A PROJECT REPORT ON

Conversational Database Interaction Using Lang Chain and NLP

Submitted in partial fulfillment of the requirements for the award of the degree.

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (Data Science)

Under the guidance of

Mrs. R. Ramya

Assistant Professor / Computer Science and Engineering (Data science)

BY

C. JYOSHITHA **21751A3211**

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES, CHITTOOR-517127, A.P.
(Autonomous)
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

(2025)

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES, CHITTOOR-517127, A.P.
(Autonomous)
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

BONAFIDE CERTIFICATE

This is to certify that the project work entitled "Conversational Database Interaction Using Lang Chain and NLP**"** is a genuine work of

C. JYOSHITHA **21751A3211**

Submitted to the department of Computer Science and Engineering – Data Science, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (DATASCIENCE)** from Sreenivasa Institute of Technology and Management Studies, Chittoor, A.P.

Signature of the Supervisor
Mrs. R. Ramya
Assistant Professor,
Department of CSE - DS
Sreenivasa Institute of Technology and Management Studies, Chittoor, A.P.

Signature of the Head of Department
Mr. A. Srinivasan
Associate Professor & HOD (CSE – DS),
Department of CSE - DS
Sreenivasa Institute of Technology and Management Studies, Chittoor, A.P.

Submitted for Examination (Viva-Voce) held on……………………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

VISION AND MISSION

INSTITUTE VISION:

To emerge as a Centre of Excellence for Learning and Research in the domains of engineering, computing and management.

INSTITUTE MISSION:

IM1: Provide congenial academic ambience with necessary infrastructure a learning resource.

IM2: Ignite the students to acquire self-reliance in state-of –the-Art technologies.

IM3: Inculcate confidence to face and experience new challenges from industry and society.

IM4: Foster enterprising spirit among students.

IM5: Work collaboratively with Technical Institutes / Universities / Industries of National, International repute.

DEPARTMENT VISION:

To train students to become competent Data analytic experts and expand their capacity to contribute in the Field of data science by providing solution in public aspects.

DEPARTMENT MISSION:

DM 1: To develop professionals who are skilled in the area of Data analytics.

DM 2: To teach quality and value-based education and contribute towards the innovation of computing, expert system, Data Science to raise satisfaction level of all stakeholders.

DM 3: To integrate research into practical, relevant solutions to address business and societal challenges.

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES, CHITTOOR-517127, A.P.
(Autonomous)
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

PROGRAMME EDUCATIONAL OBJECTIVES (PEO's):

PEO1: Excel in Computer Science and Engineering program through quality studies, enabling succession computing industry. (Professional Competency).

PEO2: Surpass in one's career by critical thinking towards successful services and growth of the organization, or as an entrepreneur or in higher studies. (Successful Career Goals).

PEO3: Enhance knowledge by updating advanced technological concepts for facing the rapidly changing world and contribute to society through innovation and creativity (Continuing Education and Contribution to Society).

PROGRAM SPECIFIC OUTCOMES (PSO's):

PSO1: Have Ability to understand, analyses and develop computer programs in the areas like algorithms, system software, web design, big data analytics, and networking.

PSO2: Deploy the modern computer languages, environment, and platforms in creating innovative products and solutions.

PROGRAMME OUTCOMES (PO's):

PO1 - Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2 - Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3 - Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

PO4 - Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 - Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6 - The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 - Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8 - Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 - Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10 - Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 - Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 -Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES, CHITTOOR-517127, A.P.
(Autonomous)
(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

<u>Course Outcomes for project work</u>

On completion of project work we will be able to,

CO1. Demonstrate in-depth knowledge on the project topic. (PO1)

CO2. Identify, analyze and formulate complex problem chosen for project work to attain substantiated conclusions. (PO2)

CO3. Design solutions to the chosen project problem. (PO3)

CO4. Undertake investigation of project problem to provide valid conclusions. (PO4)

CO5. Use the appropriate techniques, resources and modern engineering tools necessary for project work. (PO5)

CO6. Apply project results for sustainable development of the society. (PO6)

CO7. Understand the impact of project results in the context of environmental sustainability. (PO7)

CO8. Understand professional and ethical responsibilities while executing the project work. (PO8)

CO9. Function effectively as individual and a member in the project team. (PO9)

CO10. Develop communication skills, both oral and written for preparing and presenting project report. (PO10)

CO11. Demonstrate knowledge and understanding of cost and time analysis required for carrying out the project. (PO11)

CO12. Engage in lifelong learning to improve knowledge and competence in the chosen area of the project. (PO12)

SREENIVASA INSTITUTE OF TECHNOLOGY AND MANAGEMENT STUDIES, CHITTOOR-517127, A.P.

(Autonomous)

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

CO – PO MAPPING

COs\POs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PS01	PSO2
CO.1	3												3	3
CO.2		3											3	3
CO.3			3										3	3
CO.4				3									3	3
CO.5					3								3	3
CO.6						3							3	3
CO.7							3						3	3
CO.8								3					3	3
CO.9									3				3	3
CO.10										3			3	3
CO.11											3		3	3
CO.12												3	3	3
COs	3	3	3	3	3	3	3	3	3	3	3	3	3	3

ACKNOWLEDGEMENT

A Project of this magnitude would have not been possible without the guidance and co-ordination of many people. I am fortune in having top quality people to help, support and guide us in every step towards our goal.

Our team is very much grateful to the Chairman **Sri K. Ranganatha Garu,** for his encouragement and stalwart support. We are also extremely indebted to the Secretary **Sri D.K. Badri Narayana Garu,** for his constant support.

We express our sincere thanks to our Academic Advisor **Dr. K.L. Narayana**., further, we would like to express our profound gratitude to our principal **Dr.N.Venkatachalapathi, B.Tech., MTech., Ph.D., PGDPE(CIPET).,**

PGDIRPM., F.I.E., for providing all possible facilities throughout the completion of our project work.

We express our sincere thanks to our Dean (Academics), **Dr.M. Saravanan, MTech, Ph.D.,** further we express our sincere thanks to our Head of the Department

Mr. A. Srinivasan, MTech, for his co-operation and valuable suggestions towards the completion of project work.

We express our sincere thanks to our guide **Mrs. R. Ramya., MTech, (Ph.D.),** for offering us the opportunity to do this work under her guidance.

We express our sincere salutation to all other teaching and non-teaching staff of our department for their direct and indirect support given during our project work. Last but not the least, we dedicate this work to our parents and the Almighty who have been with us throughout and helped us to overcome the hard times.

C. JYOSHITHA **21751A3211**

DECLARATION

I certify that

- The work contained in this report is original and has been done by me under The Guidance of my supervisor.

- The work has not been submitted to any other Institute for any degree or diploma.

- I have followed the guidelines provided by the Institute in preparing the report.

- I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

- Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further,

  I have taken permission from the copyright owners of the sources,

  whenever necessary.

C. JYOSHITHA **21751A3211**

ABSTRACT

In the era of AI-driven data accessibility, natural language processing (NLP) is revolutionizing how users interact with databases. This project explores the integration of Lang Chain with SQL databases to facilitate conversational data retrieval through natural language queries. By leveraging Lang Chain's advanced language models, the system translates user queries into structured SQL statements, eliminating the need for SQL expertise and enabling intuitive data exploration. For demonstration, the project utilizes the Chinook dataset, a sample database modeled after a digital media store, containing structured information about customers, employees, artists, albums, tracks, invoices, and sales transactions. This dataset is ideal for simulating real-world applications such as e-commerce analytics, customer insights, and business intelligence. The system is implemented with SQLite but is designed to scale seamlessly to more robust database management systems like MySQL or PostgreSQL. Users interact via a chat-based interface, querying the dataset with questions like "List all albums by The Beatles" or "Show the top 5 customers by total purchases". Lang Chain processes these requests by dynamically generating SQL queries, retrieving the relevant data, and presenting the results in a user-friendly format. This approach enhances accessibility, enabling business professionals, analysts, and non-technical users to extract valuable insights effortlessly. By combining natural language understanding with database management, the project lays the foundation for AI-powered data exploration, streamlining decision-making processes and democratizing access to structured information.

Keywords: Lang Chain, SQL databases, Natural language queries, Structured SQL statements, Conversational data retrieval, Chinook dataset, MySQL, Dynamic SQL query generation, Chat-based interface, User-friendly data presentation, AI-powered data exploration, Decision-making enhancement

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Objective

The main objective of this project is to create a system that allows users to interact with a database using natural human language. Instead of writing SQL queries, users can simply ask questions in plain English and get meaningful results. This approach aims to simplify the process of data retrieval and make it more accessible to people from non-technical backgrounds.

Another goal is to integrate powerful tools like Lang Chain and NLP to understand and convert user input into accurate SQL commands. By using these technologies, the system can intelligently interpret what the user is asking, generate the correct database query, and present the data in a user-friendly format. This allows the system to act as a smart interface between humans and structured data.

Overall, the objective is to break down the barrier between users and databases, reducing the dependency on manual SQL knowledge. This can be especially helpful in business environments where quick access to data is essential but not everyone has the technical skills to retrieve it using traditional methods.

- Business Intelligence (BI) Tools – Software such as Tableau, Power BI, and Looker provide data visualization capabilities but depend on structured input and SQL queries.
- Predefined Reports & Dashboards – Many organizations rely on static reports, which require IT or database administrators to modify them for new insights.

1.2 Problem Identification

In many organizations, a lot of valuable information is stored in databases, but accessing it requires knowledge of SQL or help from a technical expert. This creates a gap between the data and the people who need it the most, such as business managers, sales teams, or support staff. Not everyone has the time or skills to write SQL queries, which slows down decision-making and reduces efficiency.

Even simple tasks like checking total sales, finding customer details, or generating reports require writing queries, which can be intimidating for non-technical users. This creates a constant dependency on developers or data analysts, which not only wastes time but also increases workload for technical teams. In fast-paced environments, this delay can impact productivity and responsiveness.

There is a clear need for a system that allows users to communicate with databases in their own language, without needing to learn SQL. By identifying this problem, this project focuses on building a natural language interface that acts as a bridge between the user and the database. This can help organizations save time, empower users, and make data-driven decisions faster.

1.3, Literature Survey

Natural Language to SQL (NL2SQL) systems has evolved from early rule-based methods like PRECISE and NALIR, which relied on grammatical parsing and predefined syntax rules. While effective in limited domains, these systems required extensive manual effort to maintain, making them impractical for large, evolving databases [1]. To address this, machine learning-based models such as SQL Net and Rat-SQL were introduced, learning from large datasets to improve SQL accuracy. However, these models still struggled with complex query structures like multi-table joins and nested conditions [1].

The introduction of Large Language Models (LLMs) like GPT-3 significantly advanced NL2SQL systems by enabling few-shot and zero-shot learning, reducing the need for extensive labelled data. Techniques such as few-shot learning and Chain-of-Thought Prompting allowed LLMs to break down complex queries into logical steps, improving SQL generation. Despite these improvements, LLM-based models still faced challenges with ambiguous queries, domain-specific constraints, and maintaining context in multi-turn conversations [2][3].

Lang Chain offers a hybrid solution by integrating LLMs with dynamic database querying mechanisms. By combining modular pipelines, conversational memory, and context-aware query generation, Lang Chain enhances SQL accuracy while simplifying interactions for non-technical users. This approach integrates rule-based heuristics with LLM-driven techniques, making database access more intuitive for enterprise applications. Future research should focus on improving explainability, domain adaptability, and scalability to further refine NL2SQL systems [2].

CHAPTER 2

SYSTEM ANALYSIS

2.1 Existing System

Existing database systems mainly rely on writing SQL queries or using BI tools like Power BI and Tableau to retrieve data. These methods require users to understand SQL syntax, table structures, and database schema. While GUI-based tools and dashboards help visualize data, they still depend on predefined settings and need technical knowledge or support to modify or extract new insights.

2.1.1 Demerits

1 **Manual SQL Querying** – Users must manually write and execute SQL queries in database management tools.

2 **Graphical User Interfaces (GUIs) for Databases** – Some platforms provide GUI based query builders, but they still require an understanding of relational database concepts.

3 **Business Intelligence (BI) Tools** – Software such as Tableau, Power BI, and Looker provide data visualization capabilities but depend on structured input and SQL queries.

4 **Predefined Reports & Dashboards** – Many organizations rely on static reports, which require IT or database administrators to modify them for new insights.

By addressing these challenges, the proposed AI-driven natural language querying system bridges the gap between structured databases and human interaction, making data retrieval more efficient, user-friendly, and accessible

2.2 Proposing System

The proposed system leverages Lang Chain and MySQL to create an AI-powered conversational interface that allows users to query databases using natural language. Instead of writing complex SQL queries, users can simply input questions in plain English, and the system will automatically generate and execute the corresponding SQL query.

These 3 systems eliminate the need for technical expertise, making data access more intuitive and efficient.

2.2.1 Merits

1. **AI-Driven Query Generation** – Uses NLP models to interpret user queries and generate structured SQL statements.
2. **Real-Time Data Retrieval** – Provides immediate responses by executing SQL queries dynamically.
3. **User-Friendly Interface** – Enables seamless interactions without requiring SQL knowledge.
4. **Scalability & Flexibility** – Can be integrated with various databases like MySQL, PostgreSQL, and SQLite.
5. **Error Handling & Security** – Implements query validation and security checks to prevent SQL injection.

2.3 Feasibility Study

2.3.1 Technical Feasibility

This project is technically possible because it uses tools and technologies that are already available and well-supported. Lang Chain helps connect natural language to structured databases, and Python makes it easy to build and test the system. We can also use MySQL or any other standard database without needing special hardware or expensive software.

All the components used in this project—like NLP models, database connectors, and APIs—work well together and can be set up even by a student or developer with basic knowledge. Tutorials and documentation are also easy to find online, which makes the development and testing process smooth and achievable.

2.3.2 Operational Feasibility

The system is easy to use for both technical and non-technical users. Instead of learning how to write complex SQL queries, users can just ask their questions in plain English and get the results. This makes it especially helpful in workplaces where people need quick data but don't have the time or skills to write queries themselves.

By making data more accessible, the system can improve how teams work. It reduces the need to depend on IT staff for small data requests, which saves time and effort. It fits well into the way most organizations work today—focusing on speed, simplicity, and self-service access to information.

2.3.3 Economic Feasibility

This project is cost-effective because most of the tools used are free or open-source. Python, MySQL, and Lang Chain can all be used without paying any license fees. If APIs like OpenAI's GPT are used, there may be a small cost based on how often they are used, but it can be controlled depending on the budget.

When compared to the cost of hiring extra technical staff or training employees in SQL, this solution saves money in the long run. It also increases productivity, which brings more value to the organization. So overall, it's a smart investment for anyone who wants to make data easier to access without spending too much.

CHAPTER 3

SYSTEMS DEVELOPMENT MODEL

1. Planning and Requirements Gathering

- Problem Identification: Users struggle with writing SQL queries. Our goal is to let users ask questions in plain English and still get accurate data from a database.
- Requirements Collection: We talked to potential users and identified that the system should understand natural language, connect to a database, and return results clearly.
- Scope Definition & Planning: We planned to include core features like natural language input, AI processing, and SQL execution. A basic project timeline was created to manage tasks.

2. Analysis and Risk Assessment

- Requirement Analysis: We broke down the needs into clear technical goals—like using Lang Chain with OpenAI to convert text into SQL and integrating MySQL.
- Risk Identification & Management: Risks included wrong SQL generation, vague queries, and API errors. We reduced these by adding logs, input checks, and clear prompts.

3. Design and Prototype

- System Design: We planned the architecture with three parts: user input, language model processing, and database connection.
- Low-Level Planning: We designed flow diagrams and pseudocode to understand how the system would work behind the scenes.
- Prototype Testing: A small working version was built to test basic queries and check if the flow from user input to SQL execution worked correctly.

4. Implementation and Testing

- Module Development: We built the main parts of the system—Lang Chain setup, OpenAI integration, user interaction, and MySQL connection.
- Testing & Debugging: We tested with different types of questions, fixed errors, and improved the accuracy of query conversion and results.
- Feedback Collection: Feedback from users helped us improve clarity and handle wrong or confusing inputs better.

5. Evaluation and Improvement

- System Review: The system was good with simple and direct queries but needed guidance for more complex ones.
- Improvements Made: We improved user instructions, added safe error handling, and made the responses more user-friendly.
- Final Testing: After final testing, the system was stable, reliable, and ready for future improvements like adding a graphical interface.

CHAPTER 4

SYSTEM REQUIREMENT

4.1 Hardware Requirement

- Processor: Intel Core i5 or higher (or equivalent)
- RAM: Minimum 8GB (Recommended: 16GB for better performance)
- Storage: Minimum 20GB free space (Recommended: SSD for faster query execution)
- GPU (Optional): Required for advanced AI model processing (e.g., NVIDIA CUDA-enabled GPU)

4.2. Software Requirements:

- Operating System: Windows 10/11, macOS, or Linux
- Programming Language: Python (Version 3.8 or later)
- Database Management System (DBMS): MySQL (Preferred) / PostgreSQL / SQLite
- Development Environment: Jupyter Notebook / VS Code / PyCharm
- Dependency Management: pip or conda for package installation

4.3. Required Libraries & Frameworks:

- Lang Chain: For natural language processing and SQL query generation
- MySQL Connector (mysql-connector-python): To connect Python with MySQL
- SQL Alchemy: For database interaction
- NLTK / SpaCy: For NLP processing (if required)
- OpenAI API / LlamaIndex (Optional): For advanced language model interactions
- Flask / FastAPI (Optional): For building a web interface for the system

CHAPTER 5

SYSTEMS DESCRIPTION

5.1 Problem Description

In today’s world, data plays a major role in decision-making. However, accessing that data from a database often requires knowledge of SQL and an understanding of how databases are structured. This creates a gap between non-technical users and the information they need. For someone who doesn't have a background in programming or databases, even asking a simple question like “What were the sales in March?” can become a complicated task. This results in delays and over-dependence on technical teams just to retrieve basic information.

Even though there are tools like Power BI, Tableau, or database GUIs, they still require users to have some idea of how tables are related or how to apply filters and joins. These tools are useful but not always friendly for a person who just wants a quick answer without going through menus, filters, or learning database logic. Relying on predefined reports and dashboards means users can only get the data the report creator assumed they would need, and anything new or different requires support from IT or data analysts.

The main problem this project aims to solve is this barrier between users and their data. By allowing users to simply ask questions in plain English, the system removes the need for SQL knowledge. This makes data interaction easier, faster, and more inclusive. Whether it’s a manager, a marketer, or a student, anyone should be able to ask a question and instantly get an accurate answer — and this project makes that possible by using the power of natural language processing and Lang Chain.

5.2 Overview of system

This system is designed to be a smart interface that connects people with databases in a much easier way. Instead of writing complicated SQL queries, users can just type what they want in plain English, and the system will do the rest. For example, someone can ask “List all customers from Hyderabad,” and the system will figure out the user's intention, convert that into a SQL query, and fetch the correct data from the database. This reduces the need for any technical skill and gives direct access to information.

To do this, the system uses Natural Language Processing (NLP) techniques and Lang Chain, which help the machine understand what the user is trying to ask. Lang Chain works with language models to process the sentence, identify the keywords like table names or columns, and then generates the most suitable SQL query for that sentence. Once the query is generated, it is sent to the backend database, where the results are fetched and shown to the user in a clean and easy-to-read format.

This kind of system can be used in companies, colleges, hospitals, or anywhere that works with databases. It helps people get the answers they need without needing to wait for someone else to write the query. It also saves time, reduces the chances of errors, and allows users to focus more on decision-making rather than figuring out how to access data. Overall, it makes working with data feel like having a conversation, just like asking a question to a human expert.

5.3 System Architecture Diagram

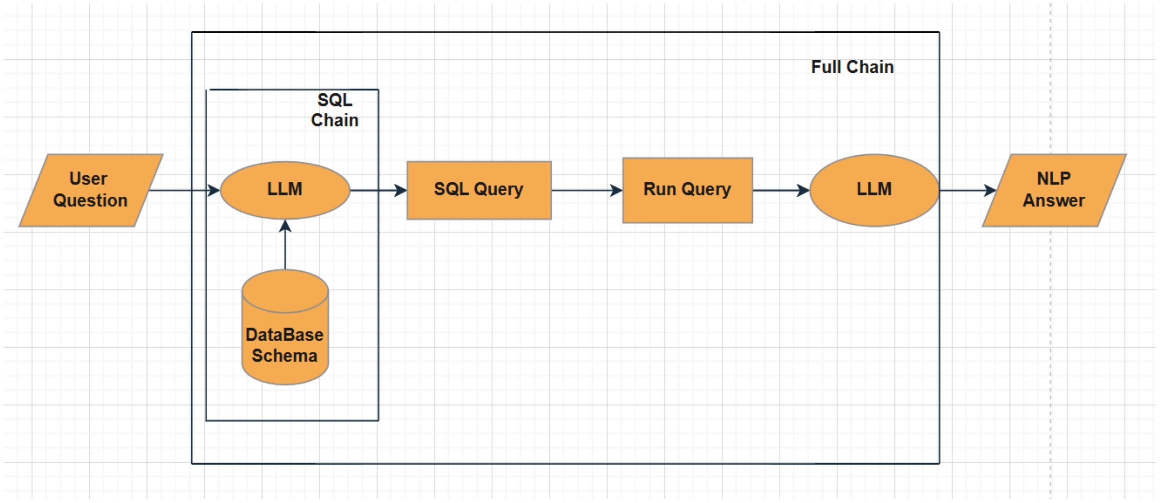


Figure 5.3 System Architecture Diagram

1. User Question - Everything starts when a user asks a question in natural language. This could be something like "What are the total sales for last month?" The user doesn't need to know SQL or how the database is structured.

2. Full Chain - This part is responsible for converting the user's question into an SQL query that the database can understand. It happens in the following steps:

- LLM (Large Language Model): The LLM takes the user’s question and, using its knowledge of language and databases, generates an SQL query.
- Database Schema: The LLM refers to the database schema, which contains details about tables, columns, and relationships. This helps it create an accurate SQL query.
- SQL Query: The system then produces a properly structured SQL query that can be executed on the database.

3. Run Query - Once the SQL query is generated; it is sent to the database for execution. The database runs the query and retrieves the requested information.

4. Full Chain - Now that we have the raw data from the database, it needs to be converted into a user-friendly answer:

- LLM: The retrieved data is processed by the LLM, which converts it into a well-structured response in natural language.

5. NLP Answer - Finally, the user receives a clear, readable answer instead of raw database results. For example, instead of showing a table of numbers, the system might say:

"The total sales for last month were $50,000."

CHAPTER 6

SYSTEM DESIGN

6.1 Module Description

6.1.1 Installation of Required Tools Module

Installing MySQL

MySQL is a relational database management system that stores and manages structured data. To install it:

1. Download MySQL
    - Visit the MySQL official website and download MySQL Community Server.
    - Choose the version suitable for your operating system (Windows, macOS, Linux).
2. Installation Process
    - Run the installer and follow the setup instructions.
    - During installation, you'll be asked to configure:
        - Root Password: Set a strong password (you'll need it to connect later).
        - Port Number: Default is 3306 (keep it unchanged unless required).
        - Database Setup: You can create a sample database like chinook for testing.
3. Verify Installation
    - Open the terminal (Command Prompt or Terminal) and run:
        - mysql -u root -p

- Enter the root password you set during installation. If successful, you'll enter the MySQL shell.

Installing Python

Python is required to run Lang Chain and interact with MySQL.

1. Download Python
    - Go to the Python official website and download the latest version.
2. Install Python
    - Follow the installation steps and check "Add Python to PATH" during setup.
3. Verify Installation
    - Open the terminal and run: python --version
    - This should display the installed Python version.
4. Ensure pip is Installed
    - Pip is Python’s package manager. Ensure it’s installed by running.

Installing Lang Chain and Required Packages

Lang Chain helps in AI-powered SQL generation. You'll also need connectors to interact with MySQL.

1. Installing the Required Packages

   Run the following commands:

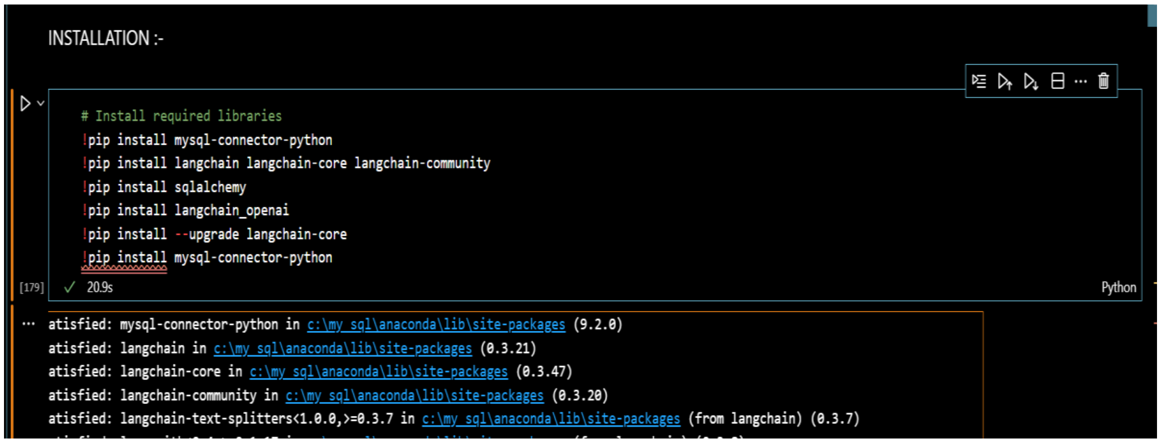
   ```
   pip install mysql-connector-python
   pip install langchain langchain-core langchain-community
   pip install sqlalchemy
   ```
2. Verify Installation

- After installation, you can check if Lang Chain is installed correctly by running:

  import langchain_core

  print(langchain_core.__version__)

- If it prints the version number, the installation is successful!



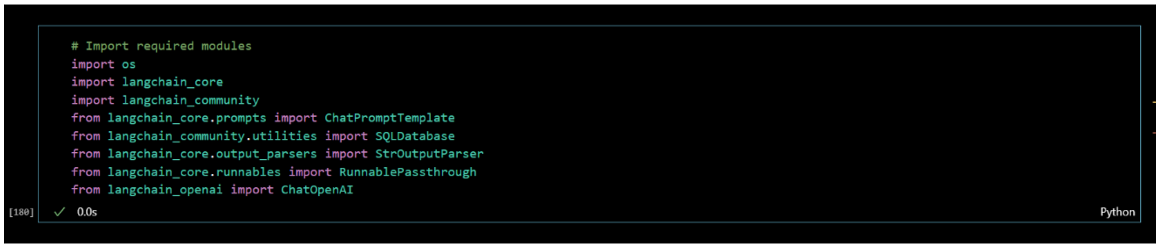
INSTALLATION :-

```
# Install required libraries
!pip install mysql-connector-python
!pip install langchain langchain-core langchain-community
!pip install sqlalchemy
!pip install langchain_openai
!pip install --upgrade langchain-core
!pip install mysql-connector-python
```

[179] ✓ 20.9s Python

```
atisfied: mysql-connector-python in c:\my sql\anaconda\lib\site-packages (9.2.0)
atisfied: langchain in c:\my sql\anaconda\lib\site-packages (0.3.21)
atisfied: langchain-core in c:\my sql\anaconda\lib\site-packages (0.3.47)
atisfied: langchain-community in c:\my sql\anaconda\lib\site-packages (0.3.20)
atisfied: langchain-text-splitters<1.0.0,>=0.3.7 in c:\my sql\anaconda\lib\site-packages (from langchain) (0.3.7)
```

Fig 6.1.1.a Installing Required Packages



```
# Import required modules
import os
import langchain_core
import langchain_community
from langchain_core.prompts import ChatPromptTemplate
from langchain_community.utilities import SQLDatabase
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
from langchain_openai import ChatOpenAI
```

[180] ✓ 0.0s Python

Fig 6.1.1.b Importing Required Packages

6.1.2, SQL CHAIN SETUP MODULE

Step 1 - Configuring the MySQL Connection

The first step in setting up an SQL Chain is establishing a connection between MySQL and Lang Chain. This ensures that queries can be executed dynamically from the AI model to the database. The connection is made using a **connection string (URI)** that contains essential credentials, including the MySQL username, password, host, and database name. The URI follows a structured format:

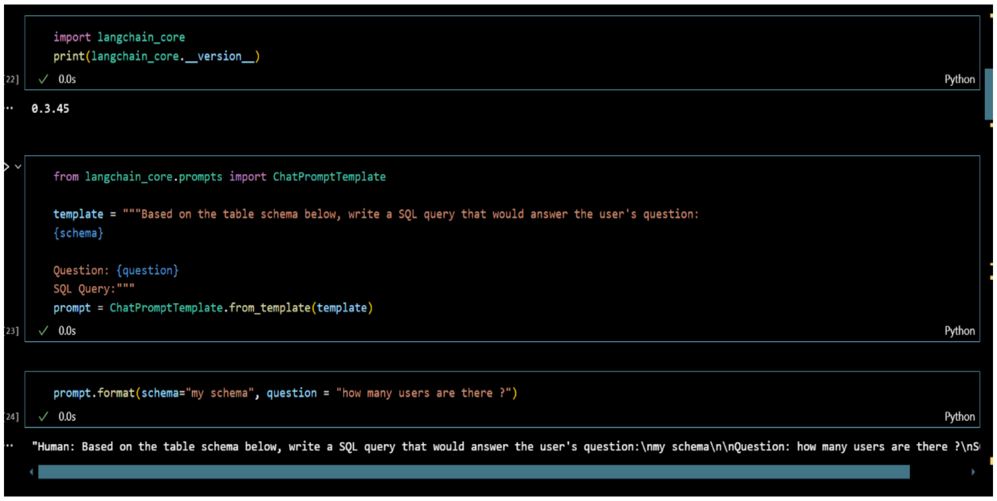
➔ mysql+mysqlconnector://<username>:<password>@<host>/<database>

Here, <username> represents the MySQL username (e.g., root), <password> is the MySQL password (which must be URL-encoded if it contains special characters), <host> is the server address (for local setups, this is localhost), and <database> is the specific database being used (e.g., chinook). Lang Chain uses SQLDatabase.from_uri() to establish the connection. This approach provides a **secure and structured method** for interacting with the database without manually logging in every time. Proper configuration ensures smooth communication between the AI model and MySQL.

Step 2 - Verifying the Database Connection

Once the connection is established, it is necessary to verify that Lang Chain can successfully communicate with MySQL. A key part of this verification process is checking whether the database connection is valid, ensuring that authentication details, such as username and password, are correct. If authentication fails, error messages will indicate issues such as incorrect credentials, firewall restrictions, or MySQL not running.

After successful authentication, retrieving the list of usable tables confirms that Lang Chain can access the database schema. This step is crucial because it allows the AI model to understand the database structure before generating queries. Additionally, handling potential errors ensures that any misconfigurations are detected early, preventing unexpected failures when executing queries. A verified database connection guarantees consistent and error-free interactions between Lang Chain and MySQL.



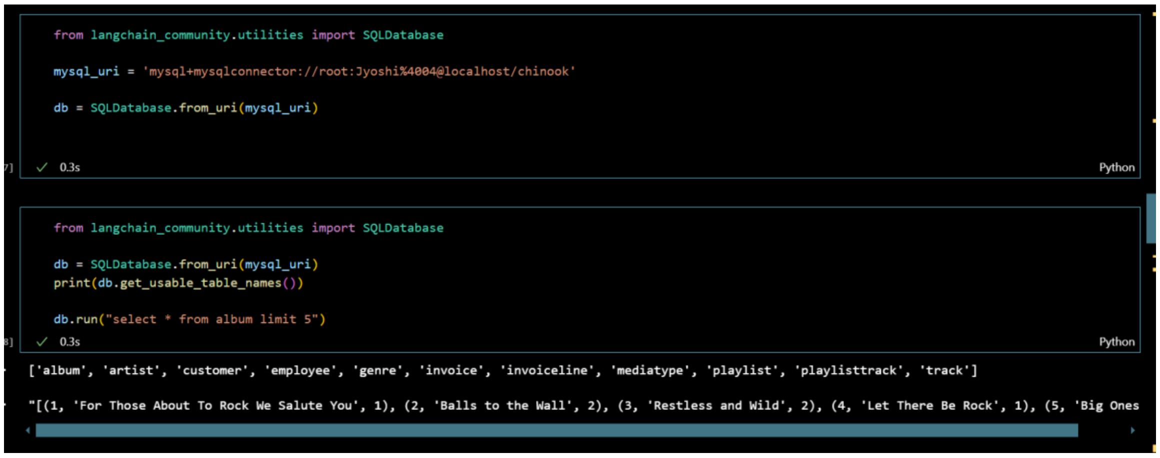
```
import langchain_core
print(langchain_core.__version__)
```
```
0.3.45
```
```
from langchain_core.prompts import ChatPromptTemplate

template = """Based on the table schema below, write a SQL query that would answer the user's question:
{schema}

Question: {question}
SQL Query:"""
prompt = ChatPromptTemplate.from_template(template)
```
```
prompt.format(schema="my schema", question = "how many users are there ?")
```
```
"Human: Based on the table schema below, write a SQL query that would answer the user's question:\nmy schema\n\nQuestion: how many users are there ?\nS
```

Fig 6.1.2.a, MySQL Database connection

Step 3 - Running an SQL Query in the Chain

After verifying the database connection, the next step is testing whether SQL queries can be executed within the SQL Chain. This process begins with running a simple test query, such as retrieving the first few rows from a table. A common example is querying the album table:

SELECT * FROM album LIMIT 5;



```
from langchain_community.utilities import SQLDatabase

mysql_uri = 'mysql+mysqlconnector://root:Jyoshi%4004@localhost/chinook'

db = SQLDatabase.from_uri(mysql_uri)
```

✓ 0.3s Python

```
from langchain_community.utilities import SQLDatabase

db = SQLDatabase.from_uri(mysql_uri)
print(db.get_usable_table_names())

db.run("select * from album limit 5")
```

✓ 0.3s Python

```
['album', 'artist', 'customer', 'employee', 'genre', 'invoice', 'invoiceline', 'mediatype', 'playlist', 'playlisttrack', 'track']

"[(1, 'For Those About To Rock We Salute You', 1), (2, 'Balls to the Wall', 2), (3, 'Restless and Wild', 2), (4, 'Let There Be Rock', 1), (5, 'Big Ones
```

Fig 6.1.2.b Running SQL query within the langchain

Executing this query within the Lang Chain environment confirms that data retrieval is functioning correctly. The output is returned in a structured format, either as a JSON object or a tabular dataset. If the query executes successfully, it means Lang Chain has full access to the database. If errors occur, they often indicate issues such as incorrect table names, missing database permissions, or SQL syntax errors. Handling these errors ensures that the AI-generated queries will be executed efficiently without runtime failures. This step is a crucial checkpoint in confirming that the SQL Chain is ready for integration into AI-driven workflows.

Step 4 - Giving a Question and Generating an SQL Query

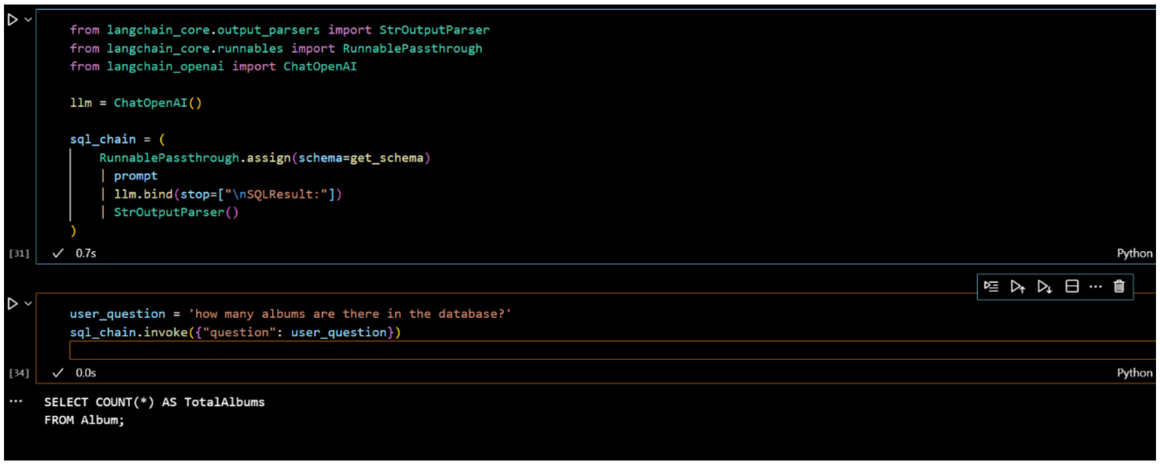
With a functional SQL Chain, the next step is integrating natural language query processing. Instead of writing SQL queries manually, users can ask questions in plain English, and Lang Chain will generate the corresponding SQL query. This is achieved using a prompt template that guides the AI in structuring SQL statements based on the database schema and user input.

For instance, if a user asks:

How many albums are in the database?

Lang Chain translates this question into a valid SQL query:

SELECT COUNT(*) FROM album;



```
from langchain_core.output_parsers import StrOutputParser
from langchain_core.runnables import RunnablePassthrough
from langchain_openai import ChatOpenAI

llm = ChatOpenAI()

sql_chain = (
    RunnablePassthrough.assign(schema=get_schema)
    | prompt
    | llm.bind(stop=["\nSQLResult:"])
    | StrOutputParser()
)
```

[31] ✓ 0.7s Python

```
user_question = 'how many albums are there in the database?'
sql_chain.invoke({"question": user_question})
```

[34] ✓ 0.0s Python

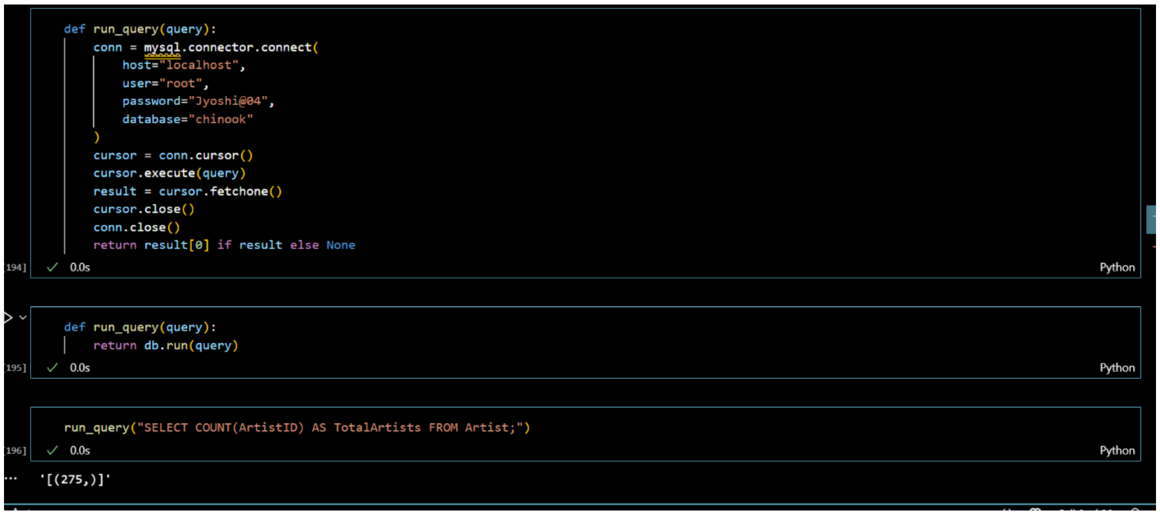
```
SELECT COUNT(*) AS TotalAlbums
FROM Album;
```

Fig 6.1.2.c Query Generation

Once generated, the SQL query is executed on the MySQL database, and the result is returned in a human-readable format. This eliminates the need for users to have prior SQL knowledge, as they can retrieve database insights through conversational queries. The ability to dynamically generate SQL queries makes the SQL Chain a powerful tool for data retrieval, analysis, and business intelligence applications.

Step 5 – Run the Generated SQL Query

The provided code defines a function run query(query) to execute SQL queries on a MySQL database named "chinook." The first implementation of run_query() establishes a connection using mysql.connector.connect(), specifying localhost as the host, "root" as the user, and "joshi@@04" as the password. The function creates a cursor, executes the query, fetches one result using fetchone(), and then closes the cursor and connection. The return statement ensures that if a result exists, the first value is returned; otherwise, it returns None. This method allows direct interaction with the database while ensuring proper connection handling.



```
[194]
def run_query(query):
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="Jyoshi@04",
        database="chinook"
    )
    cursor = conn.cursor()
    cursor.execute(query)
    result = cursor.fetchone()
    cursor.close()
    conn.close()
    return result[0] if result else None
✓ 0.0s                                                        Python

[195]
def run_query(query):
    return db.run(query)
✓ 0.0s                                                        Python

[196]
run_query("SELECT COUNT(ArtistID) AS TotalArtists FROM Artist;")
✓ 0.0s                                                        Python
... '[(275,)]'
```

Fig 6.1.2.d Running the generated SQL Query

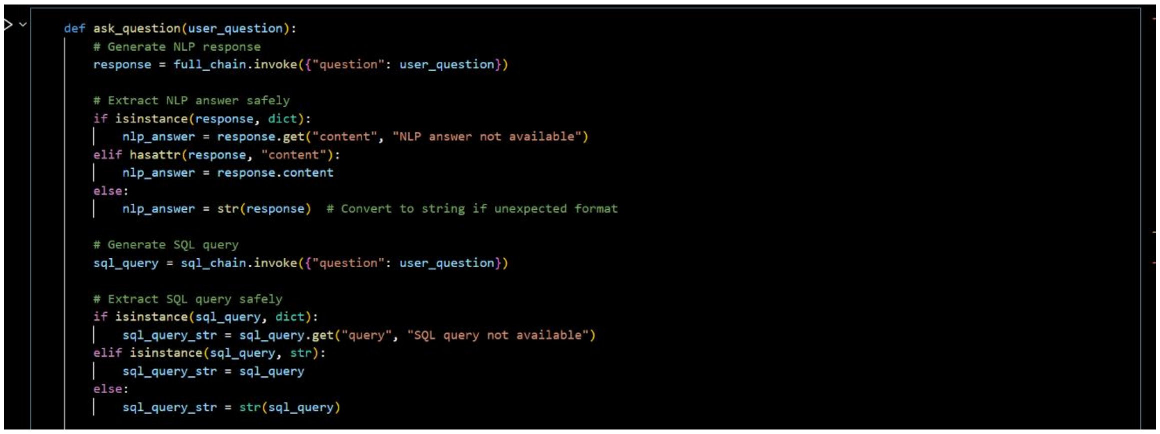
The second implementation of run_query(query) is a simplified version that directly calls db.run(query), assuming db is an established database object with a run method. This abstraction removes the need to manually handle database connections, making the function more modular and reusable. The final execution runs a query to count the total number of artists in the Artist table, returning 275, formatted as a tuple inside a list. This suggests that fetchone() might be returning data in a wrapped format, which could be adjusted based on the desired output format.

6.1.3, LANG CHAIN SETUP MODULE

The provided code establishes a LangChain-powered pipeline to convert natural language questions into SQL queries, execute them, and return human-readable answers. The sql_chain component plays a crucial role by structuring a sequence of operations that process the input question.

Initially, RunnablePassthrough.assign(schema=lambda _: get_schema()) retrieves the database schema, which helps the language model understand the database structure.

The question is then passed through a prompt template and processed by an LLM, which is configured to stop at "SQLResult:", ensuring that only the SQL query is extracted. Finally, StrOutputParser() ensures a clean string output, and RunnableLambda(extract_sql_query) extracts the SQL query from the generated response.



```python
def ask_question(user_question):
    # Generate NLP response
    response = full_chain.invoke({"question": user_question})

    # Extract NLP answer safely
    if isinstance(response, dict):
        nlp_answer = response.get("content", "NLP answer not available")
    elif hasattr(response, "content"):
        nlp_answer = response.content
    else:
        nlp_answer = str(response)  # Convert to string if unexpected format

    # Generate SQL query
    sql_query = sql_chain.invoke({"question": user_question})

    # Extract SQL query safely
    if isinstance(sql_query, dict):
        sql_query_str = sql_query.get("query", "SQL query not available")
    elif isinstance(sql_query, str):
        sql_query_str = sql_query
    else:
        sql_query_str = str(sql_query)
```

Fig 6.1.3.a From Response to NLP Answer

To execute the SQL query, the run_query(query) function is used, calling db.run(query) to fetch results from the connected database. The full response generation chain (full_chain) operates in several stages:

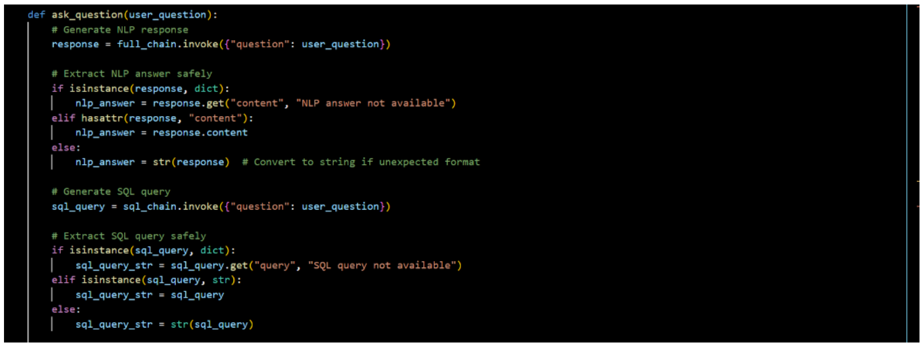
(1) generating the SQL query from the user's natural language question,

(2) fetching the schema to provide necessary database structure,

(3) executing the generated SQL query using run_query(vars["query"]), and

(4) formatting the output using prompt_response,

which presents the schema, question, SQL query, and its output in a structured response. The final step utilizes an LLM to convert the SQL query’s response into natural language. The effectiveness of this pipeline is tested using an example question, "How many albums are there in the database?", which follows the outlined process to provide a well-formatted answer.



```python
def ask_question(user_question):
    # Generate NLP response
    response = full_chain.invoke({"question": user_question})

    # Extract NLP answer safely
    if isinstance(response, dict):
        nlp_answer = response.get("content", "NLP answer not available")
    elif hasattr(response, "content"):
        nlp_answer = response.content
    else:
        nlp_answer = str(response)  # Convert to string if unexpected format

    # Generate SQL query
    sql_query = sql_chain.invoke({"question": user_question})

    # Extract SQL query safely
    if isinstance(sql_query, dict):
        sql_query_str = sql_query.get("query", "SQL query not available")
    elif isinstance(sql_query, str):
        sql_query_str = sql_query
    else:
        sql_query_str = str(sql_query)
```

Fig 6.1.3.b Natural language Response Generation

6.2, MODELS

6.2.1, LangChain Model

Lang Chain is one of the core models used in this project. It’s not a language model itself, but a framework that helps connect large language models (LLMs) with other tools like databases, APIs, and memory systems. In this project, Lang Chain acts as the backbone that allows the user’s natural language question to be processed and transformed into a meaningful SQL query that can interact with a database. Its chain-based structure means it can handle steps like interpreting the query, managing conversation flow, and connecting with the backend database — all in a smart, manageable way.

The real strength of Lang Chain is its modularity. That means you can break the process into parts like: user input → language understanding → SQL generation → database query → output formatting. Each of these parts can be controlled and adjusted separately. This makes the system flexible and customizable. For example, you can connect different LLMs to the system, change how results are formatted, or even allow follow-up questions from the user without starting from scratch — all thanks to Lang Chain.

In simple terms, Lang Chain is like a smart middleman between the user and the machine. It listens to what the user is saying, understands what is being asked, and coordinates all the actions needed to get the correct data and respond clearly. Without Lang Chain, the system wouldn’t know how to combine the natural language abilities of a model with the logic of querying a database — so it plays a very crucial role in making the project work smoothly.

6.2.2 NLP (Natural Language) Model

NLP is what allows the system to understand and interpret the user's question written in plain English. It involves techniques that analyse human language to figure out the intent behind it. For example, when a user types "Give me the number of employees in each department," NLP helps break that sentence into parts — it recognizes action words like "give," data terms like "employees," and logical concepts like "each department." This understanding is the first step in converting a natural sentence into a structured SQL query.

Behind the scenes, this is often powered by pre-trained language models like GPT (OpenAI), BERT, or similar models. These models have been trained on massive amounts of text data, so they can understand a wide variety of questions, sentence structures, and even common errors in language. In your project, such a model (connected through Lang Chain) helps the system correctly understand the user's intent, even if the question is not perfectly worded.

NLP plays a huge role in making the system feel natural and human-friendly. It eliminates the need for users to use perfect grammar, spelling, or keywords. It allows follow-up questions and even context-based understanding (e.g., the system remembering the last question asked). This makes data interaction feel more like a conversation and less like programming. In short, NLP is what gives your system the "brains" to talk to people in a natural and intelligent way.

6.3, Data Flow Diagram

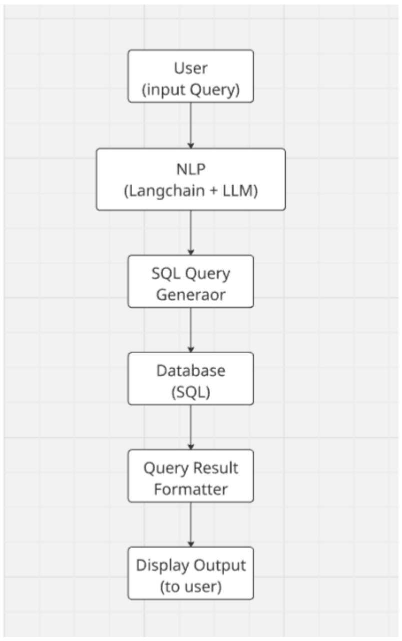


Fig 6.3 Data Flow Diagram

1. User Input – The user types a natural language question like "Show me total sales in January."
2. Natural Language Processor – The input is processed by Lang Chain and an NLP model to understand the user's intent and extract relevant keywords.
3. SQL Query Generator – The system then converts the interpreted intent into an actual SQL query (e.g., SELECT SUM(sales) FROM table WHERE month='January';)
4. Database Interaction – The query is executed on the backend SQL database to retrieve the requested information.
5. Result Formatting – The raw query results are formatted into a human-friendly response, possibly with charts or text explanation.
6. Output Display – The final result is shown to the user in a readable and understandable format.

6.4, UML Diagrams

6.4.1 Use Case Diagram

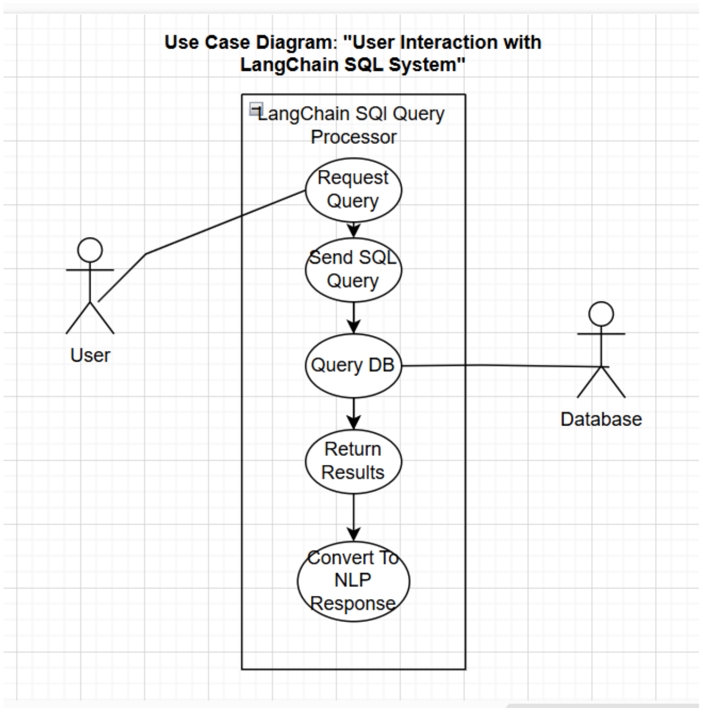

Use Case Diagram: "User Interaction with LangChain SQL System"
LangChain SQl Query Processor
Request Query
Send SQL Query
Query DB
Return Results
Convert To NLP Response
User
Database

Fig 6.4.1 Use Case Diagram

6.4.2 Class Diagram

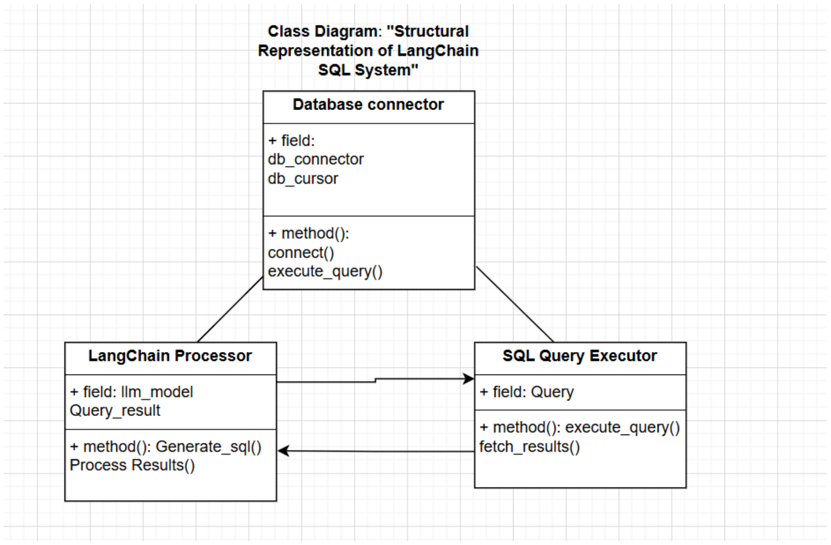


Fig 6.4.2 Class Diagram

6.4.3 Sequence Diagram

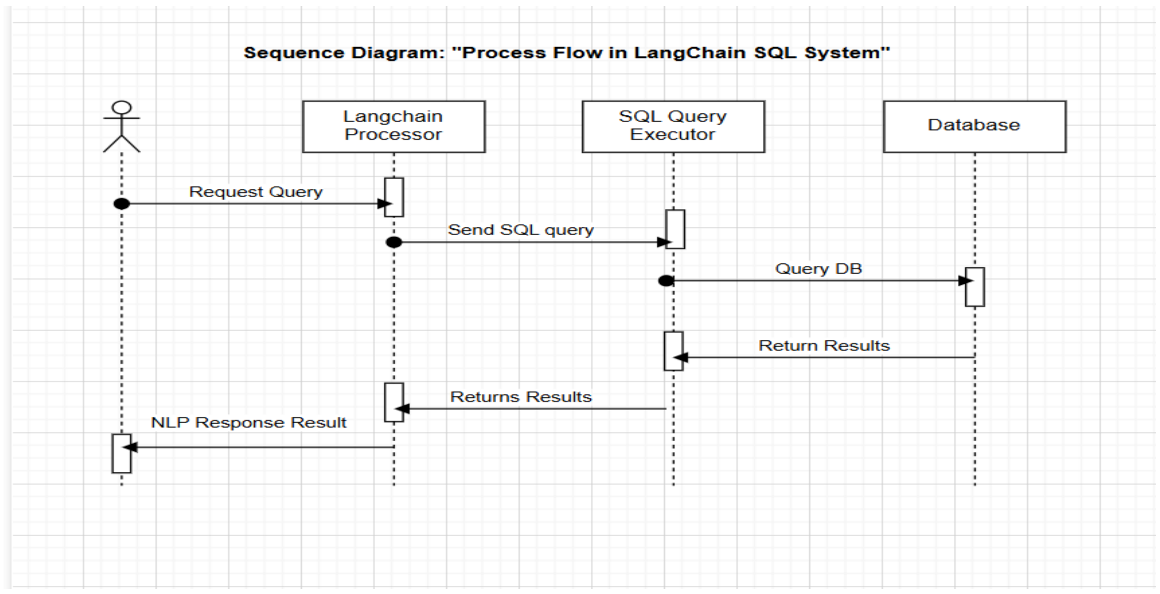


Fig 6.4.3 Sequence Diagram

6.5, Input Design

The input design of this project is focused on simplicity and user-friendliness. Users interact with the system by typing natural language queries, just like they would while chatting with a person. For example, instead of writing complex SQL queries, a user can simply type something like *"What are the total sales for March?"* or *"List all employees in the marketing department."* This makes the system accessible to users who have no technical knowledge of databases or SQL. The input field can be part of a web-based interface or a simple chat box where the user enters their question.

Behind the scenes, the system captures this input and sends it to the NLP engine, which works with Lang Chain to understand the meaning and convert it into a structured format. Input validation is also included to make sure that the query is meaningful and not empty. If the input is unclear, the system can prompt the user to rephrase their question, making the interaction more natural and effective. The design ensures that users don't need to learn any new commands or technical terms – plain English is enough.

This user-centric input approach ensures better usability, wider accessibility, and less friction while interacting with complex data. It encourages even non-technical users to confidently explore and retrieve information from structured databases, which is a major improvement over traditional input systems that rely on manual query writing.

6.6, Output Design

The output design focuses on clarity, simplicity, and usefulness. Once the system processes the user's query and fetches data from the database, it presents the results in a clean, understandable format. Depending on the query type, the output can be displayed as text, a table, or even a simple visual like a bar graph or pie chart. For example, a question like "Show department-wise salary distribution" could result in a neat chart, while "List all products under $100" would return a well-formatted table of items.

The key goal of the output design is to make sure that users can easily understand the results without needing to interpret raw data. The system avoids technical database terms in the output and instead uses human-friendly language. It may also offer follow-up suggestions or questions to improve the conversation, like "Would you like to filter by region?" or "Do you want this data as a download?" which improves interactivity and user engagement.

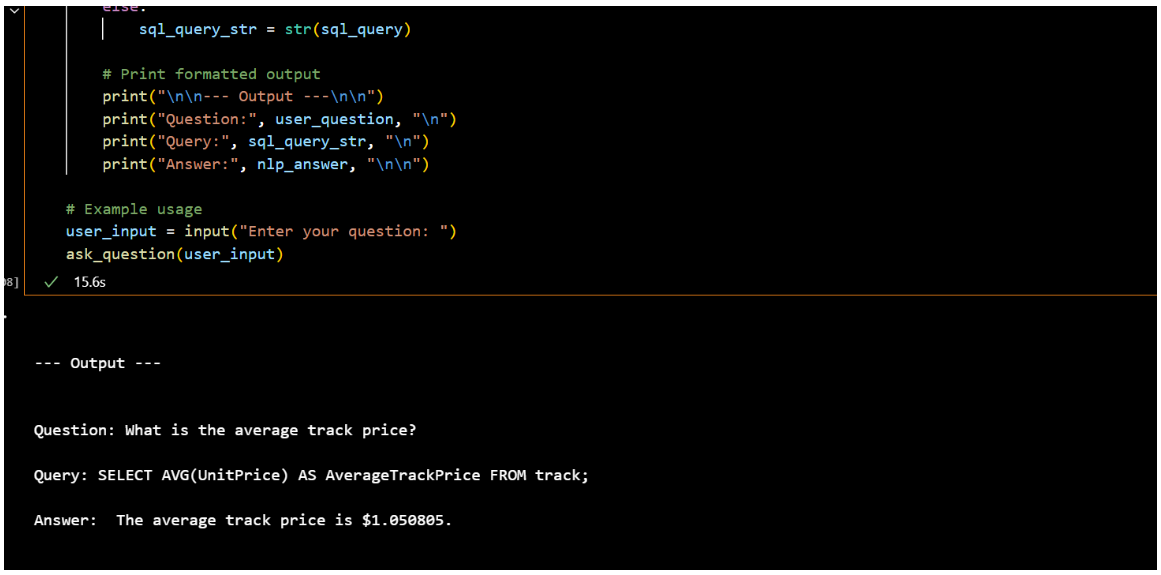
In summary, the output is designed to not just give answers, but to help users make sense of the data quickly. This bridges the gap between databases and decision-making by giving people the power to get insights from data in a conversational, visual, and intuitive way.

CHAPTER 7

SYSTEM TESTING

7.1 Definition

System testing checks the complete system to ensure it works according to the defined requirements. It validates the overall flow — from taking natural language input, converting it to SQL using Lang Chain and NLP, executing it on the database, and showing the correct output. It helps catch bugs, ensure smooth performance, and confirm that the system is user-ready.



```
        else:
            sql_query_str = str(sql_query)

        # Print formatted output
        print("\n\n--- Output ---\n\n")
        print("Question:", user_question, "\n")
        print("Query:", sql_query_str, "\n")
        print("Answer:", nlp_answer, "\n\n")

    # Example usage
    user_input = input("Enter your question: ")
    ask_question(user_input)
 ✓ 15.6s


--- Output ---


Question: What is the average track price?

Query: SELECT AVG(UnitPrice) AS AverageTrackPrice FROM track;

Answer:  The average track price is $1.050805.
```

Fig 7a Testing

System testing checks the complete system to ensure it works according to the defined requirements. It validates the overall flow — from taking natural language input, converting it to SQL using Lang Chain and NLP, executing it on the database, and showing the correct output. It helps catch bugs, ensure smooth performance, and confirm that the system is user-ready.

The output of this Lang Chain-powered system consists of three main components: the user's question, the generated SQL query, and the NLP-processed answer. When a user enters a question (e.g., *"How many albums are there in the database?"*), the system first constructs an SQL query that accurately retrieves the required information (e.g., SELECT COUNT(*) FROM Album;). This query is then executed on the connected database, fetching the raw result. Finally, the system processes the SQL response into a human-friendly, natural language answer (e.g., *"There are 275 albums in the database."*). The structured output format ensures clarity, making it easy to interpret both the query and its corresponding answer.

System testing is a very important phase in the development of this project. It helps make sure that the entire system works as expected — from the moment a user types in a question to when they receive the correct and meaningful output. In this project, we test whether the natural language queries are properly understood, converted into correct SQL queries, and return accurate results from the database. This phase also checks whether the system can handle both simple and complex queries without crashing or giving incorrect results.

During testing, we simulate various real-life scenarios that a user might encounter. For example, we test with queries like *“List all employees from the IT department”* or *“Show me the sales in March”* to see if the system correctly converts them into SQL and retrieves the right data. We also try incorrect or incomplete queries such as *“Employees sales March”* to check if the system can guide the user to rephrase or clarify their question. This helps ensure the system is not just functional, but also user-friendly and fault-tolerant.

Another important part of system testing is checking how all components work together — the front-end interface, Lang Chain engine, NLP model, and the SQL database. We check performance, accuracy, responsiveness, and error handling. If all the pieces interact smoothly and the final output is correct and understandable, we can say the system is ready for use. This testing gives us confidence that the application will work properly when used by real users in a real environment.

7.2 Levels of Testing

Unit Testing

The objective of unit testing is to ensure that each individual function or component of the project performs as expected when tested separately. The method involves isolating every module—like the NLP parser, SQL query generator, or response handler—and checking their logic using sample inputs and expected outputs. For this, we used tools like Python’s built-in unit test framework and pytest, which helped us automate and validate the behavior of these individual pieces.

Integration Testing

The objective here is to confirm that different modules of the system work properly when combined. In our project, this meant testing how the language processing system integrates with the database layer via LangChain and whether the queries are generated and executed correctly. We followed the method of gradually combining modules (top-down or bottom-up approach), and tools like Postman were used to test API requests, while Selenium could be used for interface-based flows if needed.

System Testing

System testing aims to validate the entire system as a whole and ensure it meets the functional and non-functional requirements. This involved end-to-end testing of the natural language input through to data retrieval and output presentation. The methods used included load testing, stress testing, and functionality verification. These tests were manually verified and partially automated to simulate user scenarios.

Performance Testing

The main objective of performance testing is to make sure the system behaves well under expected and peak load conditions. This included checking response times when multiple queries are fired, or when large datasets are accessed. The method included simulating concurrent users and high data volume. We used tools like Apache JMeter for load testing and basic Python scripts to mimic query floods.

User Acceptance Testing (UAT)

The objective of UAT is to verify that the system meets the needs and expectations of the end users. We involved potential users—like students and faculty—to interact with the system and provide feedback. The method was to observe their experience, gather feedback through forms or discussions, and refine the interface or flow based on real-world usability. No specific tools were required here; testing was more observational and feedback-driven.

Security Testing

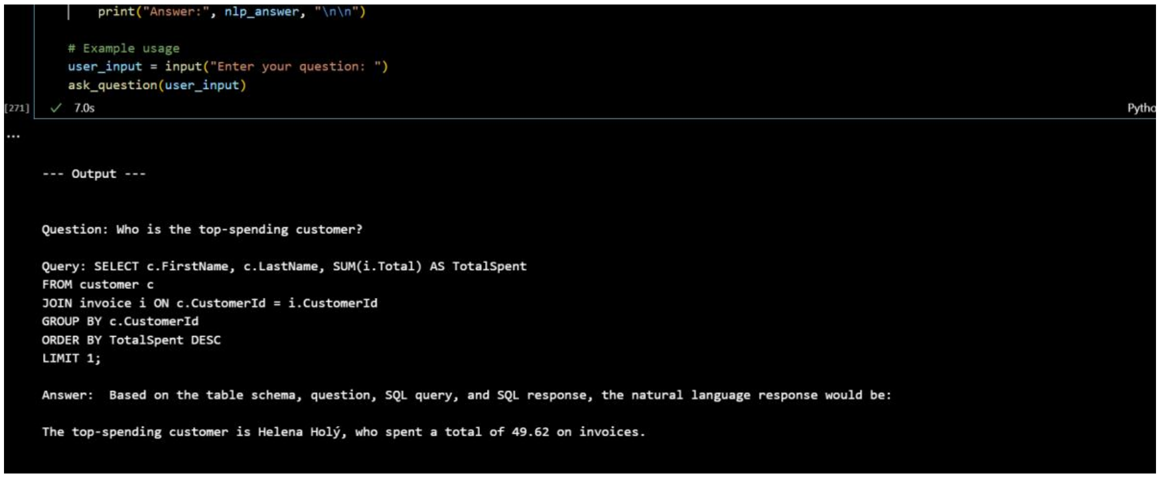
The purpose of security testing is to identify and address vulnerabilities that could expose data or allow unauthorized access. In our case, we tested the system for SQL injection threats and ensured the OpenAI API was used securely. The method involved simulating malicious inputs and scanning for weak points, using tools like OWASP ZAP and Burp Suite for identifying security flaws.

Regression Testing

Regression testing ensures that new changes or updates in the codebase do not negatively affect existing functionalities. Whenever we updated a feature, we re-ran earlier test cases to make sure everything still worked fine. We used automated test scripts that were run regularly using GitHub Actions or Jenkins in a CI/CD environment to catch any unintended side effects early.

Deployment Testing

The objective of deployment testing is to verify that the system performs correctly in the real-world environment where it will be used. We deployed the application in a staging environment first to identify any setup or compatibility issues. Methods included simulating live use, checking database connectivity, and validating overall behaviour before moving to production.



```
    |   print("Answer:", nlp_answer, "\n\n")

    # Example usage
    user_input = input("Enter your question: ")
    ask_question(user_input)
[271]  ✓ 7.0s                                                    Pytho
...

--- Output ---


Question: Who is the top-spending customer?

Query: SELECT c.FirstName, c.LastName, SUM(i.Total) AS TotalSpent
FROM customer c
JOIN invoice i ON c.CustomerId = i.CustomerId
GROUP BY c.CustomerId
ORDER BY TotalSpent DESC
LIMIT 1;

Answer:  Based on the table schema, question, SQL query, and SQL response, the natural language response would be:

The top-spending customer is Helena Holý, who spent a total of 49.62 on invoices.
```

Fig 7b – Testing

7.3 Test Case Design Techniques

Equivalence Partitioning

The objective of this technique is to minimize the number of test cases while still maintaining good coverage. It does this by dividing all possible inputs into groups or "partitions" where the system should behave similarly. For instance, if a user can input a number between 1 and 100, we might test one valid input from that range and one invalid input outside of it. This helps catch input-related errors without testing every single possible value.

Boundary Value Analysis

The main goal here is to test the system's behaviour at the edge of valid input ranges. It's commonly known that errors often occur at the boundaries rather than in the middle of the input range. So, we test the minimum and maximum valid inputs, and just below and just above those limits. This helps identify how the system handles extreme cases, like the smallest and largest values accepted.

Error Guessing

This technique is based on intuition and experience. The idea is to guess where the system might fail—usually places where issues have occurred before or where complexity exists. For example, entering special characters in a text field or rapidly clicking a button multiple time. The objective is to proactively test for common or likely user mistakes to make the system more robust.

Decision Table Testing

The objective of this method is to ensure that the system behaves correctly for all possible combinations of input conditions. It is especially useful when the system has complex business rules or logical conditions. Each condition and its outcome are written in a table format, and test cases are derived for every combination. This helps avoid logical gaps and ensures comprehensive decision coverage.

Use Case Testing

This technique focuses on testing real-world user scenarios from start to finish. The objective is to validate that users can complete typical tasks with the system as intended. For example, in our project, a use case might be: a user asks a question in natural language, the system processes it, fetches the data, and displays the results. Testing the entire flow ensures the system is functional and user-friendly from a practical perspective.

CHAPTER 8

Conclusion and Future Work

8.1 Conclusion

The implementation of Lang Chain for SQL query generation and response automation provides a powerful framework for natural language interaction with databases. By utilizing Lang Chain's modular design, the system effectively translates user queries into SQL commands, retrieves the relevant data, and returns a meaningful response. This eliminates the need for users to have SQL expertise, making data retrieval more accessible. Additionally, the use of Runnable Lambda, structured prompt templates, and function chaining ensures a streamlined process that is both scalable and efficient.

Moreover, the system's ability to dynamically extract and execute SQL queries ensures flexibility across different use cases. Whether it's answering simple data retrieval questions or handling complex queries, Lang Chain's structured approach enhances accuracy and usability. By integrating error-handling mechanisms and ensuring safe query execution, this setup minimizes failures and improves reliability, making it a robust AI-driven database interaction tool.

8.2 Future Work

To further enhance this system, several improvements can be considered:

1. Query Optimization – Implement AI-driven query optimization techniques to improve the performance and efficiency of SQL execution.

2. Multi-Database Support – Extend compatibility to multiple databases beyond MySQL (e.g., PostgreSQL, MongoDB) for greater versatility.

3. Enhanced NLP Understanding – Fine-tune the language model to handle complex and ambiguous queries, making interactions more intuitive.

4. Interactive UI Development – Create a web-based or chatbot interface for user queries to improve usability and accessibility.

REFERENCES

Data Source: https://github.com/lerocha/chinook-database

[1] Chowdhery A, Narang S, Devlin J, Bosma M, Mishra G, et al., "PaLM: Scaling language modeling with pathways," *arXiv preprint*, arXiv:2204.02311, 2022.

[2] Dasgupta S, Ray S, Talukdar P, "LangChain: A framework for building applications with LLM APIs," *arXiv preprint*, arXiv:2211.03786, 2022.

[3] Kojima T, Gu SS, Reid M, Gribovic Y, Neubig G, "Large language models are zero-shot learners," *arXiv preprint*, arXiv:2207.04344, 2022.

[4] Scholak T, Schubotz M, Crain S, "Language models can explain their behavior on language tasks," *arXiv preprint*, arXiv:2212.09783, 2022.

[5] Wei J, Tay Y, Raffel C, Zoph B, Abbeel P, et al., "Chain of thought prompting elicits reasoning in large language models," *arXiv preprint*, arXiv:2201.11903, 2022.

[6] Guo J, Zhan Z, Gao Y, Xiao Y, Lou JG, et al., "Towards complex text-to-SQL in cross-domain database with intermediate representation," *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2023.

[7] Hwang W, Yim J, Park S, Seo M, "WikiSQL with table-aware word contextualization," *arXiv preprint*, 2023.

[8] Wang C, Liang P, Manning CD, "Rat-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers," *Proceedings of the Association for Computational Linguistics*, 2023.

[9] Zhong V, Xiong C, Socher R, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," *arXiv preprint*, 2023.

[10] Arpan Shaileshbhai Korat, "AI-Augmented LangChain: Facilitating Natural Language SQL Queries for Non-Technical Users," *Journal of Artificial Intelligence & Cloud Computing*, 2024.

APENDIX – Source Code

```
# === Conversational Database Assistant ===
# File: conversational_db_assistant.py


import os
from langchain.chat_models import ChatOpenAI
from langchain.chains import SQLDatabaseChain
from langchain.sql_database import SQLDatabase
import mysql.connector


# Step 1: Set your OpenAI API Key
# Either use environment variables or replace the key directly below
os.environ["OPENAI_API_KEY"] = "your_openai_api_key_here"


# Step 2: Connect to your MySQL database
def get_database_connection():
    # Replace these credentials with your actual DB info
    db_uri =
"mysql+mysqlconnector://root:yourpassword@localhost/your_database_name"
    return SQLDatabase.from_uri(db_uri)


# Step 3: Initialize the LangChain with SQL database
def initialize_langchain_chain():
    llm = ChatOpenAI(temperature=0, model="gpt-3.5-turbo")
    db = get_database_connection()
    db_chain = SQLDatabaseChain.from_llm(llm=llm, db=db, verbose=True)
    return db_chain

# Step 4: Run the chatbot interaction
def start_conversational_interface():
    print("Welcome to Conversational Database Assistant!")
    print("You can ask anything like: 'List all customers from Chennai'")
    print("Type 'exit' to stop.\n")

    db_chain = initialize_langchain_chain()

    while True:
        user_query = input(" You: ")
        if user_query.lower() in ["exit", "quit"]:
            print(" Exiting... Have a nice day!")
            break
        try:
            response = db_chain.run(user_query)
            print("\nAssistant:\n", response)
        except Exception as e:
            print("Error occurred:", str(e))

# Main Execution
if __name__ == "__main__":
    start_conversational_interface()
```

Evaluation Rubrics for Project work:

Rubric (CO)	Excellent (Wt = 3)	Good (Wt = 2)	Fair (Wt = 1)
Selection of Topic (CO1)	Select a latest topic through complete knowledge of facts and concepts.	Select a topic through partial knowledge of facts and concepts.	Select a topic through improper knowledge of facts and concepts.
Analysis and Synthesis (CO2)	Thorough comprehension through analysis/ synthesis.	Reasonable comprehension through analysis/ synthesis.	Improper comprehension through analysis/ synthesis.
Problem Solving (CO3)	Thorough comprehension about what is proposed in the literature papers.	Reasonable comprehension about what is proposed in the literature papers.	Improper comprehension about what is proposed in the literature.
Literature Survey (CO4)	Extensive literature survey with standard references.	Considerable literature survey with standard references.	Incomplete literature survey with substandard references.
Usage of Techniques & Tools (CO5)	Clearly identified and has complete knowledge of techniques & tools used in the project work.	Identified and has sufficient knowledge of techniques & tools used in the project work.	Identified and has inadequate knowledge of techniques & tools used in project work.
Project work impact on Society (CO6)	Conclusion of project work has strong impact on society.	Conclusion of project work has considerable impact on society.	Conclusion of project work has feeble impact on society.
Project work impact on Environment (CO7)	Conclusion of project work has strong impact on Environment.	Conclusion of project work has considerable impact on environment.	Conclusion of project work has feeble impact on environment.
Ethical attitude (CO8)	Clearly understands ethical and social practices.	Moderate understanding of ethical and social practices.	Insufficient understanding of ethical and social practices.
Independent Learning (CO9)	Did literature survey and selected topic with a little guidance	Did literature survey and selected topic with considerable guidance	Selected a topic as suggested by the supervisor
Oral Presentation (CO10)	Presentation in logical sequence with key points, clear conclusion and excellent language	Presentation with key points, conclusion and good language	Presentation with insufficient key points and improper conclusion
Report Writing (CO10)	Status report with clear and logical sequence of chapters using excellent language	Status report with logical sequence of chapters using understandable language	Status report not properly organized
Time and Cost Analysis (CO11)	Comprehensive time and cost analysis	Moderate time and cost analysis	Reasonable time and cost analysis
Continuous learning (CO12)	Highly enthusiastic towards continuous learning	Interested in continuous learning	Inadequate interest in continuous learning

ANNEXURE

Title of the Project : Conversational Database Interaction Using LangChain and NLP

Name of the students: C. JYOSHITHA 21751A3211

Name of the Guide & Designation: Mrs. Ramya

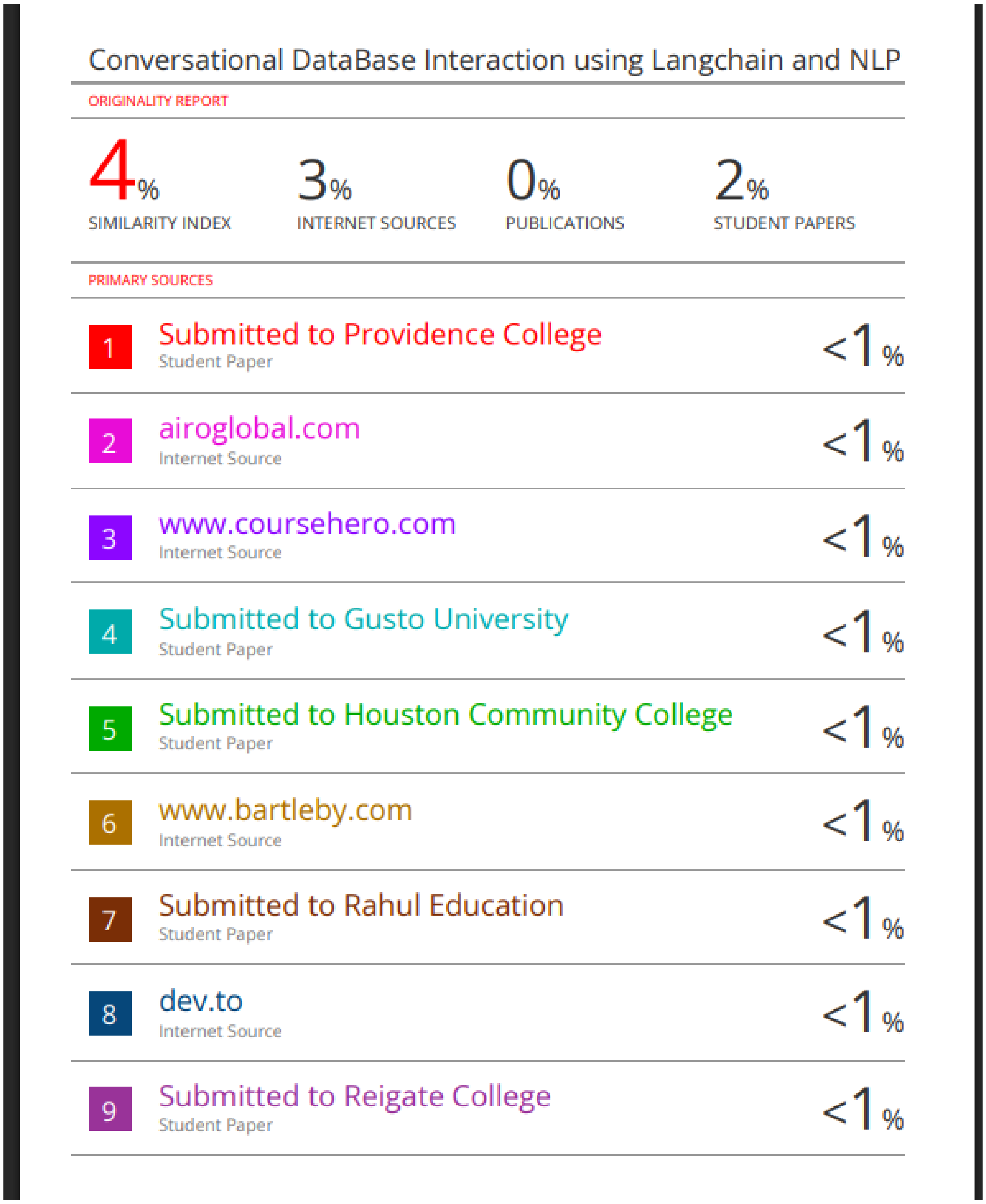
Assistant Professor, Department of CSE – DS,
Sitams, Chittoor, Andhra Pradesh.

TABLE 1: OUTCOME ATTAINED AND ITS JUSTIFICATION

PO	Justification
PO1	We gained strong knowledge in Natural Language Processing (NLP), LangChain, and how conversational AI can be applied to database systems.
PO2	We analyzed the challenges users face when interacting with databases using traditional SQL methods, especially those without technical knowledge.
PO3	We designed and developed a user-friendly system that allows natural language interaction with databases, replacing the need for manual SQL queries.
PO4	We referred to research papers, documentation, and real user experiences to identify pain points and build a solution based on practical data.
PO5	We implemented the project using modern tools like LangChain, OpenAI's language models, and integrated them with MySQL for effective data retrieval.
PO6	This project makes data access easier for everyone, increasing productivity and saving time in industries, supporting inclusive digital transformation.
PO7	By simplifying the way users interact with data, the system helps reduce errors and resource waste that often happen due to incorrect query writing.
PO8	We ensured that user data and queries are handled with integrity and privacy, following ethical standards throughout the development.

PO9	We collaborated as a team, dividing tasks like backend development, NLP integration, UI creation, and testing to complete the project efficiently.
PO10	While planning, developing, and presenting the project, our written and verbal communication skills improved significantly.
PO11	We carefully managed time and cost during development by choosing the right tools and optimizing workflows for faster and smoother implementation.
PO12	This project encouraged us to explore new technologies like LangChain and LLMs, and strengthened our interest in continuous learning in the AI and Data Science field.

PLAGARISM REPORT



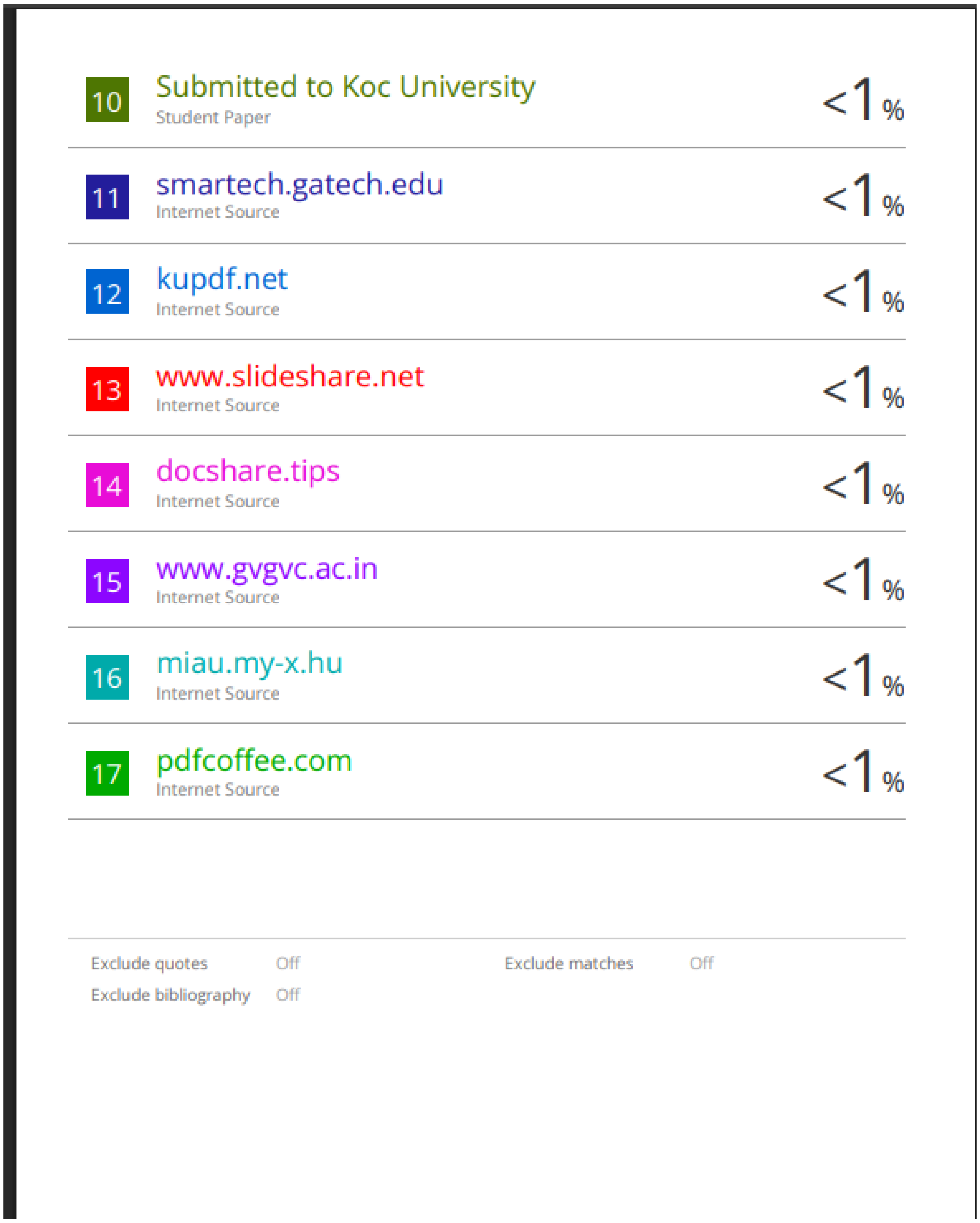
Conversational DataBase Interaction using Langchain and NLP

ORIGINALITY REPORT

4%	3%	0%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Providence College Student Paper	<1%
2	airoglobal.com Internet Source	<1%
3	www.coursehero.com Internet Source	<1%
4	Submitted to Gusto University Student Paper	<1%
5	Submitted to Houston Community College Student Paper	<1%
6	www.bartleby.com Internet Source	<1%
7	Submitted to Rahul Education Student Paper	<1%
8	dev.to Internet Source	<1%
9	Submitted to Reigate College Student Paper	<1%



10	Submitted to Koc University Student Paper	<1 %
11	smartech.gatech.edu Internet Source	<1 %
12	kupdf.net Internet Source	<1 %
13	www.slideshare.net Internet Source	<1 %
14	docshare.tips Internet Source	<1 %
15	www.gvgvc.ac.in Internet Source	<1 %
16	miau.my-x.hu Internet Source	<1 %
17	pdfcoffee.com Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		