Transforming Natural Language into SQL Queries with Generative AI:

A Comprehensive Guide to Text-to-SQL Systems

Prepared by

Ayyanar Jeyakrishnan - Principal Engineer

Abstract:

This whitepaper explores how Generative AI transforms natural language into SQL queries, enhancing data accessibility for enterprises. By democratizing data access, non-technical users can generate accurate SQL queries, boosting productivity and informed decision-making. The integration of metadata ensures precision, while multi-tenant architecture supports scalable, secure use across organizations.

Executive Summary

Generative AI revolutionizes enterprise data interaction by enabling natural language-to-SQL conversion. This whitepaper presents a detailed approach to implementing Text-to-SQL systems with advanced Large Language Models (LLMs). By integrating metadata and utilizing a multi-tenant architecture, the solution enhances data accessibility, accuracy, and security. This empowers non-technical users to generate precise SQL queries effortlessly, fostering a data-driven culture and improving overall productivity and decision-making

Business Values

- **Increased Accessibility:** Enables non-technical users to generate SQL queries from natural language prompts, reducing reliance on IT and data specialists.
- **Enhanced Productivity:** Streamlines the process of data querying, allowing faster data access and analysis, and freeing up technical resources for more strategic tasks.
- **Scalable and Secure Deployment:** Supports multi-tenant architecture with robust security measures, ensuring scalability and compliance across various organizational units.

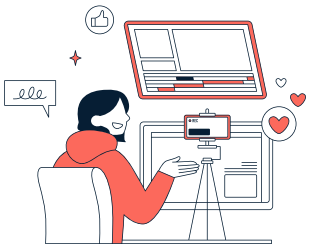
Consumers

- **Business Analysts and Executives**: Individuals who need to access and analyze data regularly but may not have extensive SQL knowledge, Data driven Decision Makers.
- **Data Scientists:** Professionals who can benefit from rapid SQL generation for data preprocessing and exploratory data analysis.
- **IT and Database Administrators:** Teams responsible for managing data access and ensuring secure and efficient database operations.

Features

- **Context-Aware SQL Generation:** Translates natural language into SQL with an understanding of context.
- **Dynamic Metadata Integration:** Incorporates up-to-date metadata from data catalogs, enhancing precision and relevance of generated SQL queries.
- **Interactive Query Refinement and Execution:** Offers real-time feedback and suggestions, allowing users to iteratively refine their queries for better outcomes and view results instantly
- **Robust Multi-Tenant Support and IAM Integration:** Ensures secure data isolation and customized access controls for different organizational units, enabling scalable deployment.
- **User-Friendly Interface:**Intuitive interface that empowers non-technical users to generate and execute SQL queries effortlessly, fostering a data-driven culture.
- **Synthetic Data Generation:** Provides capabilities to generate synthetic data for testing and development, ensuring privacy and compliance while enabling robust data simulations.

User Workflow



1

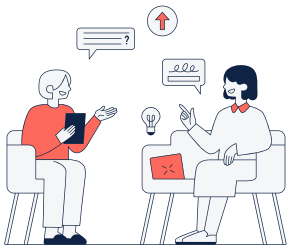
Project Selection and Database Access

Step 1: Login and Authentication: The user logs into the system using enterprise authentication (IAM). This ensures that only authorized users have access to the tool.
Step 2: Project Dashboard: Upon successful login, the user is directed to a project dashboard where they can view existing projects or create a new project.
Step 3: Project Creation: To create a new project, the user provides necessary details such as project name, description, and selects the databases they need access to. The system checks permissions and grants access accordingly.
Step 4: Database Catalog Access: The system integrates with data catalogs to display available databases and their schemas. Users select the relevant databases for their project, ensuring they have the necessary metadata for query generation.

2

User Prompt Input

- Step 5: Prompt Interface: The user navigates to the Prompt input tab within their project workspace. This interface is designed to be intuitive and user-friendly, accommodating non-technical users.
- Step 6: Entering Natural Language Queries: Users type their data queries in natural language. For example, "Show me the top 10 sales regions for last quarter."
- Step 7: Real-Time Feedback: As the user types, the system provides real-time feedback and suggestions to help refine the query. This includes autocomplete options and possible query modifications.



3

Metadata Crawling and Integration

- Step 5: Prompt Interface: The user navigates to the Prompt input tab within their project workspace. This interface is designed to be intuitive and user-friendly, accommodating non-technical users.
- Step 6: Entering Natural Language Queries: Users type their data queries in natural language. For example, "Show me the top 10 sales regions for last quarter."
- Step 7: Real-Time Feedback: As the user types, the system provides real-time feedback and suggestions to help refine the query. This includes autocomplete options and possible query modifications.

User Workflow.

4

SQL Generation

- Step 10: Processing the Query: The Generative AI model (TachYon of Gemini CodeAssist) processes the natural language query, leveraging the integrated metadata to understand the context and intent.
- Step 11: Generating SQL: The AI model translates the natural language query into a context-aware SQL statement. For example, "SELECT region, SUM(sales) FROM sales_data WHERE date BETWEEN '2023-01-01' AND '2023-03-31' GROUP BY region ORDER BY SUM(sales) DESC LIMIT 10."
- Step 12: Displaying Generated SQL: The generated SQL query is displayed to the user for review. Users can make adjustments if necessary, with further assistance from the system's real-time feedback.

- Step 13: Executing SQL: Users can execute the generated SQL query directly from the interface. The query is run against the selected databases.
- Step 14: Fetching Results: The system retrieves the query results and displays them in a tabular format within the user interface.
- Step 15: Reviewing Results: Users can review the results and, if needed, refine their queries by returning to the input step.

5

Query Execution

6

Synthetic Data Generation

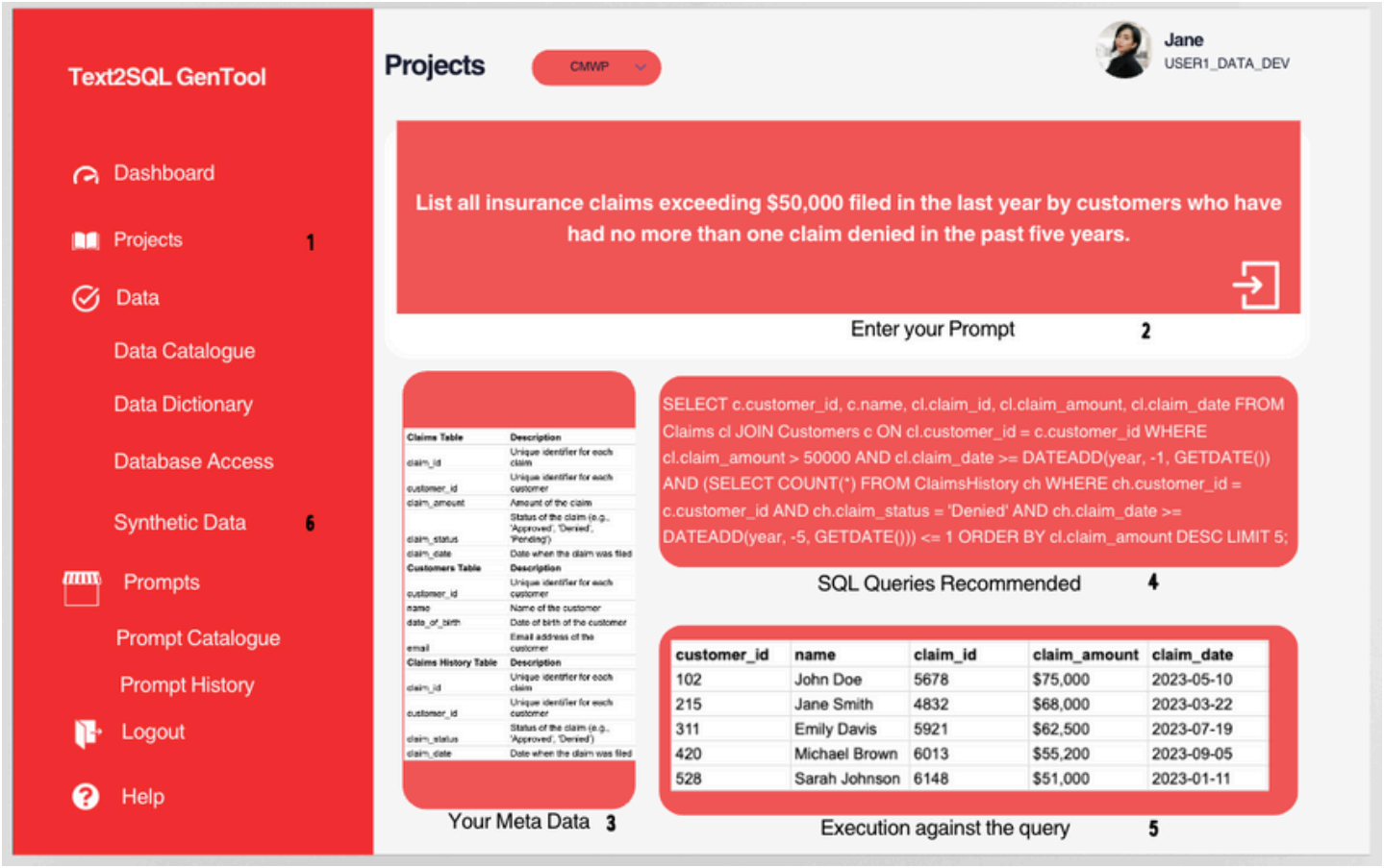
- Step 16: Synthetic Data Option: Users have the option to generate synthetic data based on their queries for testing and development purposes.
- Step 17: Generating Synthetic Data: The system uses AI algorithms to create synthetic datasets that mimic the structure and characteristics of the original data while ensuring privacy and compliance.
- Step 18: Using Synthetic Data: Synthetic data can be used for various purposes such as testing new queries, validating models, or training machine learning algorithms.

- Step 19: Iterative Refinement: Users can refine their queries iteratively based on the results and feedback provided by the system.
- Step 20: Prompt Catalog and Feedback: The system stores user prompts and feedback in a Prompt Catalog. This catalog is utilized during the review and iteration phase to offer refined suggestions and improve the overall query generation process based on previous interactions and feedbac

7

Review and Iteration

User Interface



Example Workflow

1. **Project Selection and Database Access** - Users start by selecting or creating a project from the dashboard.They gain access to the necessary database catalogs, ensuring they have the required metadata for query generation.
2. Users enter their **natural language query in the prompt input section.** Eg) "List all insurance claims exceeding $50,000 filed in the last year by customers who have had no more than one claim denied in the past five years."
3. **Metadata Integration:** The system uses metadata from data catalogs and integrates ad-hoc attachments from the user interface. Metadata information is displayed, showing the structure and schema of the databases involved.
4. **SQL Generation:** Based on the user prompt, the system generates a corresponding SQL query leverage Large Language Model
5. **Execution against the Query: (optional - Provided we get access to Database**) - The system executes the generated SQL query against the database.Results are displayed, showing the top five rows matching the query criteria.
6. **Review and Iteration:** Users can iteratively refine their queries based on feedback and results, improving the precision and relevance of their data insights and finally store their prompt in Prompt Catalogue. Your history of Prompt will be monitored under Prompt History. In case if you want to create a Synthetic data you can run against the Data / Prompt Catalogue and store the data in Short lived Manner.

Architecture

- Prompt Store
- Prompt Catalogue
- Synthetic Data with TTL

mongoDB

7

- Throttling
- Enterprise Security
- Request/Response

Apigee

6

LLM to Process the Prompt and Generate SQL

Gemini

Prompt storage

4

5

8

User Interace

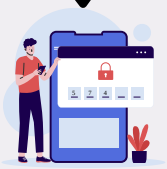
3

Data Sources

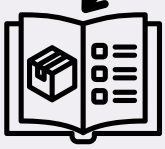
1. Data Catalogue
2. Data Dictionary (Adhoc Add via UI)
3. Any BigData (Structured)
4. RDBMS (Optional Only if to Execute the SQL)

1

2



Enterprise Authentication for Login to UI



Data Dictionary and Catalogue via IAM

Database Password - Integration to Vault

Authentication and Authorization Service

Internal

External

Detailed Workflow for Text-to-SQL Architecture

1. User Authentication and Authorization

The process begins with user authentication and authorization, ensuring secure access to the system. Users log into the interface using enterprise authentication integrated with Identity and Access Management (IAM). This step ensures that only authorized users can access the tool, maintaining the security and integrity of the data and the system.

2. Data Source Integration

Once authenticated, the system accesses various data sources to gather necessary metadata and schema information. This involves retrieving the data dictionary and catalog via IAM, which provides structured metadata. Additionally, the system integrates with a vault system for secure management of database credentials. The data sources include data catalogs, data dictionaries (with ad-hoc additions via the user interface), structured big data, and optionally, relational databases (RDBMS) or any structured database for executing SQL queries.

3. User Interface Interaction

Users interact with a user-friendly interface to receive natural language queries as a Prompt. The interface is designed to be intuitive, providing real-time feedback by executing queries against the Metadata selection and executing against the database to help users refine their queries. This step ensures that even non-technical users can easily create and refine their data queries without needing in-depth knowledge of SQL.

4**. Metadata Parsing and Integration**

Upon receiving the user query, the system uses a Large Language Model (LLM) to interpret the natural language prompt. It then initiates metadata parsing to gather the necessary schema information. This metadata is dynamically integrated with existing data catalogs and any ad-hoc metadata provided by users through the interface. By leveraging Retrieval-Augmented Generation (RAG) techniques, the system augments the user prompt with the most accurate and up-to-date metadata, ensuring the generated SQL queries are highly relevant and precise.

5. Prompt Storage

User prompts and feedback are stored in a prompt catalog within MongoDB. This storage mechanism allows the system to maintain a history of user interactions, which can be used to improve query refinement and provide iterative enhancements. By leveraging this prompt catalog, the system can learn from previous queries and feedback to offer more precise suggestions and results over time.

6. SQL Generation

The natural language query is then processed by Google's Gemini Pro Model, accessed via the API Gateway (Apigee). The API Gateway manages the communication, ensuring secure and efficient request/response handling. Google's Gemini Pro Model interprets the user Prompt Augmented with the Metadata parsed and generates a context-aware SQL statement, to accurately translate user intent into SQL.

7. Throttling and Security

To manage system performance and security, the API Gateway handles throttling, enterprise security, and request/response management. This ensures that the system can handle multiple queries efficiently while maintaining robust security protocols. The gateway's role is crucial in managing the load and protecting the system from potential security threats, ensuring smooth and secure operations.

8. Query Execution and Results Display

Finally, the generated SQL query is executed against the database. The results are fetched and displayed in the user interface, allowing users to review and analyze the data. Users can iteratively refine their queries based on the results and feedback provided by the system. This iterative process helps users to continually improve the accuracy and relevance of their data queries, leading to more precise and actionable insights.

This detailed workflow outlines how the system leverages Google's Gemini Pro Model for generating SQL from natural language queries, ensuring a secure, efficient, and user-friendly solution for enterprise data analysis.

Strategies for Security and Deployment

1) User Interface

The user interface for the Text-to-SQL generation system is developed using the React stack, providing a responsive and interactive experience. It allows users to easily select or create projects, input natural language queries, and receive real-time feedback for query refinement. Designed for user-friendliness, the interface ensures that even non-technical users can effortlessly generate and execute SQL queries, enhancing productivity and making data analysis accessible to a broader audience.

Security Around Prompt and Response Handling

To ensure the integrity and security of prompts and responses, the system employs robust guardrails using tools like NemoGuardrails or LLAMAGuard. These guardrails intercept and validate both incoming prompts and outgoing responses, ensuring only valid SQL commands are generated. This prevents security risks such as prompt injection attacks and filters out any attempts to include Personally Identifiable Information (PII) or generate toxic content, thereby protecting sensitive data and maintaining a safe output environment.

Metadata and Prompt Storage

All metadata and user prompts are securely stored within an internally deployed MongoDB instance, configured with access controls and encryption. Ad-hoc metadata uploads are stored in designated filesystem paths with restricted access. Centralizing metadata storage in MongoDB ensures faster retrieval and processing, enhancing overall performance and efficiency.

Prompt storage is handled meticulously to facilitate iterative query refinement and improve system intelligence. User prompts and feedback are cataloged within MongoDB, allowing the system to maintain a comprehensive history of interactions. This prompt catalog is leveraged to refine future queries, ensuring more precise and relevant results. By combining secure storage practices with a robust prompt catalog, the system protects user data while continuously enhancing its SQL query generation capabilities.