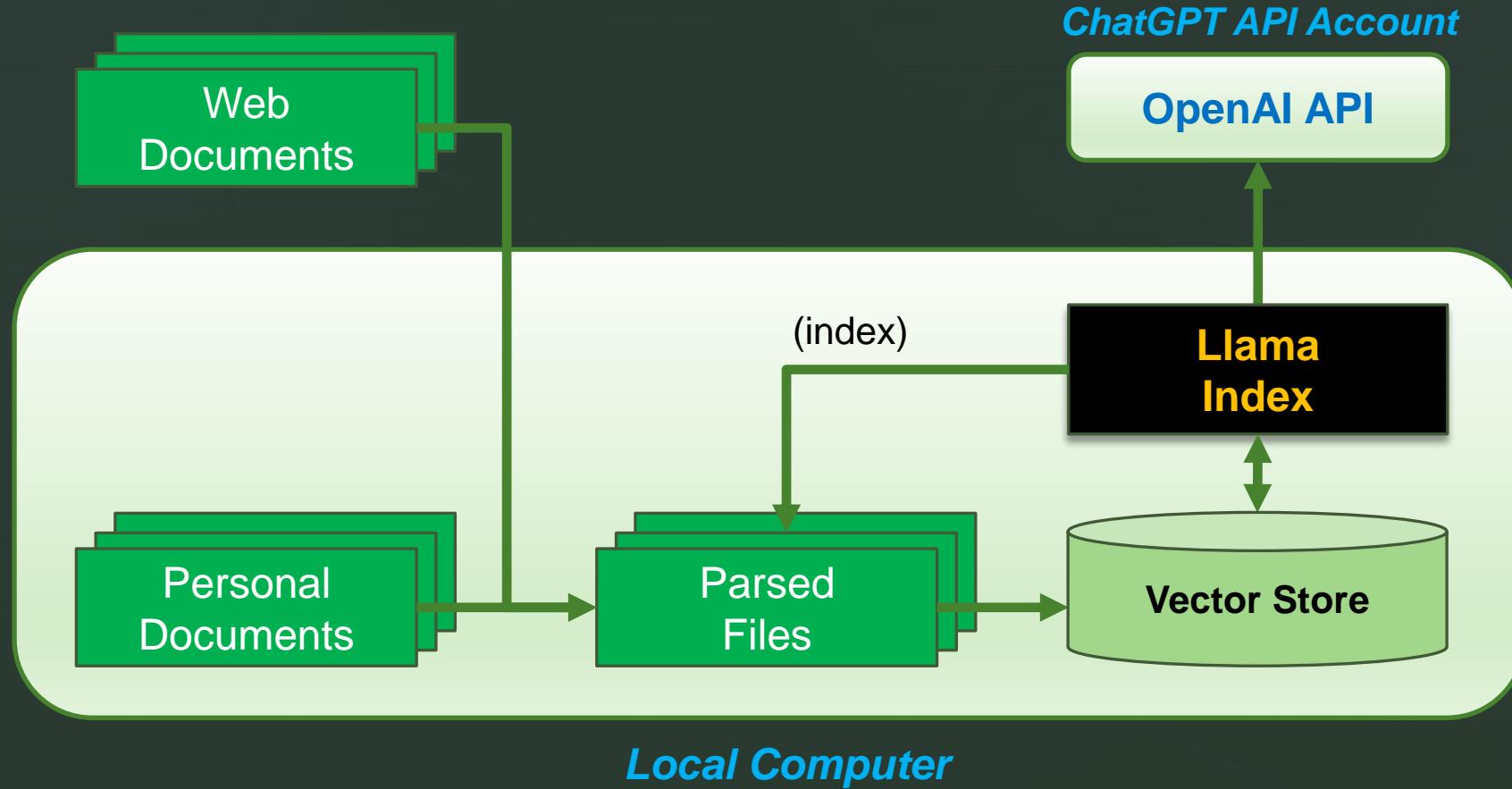
External Access Integration



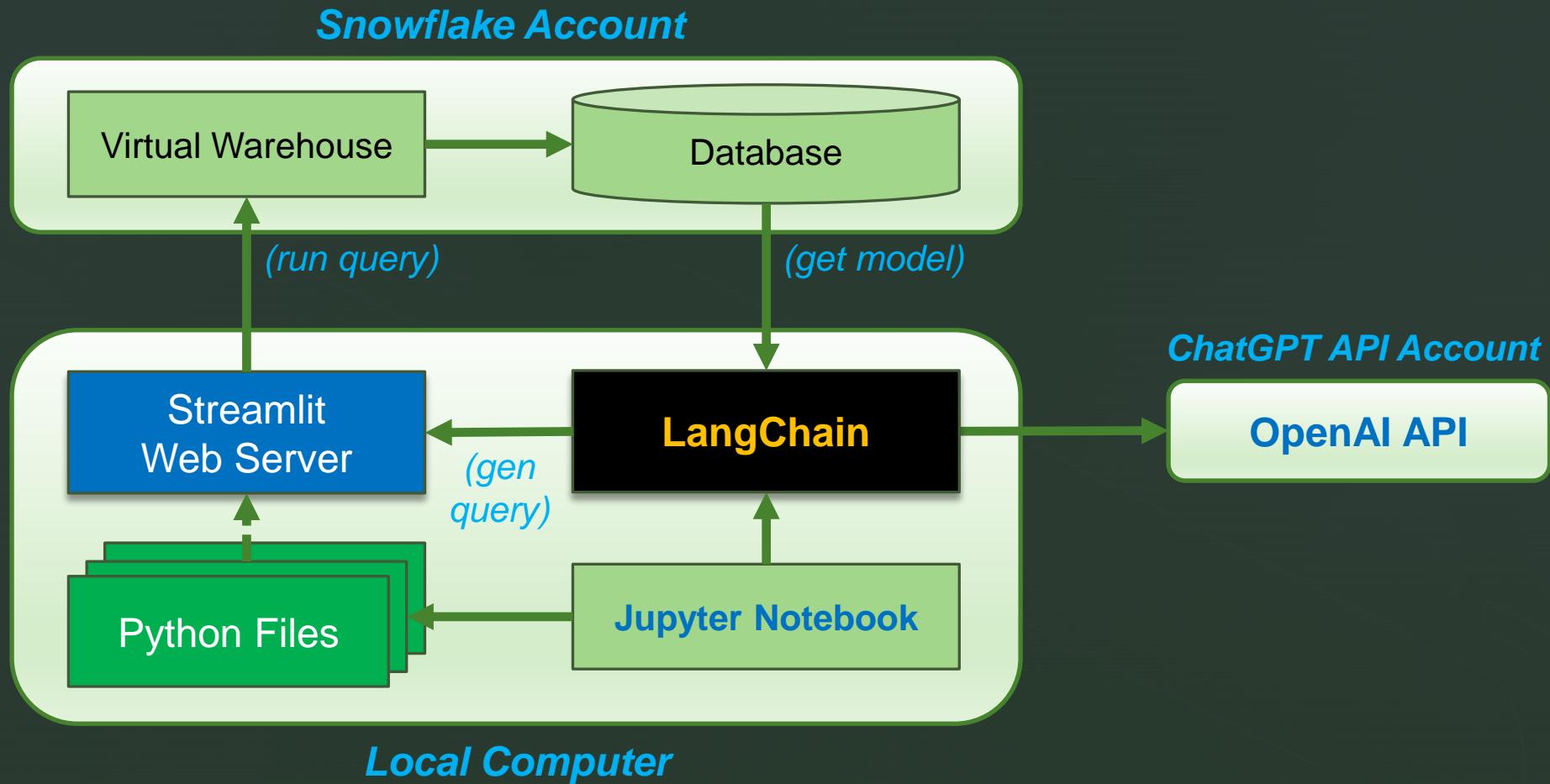
External Function (Obsolete)



Local App w/ LlamalIndex



Local App w/ LangChain



Final Thoughts

- Obvious Increase in Productivity!
- Amazing SQL Query and Python Code Generation Capabilities
- However, Must Check the Validity of what "IT" says!
- Unreliable Super-Expert? ☺
- Careful With Sharing Your Personal/Client Data!
- Jaw-Dropping *Generation* of Advanced Data Analysis and ML
- Data Enrichment With One Single Call for a Whole Dataset?
- Better Integration from within Snowflake? From Streamlit Apps?

Top 10 Snowflake Integrations with ChatGPT

