RETRIEVAL-AUGMENTED GENERATION (RAG) USING SNOWFLAKE CORTEX SEARCH

The architecture you are using combines three powerful technologies—RAG, LLM, and Snowflake Cortex Search—to create an accurate and secure question-answering system.

Breakdown of each component and how they work together to deliver value.

1. Large Language Models (LLMs)

A **Large Language Model (LLM)** is an advanced AI model trained on vast volumes of general text data, allowing it to generate original, human-like text, answer questions, and translate languages.

Function: The Generator

- **Core Capability:** LLMs are the "brain" that generates the final, readable answer. They are powerful at creating creative and engaging text.
- **The Challenge:** Since LLMs rely on their static training data, they often suffer from a **"knowledge cutoff,"** meaning they cannot access current or organization-specific information. This can lead to factual inaccuracies or "hallucinations" (confident, incorrect responses).

2. Retrieval-Augmented Generation (RAG)

RAG is an AI framework that solves the LLM's limitations by connecting it to an external, authoritative knowledge base before generating a response.

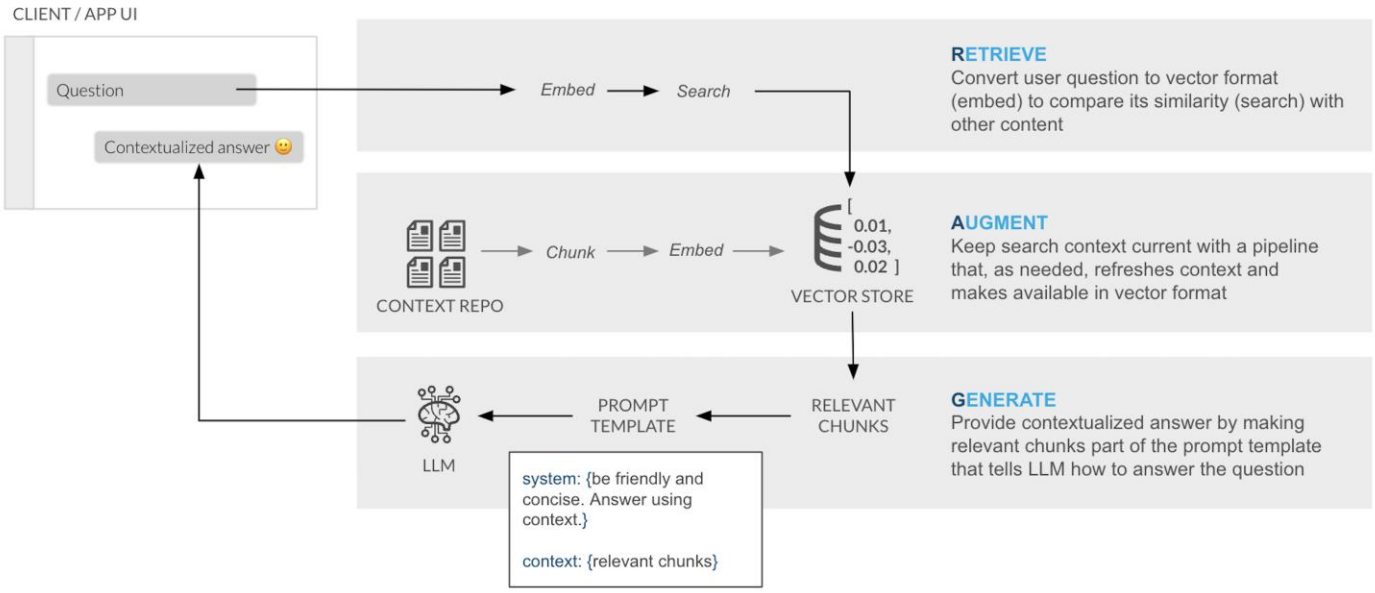
Function: The Optimizer

RAG works by introducing an information retrieval component to the AI workflow:

1. **Retrieve:** RAG first uses the user's query to search the external knowledge base (your policies and documents) for relevant information.
2. **Augment:** The retrieved, authoritative data is then added directly into the prompt given to the LLM as *context*.
3. **Generate:** The LLM uses this new, up-to-date knowledge (along with its general training data) to create a **grounded response**.

Key Benefits of RAG

- **Factual Accuracy:** RAG provides "facts" to the LLM as part of the prompt, mitigating hallucinations and grounding the output in specific data.
- **Access to Current Data:** It enables the model to use the latest, domain-specific information without the high cost and resources required to retrain the entire LLM.
- **Transparency:** RAG can provide the specific source documents (like your policy ID [R1]) that were used to form the answer, allowing the user to verify the response.



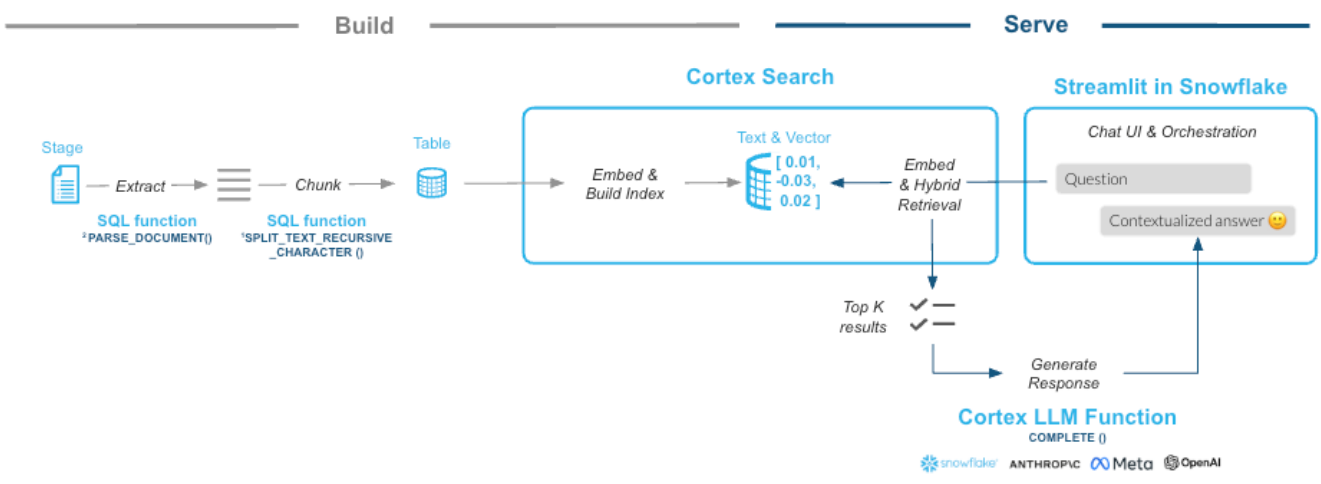
3. Snowflake Cortex Search

Snowflake Cortex Search is a fully managed indexing and retrieval service that simplifies and empower RAG applications. The service automatically creates embeddings and indexes and provides an enterprise search service that can be acessed via APIs

Function: The Retrieval Engine

Cortex Search is specifically designed to work seamlessly within the Snowflake ecosystem. It provides high-quality retrieval with minimal setup.

- **Hybrid Search:** Each query utilizes a combination of search methods to find the best results:
    - **Vector Search** (Semantic): Retrieves documents based on the **meaning** of the query.
    - **Keyword Search** (Lexical): Retrieves documents based on exact word matches.
    - **Semantic Reranking:** Scores and prioritizes the top results to ensure the most relevant documents are passed to the LLM.
- **Fully Managed:** It handles all the complex infrastructure tasks automatically, including creating and managing the text embeddings, maintaining the index, and performing ongoing index refreshes.
- **Secure:** Since the entire process runs inside the Snowflake governance boundary, your proprietary data never needs to move outside your secure environment.



How the Three Components Work Together

Our Streamlit application successfully chains these three parts into a robust system:

1. **Data in Snowflake:** Your policies are stored in a regular Snowflake table (DOCS).
2. **Retrieval:** The **Cortex Search Service** indexes the documents and acts as the retrieval engine. When a user asks a question, Cortex Search quickly returns the most relevant policy documents (R1, R6, etc.) from your internal data.
3. **Augmentation:** Your Streamlit app takes these document snippets and the user's question, packages them into a single, comprehensive prompt, and passes it to the LLM.
4. **Generation:** The **Arctic LLM** (accessible via SNOWFLAKE.CORTEX.COMPLETE) uses the context provided by Cortex Search to generate a highly accurate, cited, and domain-specific answer, which is then delivered back to the user.

STEPS TO CREATE A CORTEX SEARCH IN SNOWFLAKE

1. Environment Setup (Steps 1–4)

1.1 Create Dedicated Role and Warehouse

It's best practice to use a dedicated role (EYLON_RAG_ROLE) for ownership and a small warehouse (RAG_WH) for cost-effective indexing/querying.

-- Replace 'EYLON' with your Snowflake username

```
CREATE ROLE IF NOT EXISTS EYLON_RAG_ROLE;

GRANT ROLE EYLON_RAG_ROLE TO USER EYLON;


-- Create an XSMALL warehouse for compute

CREATE WAREHOUSE IF NOT EXISTS RAG_WH

  WITH WAREHOUSE_SIZE = 'XSMALL' 1.2 Database, Schema, and Context Switch
```

Set up the structure and switch context to ensure subsequent objects are created under the new role and schema

-- Create database and schema

```
CREATE DATABASE IF NOT EXISTS RAG_DB;

CREATE SCHEMA  IF NOT EXISTS RAG_DB.RAG;
```

-- Grant role access

```
GRANT USAGE ON DATABASE RAG_DB TO ROLE EYLON_RAG_ROLE;

GRANT ALL ON SCHEMA RAG_DB.RAG TO ROLE EYLON_RAG_ROLE;

GRANT SELECT ON ALL TABLES IN SCHEMA RAG_DB.RAG TO ROLE EYLON_RAG_ROLE;
```

-- Set context for subsequent steps

```
USE ROLE EYLON_RAG_ROLE;
```

USE SCHEMA RAG_DB.RAG;

2. Knowledge Base and Indexing (Steps 5–8)

2.1 Create Documents Table

The DOCS table holds the knowledge base content. The body column is the primary text field for searching.

-- Create the documents table

```
-- Create a table to store documents for RAG
CREATE OR REPLACE TABLE DOCS (
  id          STRING      NOT NULL,       -- unique ID for the document (controlled by us)
  title       STRING,                     -- short title or description of the document
  url         STRING,                     -- optional URL for reference (e.g., link to source)
  source_type STRING,                     -- category or type of source (e.g., 'Policy', 'FAQ', etc.)
  updated_at  TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP,  -- last updated timestamp (auto-set on insert)
  body        STRING                      -- the full text content of the document (could be plain text, HTML, etc.)
);
```

-- Enable change tracking for incremental index updates

```
ALTER TABLE DOCS SET CHANGE_TRACKING = TRUE;
```

2.2 Load Sample Data

Insert example policies and guides to populate the knowledge base.

```
-- Insert 20 sample documents (truncated for brevity, ensure all 20 are included when running)

INSERT INTO DOCS (id, title, url, source_type, body) VALUES

('R1','Return Policy – Electronics','https://kb.retailer.com/returns/electronics','Policy', 'Customers have 30 days to return electronics with receipt. Opened laptops incur a 10% restocking fee.'),

('R2','Return Policy – Apparel','https://kb.retailer.com/returns/apparel','Policy', 'Apparel items may be returned within 60 days if unworn and with tags attached.'),

('R6','Store Credit Policy','https://kb.retailer.com/payments/store-credit','Policy', 'Returned items without receipt may be refunded as store credit at the lowest price in the last 60 days.'),

('R8','Holiday Season Returns','https://kb.retailer.com/returns/holiday','Policy', 'Items purchased between Nov 1 and Dec 31 can be returned until Jan 31 with proof of purchase.'),

('R20','Omnichannel Returns','https://kb.retailer.com/returns/omnichannel','Policy', 'Items bought online can be returned in-store with packing slip, or shipped back using pre-paid label.');
```

2.3 Create Cortex Search Service

This command creates the vector index (RETAIL_KB_SEARCH) on the body column using the Arctic embedding model.

```
-- Create the semantic search index

CREATE OR REPLACE CORTEX SEARCH SERVICE RETAIL_KB_SEARCH
```

```
  ON body -- The column to vectorize and search against
  ATTRIBUTES id, title, url, source_type, updated_at -- Metadata to store/return
  WAREHOUSE = RAG_WH
  TARGET_LAG = '5 minutes' -- Update frequency
  EMBEDDING_MODEL = 'snowflake-arctic-embed-m-v1.5' -- LLM embedding model
AS
  SELECT id, title, url, source_type, body, updated_at FROM DOCS;
```

3.1 Test Search Retrieval

Verify the index works and use the filter option to demonstrate metadata filtering.

```
-- Query the search service for "return policy" and filter for 'Policy' documents
SELECT
  PARSE_JSON(
    SNOWFLAKE.CORTEX.SEARCH_PREVIEW(
      'RAG_DB.RAG.RETAIL_KB_SEARCH',
      '{
        "query": "return policy",
        "columns": [ "id", "title", "url", "source_type", "body" ],
        "filter": { "@eq": { "source_type": "Policy" } },
        "limit": 3
      }'
    )
  )['results'] AS results;
```

3.2 Full RAG Workflow (Retrieval + Generation)

This final, comprehensive query orchestrates the entire RAG flow: searching, preparing the context, and calling the Arctic LLM for the final answer.

```
-- 1. Search (preview)
WITH preview AS (
  SELECT PARSE_JSON(
         SNOWFLAKE.CORTEX.SEARCH_PREVIEW(
           'RAG_DB.RAG.RETAIL_KB_SEARCH',
           '{
```

```
      "query": "What is the retailer return policy for electronics?",
      "columns": ["id","title","url","source_type","body"],
      "limit": 5
    }'
  )
 )['results'] AS results
),

-- 2. Flatten results into rows
flat AS (
 SELECT
  value:"id"::STRING    AS id,
  value:"title"::STRING AS title,
  value:"body"::STRING  AS body
 FROM preview, LATERAL FLATTEN(input => results)
),
-- 3. Construct the LLM messages array (the RAG prompt)
history AS (
 SELECT ARRAY_CAT(
   -- Add System Message and User Question
   ARRAY_CONSTRUCT(
    OBJECT_CONSTRUCT('role','system','content',
     'Answer using ONLY the CONTEXT provided. Cite sources by their [id]. If the answer is not in the context, say you do not know.'),
    OBJECT_CONSTRUCT('role','user','content',
     'What is the retailer return policy for electronics?')
   ),
   -- Add retrieved Context Snippets (Aggregated)
   ARRAY_AGG(
    OBJECT_CONSTRUCT(
     'role','user',
     'content', '[' || id || '] ' || COALESCE(title,'') || '\n' || LEFT(body, 800)
    )
   )
```

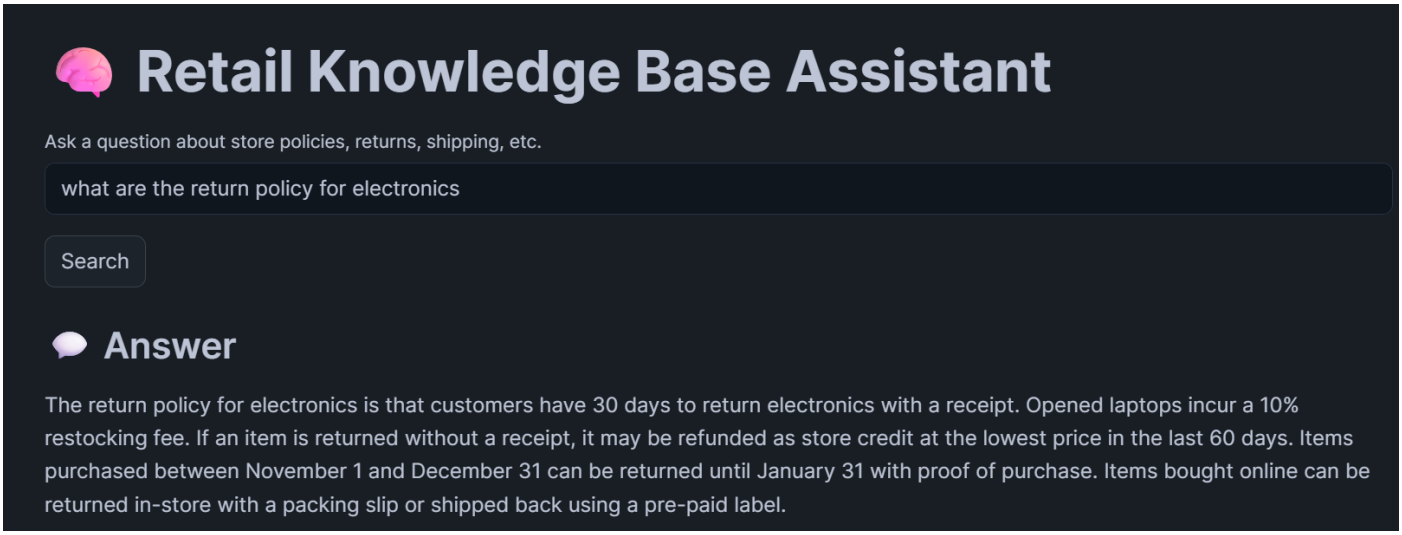
```
        ) AS messages
  FROM flat
)
-- 4. Call the Cortex COMPLETE function (Arctic LLM)
SELECT SNOWFLAKE.CORTEX.COMPLETE(
        'snowflake-arctic',
        messages,
        OBJECT_CONSTRUCT('max_tokens', 300, 'guardrails', TRUE)
      ) AS answer
FROM history;
-- Grant necessary permissions to the RAG role
GRANT USAGE, OPERATE, MONITOR ON WAREHOUSE RAG_WH TO ROLE EYLON_RAG_ROLE;

-- Enable Cortex permissions needed to create the search service
GRANT DATABASE ROLE SNOWFLAKE.CORTEX_USER TO ROLE EYLON_RAG_ROLE;
```

Retail Knowledge Base Assistant (Streamlit)

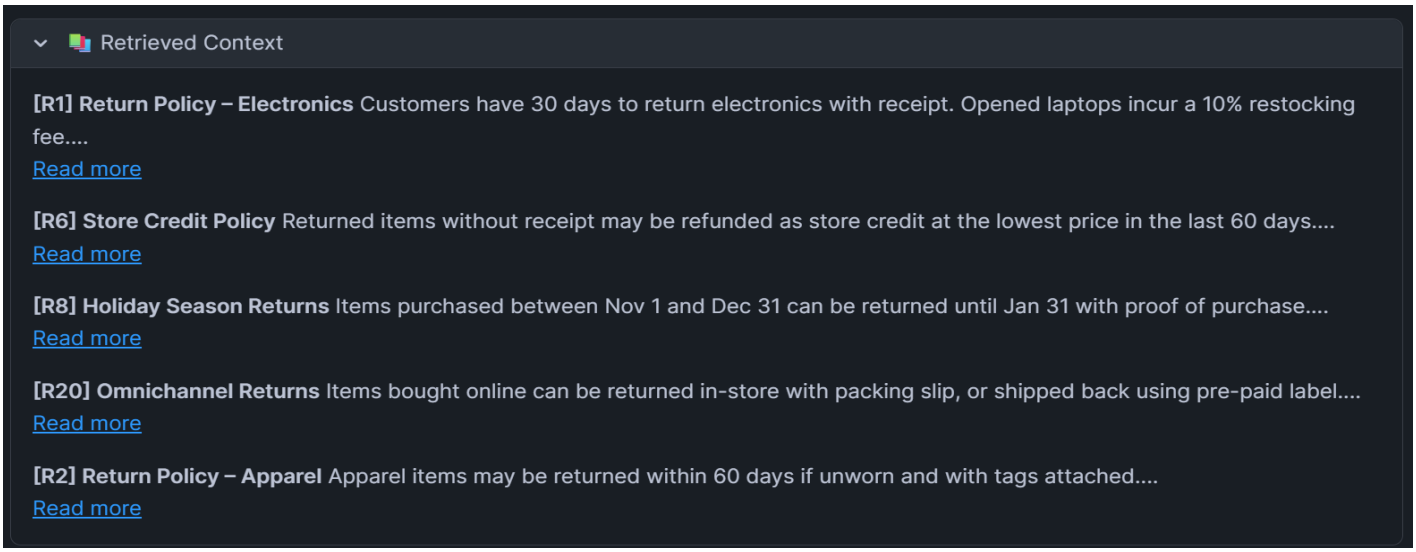
1. What the App Does (From the User's View)

The Retail Knowledge Base Assistant is a simple chat interface built to give you accurate, up-to-date information quickly.

1. Input: You type your question (e.g., "What's the return policy for electronics?") into the text box and click "Search".



2. Output: In just a moment, the app provides a clear Answer. It also shows a section called "Retrieved Context", which displays the exact internal documents it used to form the answer, proving the answer is reliable.



2. The Smart Process Behind the Scenes (RAG)

The app works using a three-step method called RAG (Retrieval-Augmented Generation), which happens entirely within Snowflake. This process ensures the answer is both coherent and factually correct.

Step 1: Retrieval (Finding the Facts) 🔎

When you ask a question, the app's first job is to find the relevant policy documents from your internal knowledge base.

- The app sends your question straight to Snowflake Cortex Search, which acts like a specialized, super-smart search engine.
- It doesn't just look for keywords; it understands the *meaning* (semantic search) of your question and pulls out the top 5 relevant policy documents.
- It also grabs the documents' details, like their unique IDs and titles.

Step 2: Prompt Construction (Building the Instructions) 📜

Once the facts are gathered, the app packages everything neatly for the AI model:

- It creates a set of instructions telling the AI: "You must answer using ONLY the facts I am giving you. You must also cite the source document ID (e.g., [R1]) in your answer.".
- It then combines your original question and all the retrieved policy snippets into one complete package—the RAG Prompt.

- This package is temporarily saved inside a Snowflake table to prepare for the next step.

Step 3: Generation (Creating the Answer) 🤖

Now, the AI model generates the final response:

- The app calls the Snowflake Arctic LLM (Large Language Model) using the prepared prompt.
- The LLM reads the instructions, analyzes the provided policy snippets, and writes a natural-sounding, concise answer, making sure to stick to the facts and include citations.
- The final text is extracted from the LLM's raw output and displayed instantly on your screen as the Answer.

This powerful RAG process ensures you always get an answer that is grounded in your organization's own data, making it highly reliable and trustworthy.