Design and Development of an AI-Powered Data Insight Chatbot Using Python and SQL

21CSA697A

Final Report

Submitted by:

KALYAN MAJEE

(AA.SC.P2MCA2107498)

in partial fulfilment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

February, 2026

Acknowledgement

I express my sincere gratitude to my project guide, faculty members, and department for their valuable guidance and support. I also thank my friends and family for their encouragement throughout the completion of this project.

Abstract

This project addresses the barrier between non-technical users and relational databases. By leveraging Large Language Models (LLMs) specifically the `llama-3.3-70b-versatile` model via Groq, we developed a system that converts natural language questions into executable SQL queries. The application uses LangChain for orchestration, managing conversation history and database schema injection to provide context-aware responses. The final product is a Streamlit-based interface that allows users to interact with MySQL databases without writing a single line of SQL code.

Chapter 1

1. Introduction

1.1 Background

In the modern corporate landscape, data is the primary driver of strategic decisions. However, accessing this data traditionally requires knowledge of Structured Query Language (SQL), creating a bottleneck where non-technical stakeholders must rely on data analysts for even simple reports. This project addresses this gap by utilizing Large Language Models (LLMs) to create a "Natural Language to SQL" interface. By translating human speech into precise database queries, we democratize data access within an organization.

1.2 Proposed Layout and Tools

The project follows a modular layout designed to ensure security, accuracy, and ease of use:

- **Groq Inference Engine:** Utilized to run the `llama-3.3-70b-versatile` model at high speeds, ensuring real-time conversational responses.
- **LangChain Framework:** Acts as the "brain" of the operation, managing the `ChatPromptTemplate` and `SQLDatabase` utilities to chain the user's question with the database schema.
- **Streamlit:** Provides a lightweight, web-based frontend for user interaction and database configuration.
- **MySQL:** Serves as the relational database management system (RDBMS) containing the target data.

1.3 Objectives and Scope

- **Background and Motivation:** To eliminate the technical barrier between users and their data by leveraging the reasoning capabilities of Llama 3.3.
- **Problem Statement:** Manual SQL generation is time-consuming and prone to syntax errors for non-experts.
- **Objectives:** To build a robust system that can understand schema context, generate valid SQL, execute it safely, and explain the results in natural language.
- **Scope:** The project is scoped for MySQL databases and focuses on query-based data retrieval (SELECT statements) rather than destructive operations (DELETE/DROP).

Chapter 2
2. Literature Review / Background Study

Existing Text-to-SQL methods often struggle with complex joins or schema ambiguity. Recent advancements in LLMs, like the Llama series, have shown superior performance in zero-shot SQL generation. This project utilizes LangChain's SQLDatabase utility to provide the LLM with the necessary metadata to overcome these limitations.

Chapter 3
3. System Design / Architecture

3.1 Architecture Overview

The system is designed around a "Double Chain" architecture. The first chain (SQL Generation) focuses on technical accuracy, while the second chain (Response Generation) focuses on human readability.

3.2 Module Descriptions

- **Prompt Engineering Module:** Uses a template that strictly instructs the LLM to write "only the SQL query and nothing else" to prevent code execution errors.
- **Schema Extraction Module:** Dynamically pulls table info using `db.get_table_info()`, allowing the model to understand foreign keys and column names without hard-coding them.
- **Context Management:** The `chat_history` is passed through each request, allowing the user to ask follow-up questions (e.g., "Who are they?" after asking "Which 3 artists have the most tracks?").

3.3 Data Flow and Algorithms

1. **Input:** User enters a query via Streamlit `chat_input`.
2. **Processing:** The system assigns the current database schema to the prompt.
3. **Execution:** The generated SQL is sent to the MySQL instance via SQLAlchemy.
4. **Formatting:** The LLM receives the raw database rows and "translates" them back into a sentence based on the original user question.

Chapter 4
4. Implementation Details

4.1 Module Implementation

- **Database Connection:** Handled via `init_db` function using SQLAlchemy URIs.
- **SQL Generation Chain:** Utilizes a `ChatPromptTemplate` that injects the database `<SCHEMA>` and `<chat_history>`to produce a raw SQL string.
- **Natural Language Chain:** Takes the raw SQL output and the database result, then formats it back into a human-readable answer.

Category	Component	Specification
Hardware	Processor	Minimum Quad-core CPU (i5/Ryzen 5)
	RAM	8GB or higher
Software	OS	Windows 10/11, macOS, or Linux
	Language	Python 3.9+
	Libraries	langchain, langchain-groq, streamlit, pymysql
	Database	MySQL 8.0+

Chapter 5
5. Testing, Validation & Results

5.1 Testing Methods

Functional testing was performed by asking questions like "which 3 artists have the most tracks?" and verifying the generated SQL against manual results.

5.2 Results

The system successfully connects to `kalyan_db` and generates accurate queries for basic and intermediate complexity SQL tasks (aggregations, counts, and limits).

Chapter 6
6. Conclusion and Future Work

The project demonstrates that Groq-powered LLMs can effectively serve as virtual data analysts.

- **Limitations:** Complex nested subqueries might occasionally require prompt tuning.
- **Future Work:** Integration of data visualization (graphs) within the Streamlit chat and support for NoSQL databases.

Chapter 7
7. References

[1] LangChain Documentation, "SQL Database Integrations," 2025.

[2] Meta AI, "Llama 3 Model Card," 2024.

[3] Groq Cloud Documentation, "Llama-3.3-70b API Reference," 2025.

Chapter 8
8. Appendix (Optional)

Github link : https://github.com/AASCP2MCA2107498/kalyan-github-repo/tree/main/

Sample Pseudo-code:

Python

```
# SQL Chain Logic
prompt = ChatPromptTemplate.from_template(template)
llm = ChatGroq(model="llama-3.3-70b-versatile")
chain = RunnablePassthrough.assign(schema=get_schema) | prompt | llm |
StrOutputParser()
```

Date : 28-Jan-2026
Student Name and Signature : Kalyan Majee

Name and Signature of the Evaluator.
Date