**GenAI-Enabled Data Retrieval Utility for Dynamic Postgres queries Generation**

**Introduction**

In modern data platforms, efficient and adaptable data retrieval is critical to support rapid development, scalable analytics, and responsive business insights. Traditional systems rely on manually written SQL queries and Python scripts for data joins and transformation, which introduces multiple operational bottlenecks such as:

* **Slow turnaround** on data change requests
* **High dependency** on SQL/Python expertise
* **Static and hardcoded queries** that are hard to maintain
* **Time-consuming debugging** and low flexibility in data imports

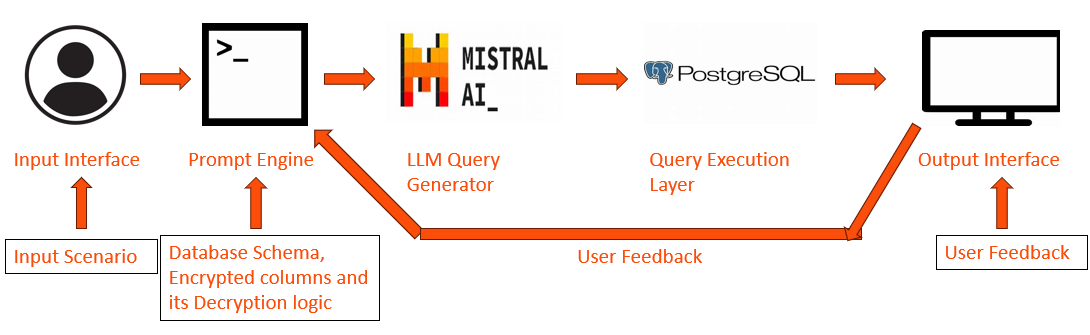
To address these challenges, we developed a **GenAI-enabled Data Retrieval Utility** powered by a **Chat-SQL Agent** that leverages the capabilities of Large Language Models (LLMs) for dynamic query generation and execution.

**Key Features**

1. **No More Complex SQL**  
   Users can input their data needs in natural language, and the utility will generate optimized SQL queries automatically.
2. **Dynamic and Flexible Imports**  
   The need to maintain hundreds of SQL templates is eliminated. SQL is generated on-the-fly per request, enabling highly dynamic data import routines.
3. **Integrated Joins and Decryption**  
   The LLM understands database schemas, keys, and encrypted fields, and auto-generates JOINs and decryption logic in a single step.
4. **Rapid Turnaround**  
   Significant reduction in development cycles — simple text changes replace long hours of SQL writing and debugging.

**Architecture Overview**

Below is the architectural flow of the utility:



* **Input Interface**: User Input - Natural Language Question or Scenario (UI/API)
* **Prompt Engine**:  Injects schema, Lists encrypted columns, Embeds decryption logic, Adds join constraints in Prompt template and incorporate User Feedback
* **LLM Query Generator**: Parses scenario, Follows prompt rules, Generates SQL SELECT
* **Query Execution Layer**: Connects to PostgreSQL, Executes generated SQL query
* **Output Interface**: Returns result (UI/API)

**Implementation Flow**

**1. Input Interface (UI/API)**

The user interacts with a simple interface and enters a natural language query like:  
*"Show me all open claims for policyholders over age 50 from the last 6 months."*

**2. Prompt Engine**

Behind the scenes, the prompt engine performs critical context injection:

* Injects schema metadata
* Identifies encrypted fields and embeds decryption rules
* Adds join constraints based on primary-foreign key mappings
* Incorporates real-time user feedback (if needed)

**3. LLM-Based Query Generator**

The enriched prompt is processed by the LLM which:

* Understands the intent of the user query
* Follows prompt constraints and schema understanding
* Constructs a valid, optimized SQL SELECT query with JOINs, WHERE filters, and decryption expressions

**4. Query Execution Layer**

The SQL is securely executed against the **PostgreSQL** database. This layer handles:

* Connection pooling
* Error handling and retries
* Audit logging of generated queries

**5. Output Interface**

The results are presented back to the user via UI or API — ready for download or further analysis.

**Example**

**Input (Natural Language):**  
*"Fetch customer transactions above $1000 made in the last 30 days, along with account type."*

**Generated SQL (by LLM):**

SELECT t.transaction\_id, t.amount, t.transaction\_date, c.customer\_name, a.account\_type

FROM transactions t

JOIN customers c ON t.customer\_id = c.customer\_id

JOIN accounts a ON c.account\_id = a.account\_id

WHERE t.amount > 1000 AND t.transaction\_date >= CURRENT\_DATE - INTERVAL '30 days';

**Execution and Result:**  
Delivered in seconds, without writing or debugging complex SQL manually.

**Conclusion**

The GenAI-powered data retrieval utility is a major leap forward in data engineering productivity. It abstracts the complexity of SQL and joins into a seamless natural language experience, enabling:

* Greater agility in responding to business needs
* Lower dependency on specialized SQL/Python resources
* Enhanced maintainability and auditability of data pipelines

By integrating prompt engineering, schema intelligence, and secure execution, this utility paves the way for **self-service data access at scale**, making data truly democratized across technical and non-technical teams.