THE UNIVERSITY OF TEXAS AT DALLAS

DatabAIse

CS 4485 / Spring 2025
Department of Computer Science
Erik Jonsson School of Engineering &
Computer Science
The University of Texas at Dallas
Richardson, TX 75080, USA

Sponsors
Steven Kirtzic, PhD
Jeffrey Gordon, MS

Technical Team

Atharva Biyani
aab200015@utdallas.edu

Sneha Elangovan
sxe200017@utdallas.edu

Neha Kandula
nxk200055@utdallas.edu

Dharshini Mahesh
dxm210052@utdallas.edu

Suvel Sunilnath
sxs210292@utdallas.edu

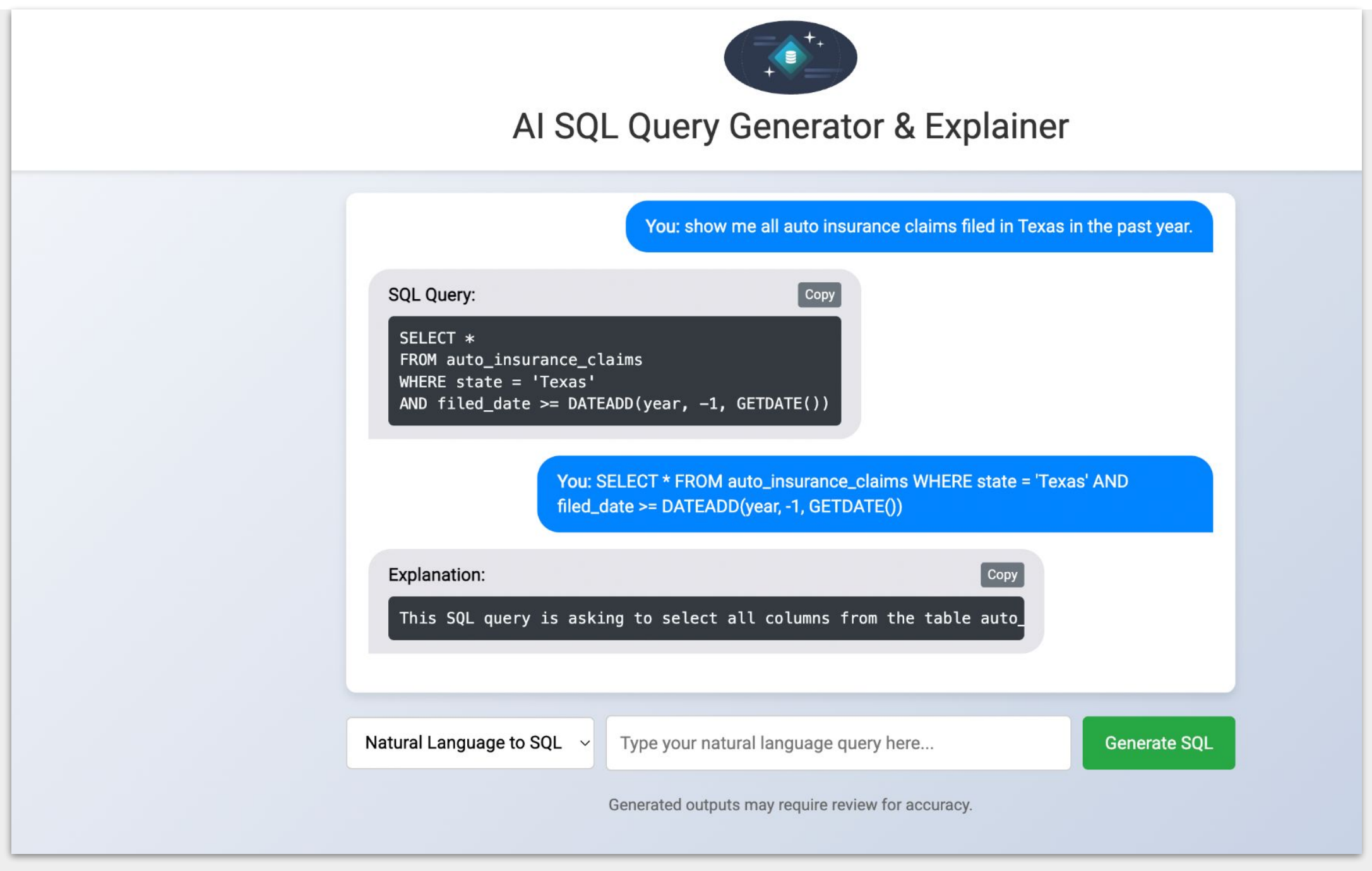
Faculty Advisor
Dr. Gopal Gupta

Abstract

Key words: Database, Natural Language Processing, Retrieval-Augmented Generation, Chatbot

Databases have become extremely complex, especially for large companies such as USAA, where issues such as redundant structures, inconsistent naming conventions, and suboptimal query patterns are becoming increasingly prevalent. The purpose of this project was to develop a user-friendly system to fix the challenges of managing and optimizing large-scale databases. The solution was achieved through two chatbots. One utilizes the Gemma-3 27b LLM with vector embeddings from Hugging Face (MiniLM-L6) that are stored in ChromaDB to get context to generate the best results according to a user's database in Snowflake. The other Chatbot converts from NLP to SQL using an OpenAI's GPT-3.5-turbo API. The project allows employees to better understand and manage their database while generating optimized queries according to their specific database.

Natural Language - SQL Conversion Chatbot


AI SQL Query Generator & Explainer

You: show me all auto insurance claims filed in Texas in the past year.

SQL Query:

```
SELECT *
FROM auto_insurance_claims
WHERE state = 'Texas'
AND filed_date >= DATEADD(year, -1, GETDATE())
```

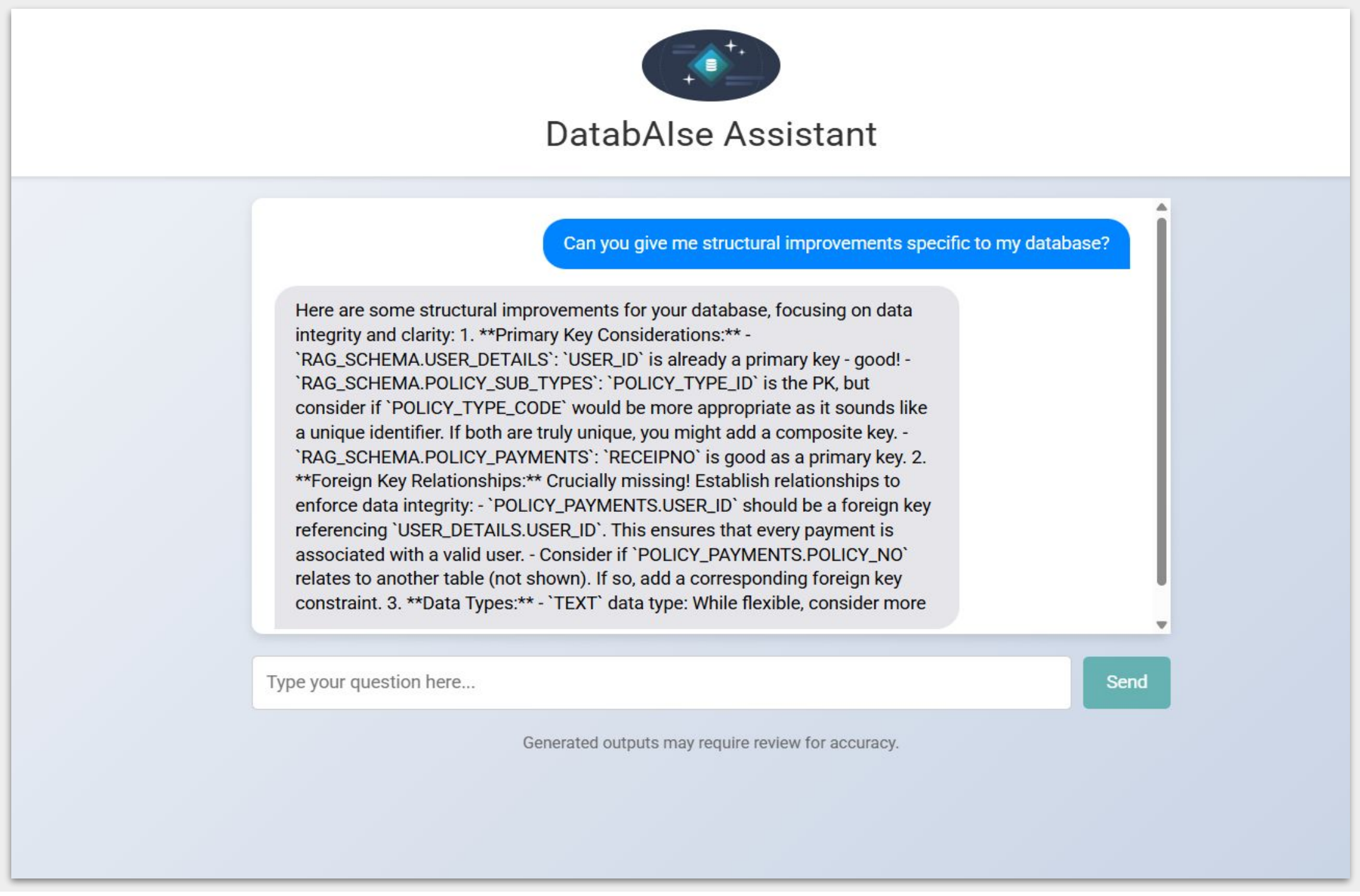
You: SELECT * FROM auto_insurance_claims WHERE state = 'Texas' AND filed_date >= DATEADD(year, -1, GETDATE())

Explanation:

```
This SQL query is asking to select all columns from the table auto_
```

Generated outputs may require review for accuracy.

RAG-LLM Chatbot


DatabAIse Assistant

Can you give me structural improvements specific to my database?

Here are some structural improvements for your database, focusing on data integrity and clarity: 1. **Primary Key Considerations:** - `RAG_SCHEMA.USER_DETAILS`: `USER_ID` is already a primary key - good! - `RAG_SCHEMA.POLICY_SUB_TYPES`: `POLICY_TYPE_ID` is the PK, but consider if `POLICY_TYPE_CODE` would be more appropriate as it sounds like a unique identifier. If both are truly unique, you might add a composite key. - `RAG_SCHEMA.POLICY_PAYMENTS`: `RECEIPNO` is good as a primary key. 2. **Foreign Key Relationships:** Crucially missing! Establish relationships to enforce data integrity: - `POLICY_PAYMENTS.USER_ID` should be a foreign key referencing `USER_DETAILS.USER_ID`. This ensures that every payment is associated with a valid user. - Consider if `POLICY_PAYMENTS.POLICY_NO` relates to another table (not shown). If so, add a corresponding foreign key constraint. 3. **Data Types:** - `TEXT` data type: While flexible, consider more

Generated outputs may require review for accuracy.

Architecture

Front-End - Flask: Used to build the web interface that allows users to interact with the DatabAIse system through natural language queries and view optimization results

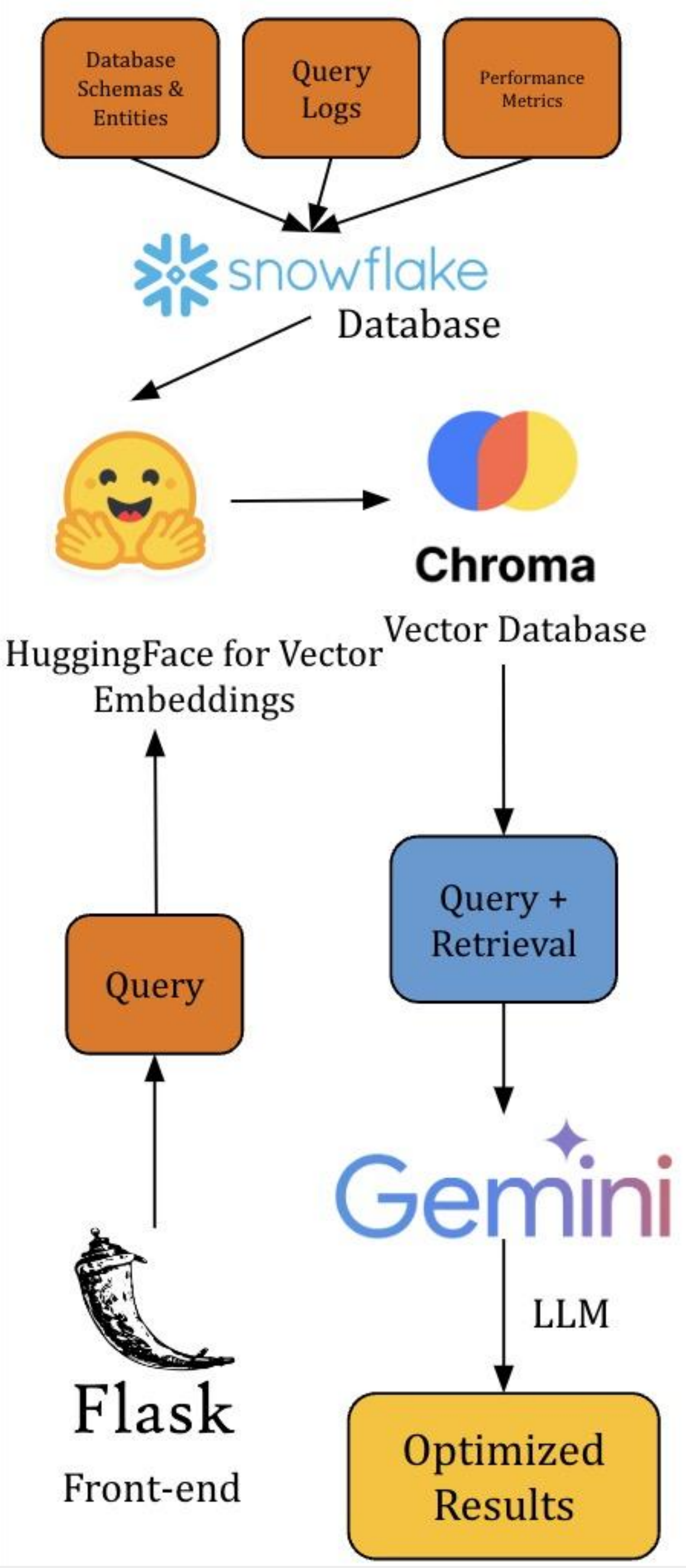
Back-End - Python: Powers the backend logic integrating LLM's, data processing pipelines, and database interactions

Vector Database - ChromaDB: Stores and embeds representations of database documentation and metadata for efficient semantic search and retrieval

Metadata Database - Snowflake: Houses the relational schema and DDLs that the system analyzes to identify structural redundancies and optimization opportunities

Large Language Model - Gemini: Serves as the core LLM to generate the optimization recommendations and SQL queries based on natural language input and retrieved context

Connector - LangChain: Connects the LLM with external data sources and tools for Retrieval-Augmented Generation and LLM workflows



Features

- ➔ Provides natural language explanations of SQL queries
- ➔ Generates SQL queries from natural language prompts
- ➔ Suggests improvements to table organization, redundancy, and naming conventions
- ➔ Recommends optimizations to reduce system complexity and improve query performance
- ➔ Enables easier database access and management without requiring SQL expertise

Performance Metrics

- Demonstrated significantly faster query optimization by using pre-embedded vectors and streamlined LLM workflows
- Successfully generated SQL Queries from diverse natural language inputs across multiple test cases
- Detected structural redundancies and inconsistent naming conventions across sample database schemas
- Enabled real time retrieval of relevant documentation and metadata using vector search with ChromaDB
- Achieved seamless integration between front-end interface and backend LLM-based optimization engine

Impact

DatabAIse transforms how large-scale DBMS are maintained and optimized in enterprise environments. By using LLM capabilities and RAG, the system can automate traditionally manual processes such as schema analysis, redundancy detection, and query optimization. This reduces system complexity and enhances data accessibility through natural language interfaces. This project promotes standardization and operational efficiency allowing organizations to make data-driven decisions faster as their systems grow.

Future Work

Future enhancements for DatabAIse involve focusing on maintaining long-term system efficiency as the data environment changes. This would include implementing data-lineage analysis to trace transformations across database objects and adding support for batch and real-time analysis to enable more dynamic optimization workflows. Lastly, ensuring version control compatibility will allow teams to track changes in schemas and recommendations over time.