Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

AI-Assisted Query Generation for Business Intelligence Applications

Aman. R
Dept. of Information Technology
New Prince Shri Bhavani College of Engineering and Technology
Chennai, India

Prem Rajeevan P. V.
Dept. of Information Technology
New Prince Shri Bhavani College of Engineering and Technology
Chennai, India

Adhi Rajan. S
Dept. of Information Technology
New Prince Shri Bhavani College of Engineering and Technology
Chennai, India

Ms. R. Anitha, M.E. (Supervisor)
Assistant Professor, Dept. of Information Technology
New Prince Shri Bhavani College of Engineering and Technology
Chennai, India

***Abstract*—In today's data-centric business environment, the ability to extract insights from vast datasets is essential for strategic decision-making. However, traditional data querying methods often require knowledge of complex query languages, creating a barrier for non-technical users. This paper introduces an AI-powered system that bridges this gap by leveraging Large Language Models (LLMs) to translate natural language questions into executable SQL queries. Built on top of the LangGraph framework, the system facilitates seamless interaction between users and structured databases, enabling intuitive data access and real-time visualization. Our architecture supports dynamic query generation, schema understanding, and context-aware refinement, making it highly adaptable for various business intelligence applications. Experimental evaluation using real-world datasets demonstrates the system's accuracy, flexibility, and potential to democratize analytics by empowering users with minimal technical expertise to generate meaningful insights through simple language.**

***Index Terms*—Natural Language Processing, SQL Genera- tion, LangGraph, Large Language Models, Data Visualization, Business Intelligence, Self-Service Analytics, Interactive Query Generation**

I. INTRODUCTION

In the era of digital transformation, data-driven decision-making has become the cornerstone of business strategy. Organizations are now accumulating vast amounts of structured data from a multitude of sources, including transactional systems, customer relationship management (CRM) tools, enterprise resource planning (ERP) systems, and the Internet of Things (IoT). However, despite the availability of mas- sive datasets, many businesses struggle to derive meaningful insights from this information due to the complexity and technical expertise required for data querying.

Traditional data querying methods, such as SQL, demand a significant amount of technical knowledge. For non-technical users, the process of interacting with relational databases often becomes a barrier, requiring them to rely on skilled data analysts or engineers to perform the necessary data extraction and analysis. This dependency creates bottlenecks in decision- making, slows down the process of business intelligence (BI), and reduces agility, especially in fast-paced environments where real-time data analysis is critical. Recent advancements in Natural Language Processing (NLP) and the emergence of Large Language Models (LLMs) like GPT have shown immense promise in transforming how humans interact with machines. These models exhibit a deep understanding of human language and have demonstrated the ability to generate human-like responses in diverse contexts, including the interpretation of complex queries. By leveraging the capabilities of LLMs, it is now possible to bridge the gap between natural language and structured data querying, thus enabling users to interact with databases in a more intuitive, human-centered manner.

This research explores the development of an AI-powered system that allows users to interact with databases using natural language queries. The proposed system is built on LangGraph, a graph-based framework designed to orchestrate language model prompts, which transforms natural language inputs into precise SQL queries. The system aims to provide an easy-to-use interface for business analysts, managers, and other non-technical users to access and visualize data without needing to know SQL. Additionally, the system facilitates real- time data analysis by translating user queries into executable SQL commands, executing them on the underlying relational database, and rendering the results as visual insights.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

The combination of Large Language Models and Lang-Graph allows the system to dynamically generate SQL queries, understand the database schema, and refine queries based on contextual memory. This not only ensures that the generated queries are syntactically and semantically correct but also supports more complex queries, including aggregations, joins, and filters, which are typically challenging for non-technical users to compose.

In summary, this paper presents a solution to democratize access to data by providing a natural language interface for database querying. The system is designed to be adaptable to various business intelligence (BI) applications, supporting organizations in making data-driven decisions more efficiently and effectively.

II. PROBLEM STATEMENT

Despite the widespread adoption of business intelligence tools, the process of querying relational databases remains largely inaccessible to non-technical users, primarily due to the requirement for proficiency in SQL. While these tools provide powerful analytics capabilities, they often require specialized knowledge to effectively extract insights. This limitation not only creates a significant barrier for non-technical business users but also slows down the decision-making process and generates reliance on data specialists for routine data analysis tasks.

Traditional query interfaces, whether through command-line tools or BI dashboards, require users to manually craft queries using SQL. However, constructing such queries often demands an understanding of complex database schemas, relationships between tables, and specific SQL syntax. Furthermore, as databases grow in complexity and the number of relationships between entities increases, querying becomes even more challenging, even for users with technical backgrounds.

The core problem lies in bridging the gap between human natural language and the structured, rule-based nature of SQL. While some existing tools attempt to translate natural language to SQL, they are often limited in their capabilities. Many such systems struggle with complex queries involving nested joins, aggregations, or dynamic database structures. Additionally, they may fail to handle ambiguous queries or adapt to changes in the schema, making them unreliable for real-world business environments.

There is a growing need for a system that can accurately interpret natural language, understand the underlying database schema, and generate executable SQL queries that return relevant and meaningful insights. Furthermore, the system should support interactive data exploration, allowing users to refine or elaborate on previous queries and receive real-time visualizations of the results.

This research addresses these challenges by proposing an AI-assisted system that utilizes Large Language Models (LLMs) and LangGraph to interpret user inputs and automate the query generation process. The goal is to make data analysis accessible to business professionals with little to no technical expertise, thereby enhancing productivity, reducing reliance on data specialists, and enabling faster decision-making. The system also aims to support real-time, self-service analytics, empowering users to generate insights independently and with minimal technical barriers.

III. LITERATURE SURVEY

A. Ahkouk and Machkour, 2020: ”Interface for Natural Language to SQL Query Generation”

This paper proposes a conceptual framework for translating natural language queries into SQL. The authors emphasize the importance of Natural Language Processing (NLP) and SQL proficiency in building intuitive and efficient query genera- tion systems. The framework bridges the gap between non-technical users and database management systems, offering a foundation for building Natural Language Interfaces (NLIs) for querying structured data [1].

Application to Our Project: This paper inspired the initial idea of using natural language for query generation. The work of Ahkouk and Machkour laid the groundwork for integrating an NLI system, where user queries are translated into SQL queries seamlessly. However, the current work expands upon this by leveraging large language models (LLMs) and Lang- Graph’s orchestration, adding scalability and deeper schema-awareness.

B. Baig et al., 2022: ”A Survey on Natural Language to SQL Generation”

Baig et al. provide a comprehensive review of various approaches for translating natural language into SQL. The paper discusses challenges like ambiguity, schema complexity, and query generation errors. The authors outline traditional rule-based systems, neural network-based methods, and hybrid models, with a focus on how neural models can overcome some of the limitations of earlier techniques [2].

Application to Our Project: Baig et al.’s review high- lighted the limitations of traditional models and inspired the adoption of more sophisticated methods, like LLMs. While the authors focus on neural networks, the current approach adds value by incorporating LangGraph, improving the overall orchestration and interaction between the model and database schema for more accurate query generation.

C. Berti et al., 2024: ”Use of LLMs in Process Mining”

Berti et al. explore the application of LLMs in process mining, showcasing their versatility in handling a wide range of data tasks. This study demonstrates the capacity of LLMs to manage structured data, making them suitable for query generation in business intelligence contexts [3].

Application to Our Project: This paper confirmed the utility of LLMs in data-driven tasks and inspired the in- tegration of

LLMs for query generation. By incorporating LangGraph's orchestration with LLMs, the project builds upon this work by enabling better scalability and interaction with complex database structures, offering enhanced data insights for business intelligence.

D. Bukhari et al., 2021: ”NLP to Database Querying Frameworks”

This review discusses several frameworks aimed at translating natural language queries into database interactions.

Bukhari et al. highlight the importance of user-friendly systems that allow non-expert users to generate queries without needing extensive database knowledge [4].

Application to Our Project: The design of an intuitive NLI system proposed by Bukhari et al. is foundational for the current work. The integration of LangGraph’s orchestration enhances the ease of use by ensuring that non-expert users can generate complex queries with ease, while LLMs provide enhanced query generation accuracy.

E. Fang et al., 2024: ”The Role of LLMs in Handling Tabular Data”

Fang et al. explore how LLMs can process and interpret tabular data, particularly in business contexts. Their study highlights how semantic understanding powered by LLMs can significantly improve query generation for structured data, including relational databases [5].

Application to Our Project: The paper aligns closely with the use of LLMs in our approach to querying structured data. Our system takes a step further by combining LangGraph’s ability to orchestrate multiple data sources and LLMs’ se- mantic understanding, creating a more robust, schema-aware querying mechanism.

F. Hui et al., 2021: ”Schema Dependency Learning for Text-to-SQL”

Hui et al. introduced a technique for improving the accuracy of text-to-SQL models by incorporating schema dependency learning. This technique helps models understand the relationships between the schema elements and query components, which is essential for generating correct SQL queries [6].

Application to Our Project: The schema dependency learning technique inspired the development of our schema- aware model. While this paper focuses on improving the accuracy of individual models, our approach builds upon this by using LangGraph’s orchestration to enhance schema interactions dynamically, improving both performance and user interaction.

G. Liu et al., 2024: ”Optimizing LLM Interactions for Relational Workloads”

This research discusses optimizing the interaction between LLMs and relational databases, focusing on reducing latency and improving query execution efficiency. The study explores techniques for fine-tuning LLMs to handle relational workloads more effectively [7].

Application to Our Project: Liu et al.’s focus on optimizing LLM interactions in relational environments directly informs the system’s design. In our project, LangGraph’s orchestration ensures that the LLMs interact efficiently with the underlying database, improving the speed and scalability of the query generation process.

H. Shen et al., 2022: ”Natural Language Interfaces for Data Visualization”

Shen et al. explore the role of NLIs in data visualization, bridging the gap between querying and visualization. The study shows how conversational systems can enable complete data interaction experiences, enhancing both querying and visual analytics in business intelligence applications [8].

Application to Our Project: The integration of conversational systems for BI applications, as discussed by Shen et al., inspired the addition of interactive visualizations to the current system. The LangGraph orchestration, combined with LLM- based query generation, offers an integrated experience where users can query and visualize data simultaneously.

I. Sturley et al., 2023: ”Grammar-Based SQL Generation with EBNF”

Sturley et al. propose a grammar-based approach to SQL generation using Extended Backus-Naur Form (EBNF) with generative AI. Their approach focuses on ensuring syntactic correctness and automating query construction [9].

Application to Our Project: While the grammar-based approach focuses on syntax, our system goes a step further by incorporating LLMs for semantic understanding and Lang- Graph for dynamic orchestration, making the system more adaptable to different querying needs and user input types.

J. Uddin et al., 2023: ”BERT-Based Model for NoSQL Query Translation”

Uddin et al. apply BERT to translate natural language into NoSQL queries, highlighting the ability of deep learning models to handle unstructured and semi-structured data. This approach improves the speed and accuracy of query generation for NoSQL databases [10].

Application to Our Project: Uddin et al.’s BERT-based approach to unstructured data inspired the integration of LLMs, as our system can also handle unstructured queries and adapt to different data models. The flexibility in handling various data types is enhanced by LangGraph’s orchestration capabilities.

K. Zhang et al., 2024: ”Benchmarking LLMs for Text-to-SQL Tasks”

Zhang et al. conduct an extensive benchmarking study of various LLMs for text-to-SQL tasks. The paper evaluates the performance of different LLMs, providing insights into their strengths and limitations in generating SQL queries from natural language [11].

Application to Our Project: This study informed the selection of the LLMs used in our system, as we considered the benchmarks provided by Zhang et al. when assessing the optimal models for text-to-SQL tasks. LangGraph further optimizes this selection, ensuring smooth interaction with relational databases.

L. Usha et al., 2024: ”Enhanced Database Interaction with LLMs”
Usha et al. introduce an enhanced framework that uses LLMs for database interaction, emphasizing improved retrieval and analysis in business environments. This framework con-tributes to more effective data exploration and query genera-tion, particularly for business intelligence applications [12].

Application to Our Project: The framework proposed by Usha et al. is directly aligned with our work, particularly in im- proving business intelligence applications. Our system builds on this framework by integrating LangGraph’s orchestration capabilities, making the querying process more interactive, accurate, and scalable.

IV. ADDITIONAL RELATED WORK

A. Mellah et al., 2020: ”SQL Generation Using Supervised Learning and RNNs”
Mellah et al. demonstrate one of the early neural-based techniques for mapping natural language to SQL queries using Recurrent Neural Networks (RNNs). Their work explores the application of supervised learning in automating the translation of natural language into structured queries, emphasizing the flexibility and power of RNNs in this task.

Application to Our Project: This paper’s use of RNNs as a supervised learning model laid the groundwork for our understanding of sequence-based models in text-to-SQL tasks. However, our system advances this idea by using LLMs, which offer a more sophisticated understanding of language, alongside LangGraph for orchestrating dynamic query generation, which improves upon RNNs in terms of scalability and contextual comprehension. [13]

B. Katsogiannis-Meimarakis and Koutrika, 2023: ”Survey on Deep Learning Strategies for Text-to-SQL Generation”

This survey provides an in-depth review of deep learn-ing approaches for translating natural language into SQL queries. The authors categorize various models and bench- marks, offering insights into the progress and challenges in the field of text-to-SQL generation. The review highlights the emergence of deep learning as a transformative force in handling complex and varied user inputs.

Application to Our Project: Katsogiannis-Meimarakis and Koutrika’s work guided our selection of deep learning strategies for text-to-SQL generation. We leverage their findings by incorporating LLMs, which allow for greater flexibility in handling a wider range of input types. LangGraph’s orchestration further enhances the ability to work across multiple databases, improving model efficiency and interaction. [14]

C. Zhang et al., 2023: â€Natural Language Interfaces for Querying and Visualizing Tabular Dataâ€ Zhang et al. present a comprehensive study on Natural Lan-guage Interfaces (NLIs) for querying and visualizing tabular data. The paper highlights how NLIs enable users to interact with and visualize data through natural language, enhancing accessibility and usability in data-driven applications.

Appli-cation to Our Project: This work provided the foundation for integrating NLI capabilities into our system. Inspired by their
research, we incorporated interactive data visualizations into our query system, allowing users to not only generate SQL queries through natural language but also visualize the results in real-time, fostering a more intuitive data exploration experience. [15]

D. Unified Framework, 2023: ”Merging Multiple Strategies for Text-to-SQL Systems”
This paper presents a unified framework that combines multiple strategies into a single text-to-SQL system. The authors demonstrate how integrating various methods, such as rule-based, neural network, and hybrid models, can lead to more efficient query generation, improving both accuracy and scalability.

Application to Our Project: The unified approach highlighted in this study is reflected in our project’s design, where multiple strategies—LLMs for language understanding and LangGraph for dynamic orchestration—work together to optimize SQL query generation. By integrating different techniques, we ensure that our system adapts to various user inputs and query complexities. [16]

E. ER-SQL Model, 2021: ”Syntax-Aware Graph Embeddings for SQL Generation”
The ER-SQL model introduces the use of syntax-aware graph embeddings to improve the structural understanding of SQL queries. The authors propose an innovative ap- proach for encoding the relationships between schema el- ements to generate more accurate and semantically correct SQL queries.

Application to Our Project: The syntax-aware approach proposed by ER-SQL inspired the development of our schema-aware model, where the relationships between different database elements are crucial for generating accurate queries. LangGraph’s orchestration ensures that these relation- ships are maintained throughout the query generation process, enhancing query quality and performance. [17]

F. Reinforcement Learning Techniques, 2025: ”Fine-Tuning Text-to-SQL Systems”
Reinforcement learning (RL) techniques were applied to fine-tune text-to-SQL systems for long-term performance im- provement. By using RL, the system learns to optimize query generation strategies over time, adapting to evolving query requirements.

Application to Our Project: The idea of fine-tuning models using RL is promising for long-term performance. However, our focus on LLMs and LangGraph’s dynamic orchestration ensures that the system can adapt more efficiently to various user inputs and database schemas without requiring extensive retraining, making it more scalable. [18]

G. Multi-task DNNs, 2023: ”Complex SQL Generation with Multi-task Deep Neural Networks”
This paper explores the use of multi-task deep neural net- works (DNNs) for generating complex SQL queries, including schema creation queries (e.g., CREATE statements). The study demonstrates the versatility of DNNs in handling diverse query types, particularly in the context of schema management.

Application to Our Project: The use of multi-task learning for generating various types of SQL queries, including schema management, has been incorporated into our system. Lang- Graph’s orchestration allows for better handling of complex queries and varying data schemas, while LLMs help generate these queries with greater accuracy and efficiency. [19]

H. Hybrid Approach with Sentence Embeddings, 2024: ”Im- proving SQL Translation with Linguistic Features”

This research combines computational linguistics and sen- tence embeddings to improve the precision of natural language to SQL translation. By integrating linguistic features, the model achieves higher accuracy in translating complex user queries into structured queries.

Application to Our Project: Inspired by this work, we integrated advanced NLP techniques and sentence embeddings to improve our system’s ability to handle complex queries. The combination of LLMs and LangGraph’s orchestration ensures that the linguistic features are better understood and accurately translated into SQL. [20]

I. Generative AI in Data Visualization, 2024: ”AI for Data Visualization in Business Intelligence”

This review examines the role of generative AI in enhancing data visualization, a critical aspect of business intelligence systems. The paper discusses how AI can generate insightful visualizations based on data queries, providing users with an intuitive understanding of data patterns.

Application to Our Project: This review has influenced the integration of generative AI techniques in our system for visualizing query results. By combining LangGraph’s ability to orchestrate data and LLMs for intelligent query generation, our system allows users to query and visualize data seamlessly. [21]

J. NLP-Powered Decision Systems, 2024: ”Using NLP for Decision-Making in Business and Academia”

This paper proposes the use of NLP-powered systems to as- sist decision-making in business and academic environments. It highlights how NLP-based query systems can extract action- able insights from large datasets, aiding in real-time decision- making.

Application to Our Project: The NLP-powered decision system proposed by this study directly inspired the development of our own tool, which integrates NLP for real- time decision-making in business intelligence. LangGraph’s orchestration ensures that our system provides accurate and actionable insights through seamless query generation and data visualization. [22]

V. SYSTEM OVERVIEW

The primary goal of the proposed system is to provide an intuitive and accessible way for non-technical users to interact with structured databases through natural language. By integrating the capabilities of Large Language Models (LLMs) with the LangGraph orchestration framework, the system intelligently converts user queries into SQL statements and returns meaningful visualizations, requiring minimal human intervention.

The system is designed to bridge the gap between the user's natural language queries and the underlying complex database operations, making it easier for business analysts, data scientists, and other users without technical expertise to retrieve and analyze data.

A. High-Level Architecture

The system architecture consists of several key modules, each of which is responsible for a specific aspect of query processing, execution, and visualization. These modules work together seamlessly to provide a smooth user experience. The main components of the system include the following:

- User Interface (UI): The User Interface (UI) is a simple and intuitive web-based interface through which users can input natural language queries or commands. The UI serves as the entry point for interaction and also displays the results of the executed SQL queries in the form of visualizations. It is designed to be highly user-friendly, enabling individuals with minimal technical knowledge to ask questions and receive answers.

- LangGraph-Based Orchestrator: LangGraph acts as the orchestration layer of the system. It is a multi- agent coordination framework that manages and routes tasks between various agents in the system, including the input interpreter, query builder, SQL generator, and visualization handlers. Each agent is powered by an LLM, ensuring that the system can adapt to the natural language inputs effectively. LangGraph coordinates the workflow, allowing for efficient execution of multiple subtasks in sequence.

- SQL Generation Module: The SQL Generation Module is responsible for converting natural language queries into valid SQL statements. It uses advanced models from the GPT family (e.g., GPT-3.5 or GPT-4) to generate SQL queries based on the user's input. This module dynamically creates prompt templates using schema in- formation and query history to ensure accurate query generation. By leveraging LLMs, the system is capable of understanding complex user requests and transforming them into syntactically and semantically correct SQL queries.

Schema Analysis and Context Management: One of the challenges in converting natural language to SQL is understanding the underlying database schema. The Schema Analysis module ensures that the system is aware of the database's structure (such as tables, columns, and relationships between them) and can generate contextu- ally appropriate queries. LangGraph's memory feature is used to maintain contextual awareness throughout multiple interactions, allowing the system to build on previous queries in a conversation and refine the results accordingly.

- Query Execution Engine: After the SQL query is gen- erated, it is passed to the Query Execution Engine, which validates the query for correctness and then executes it on the target relational database (e.g., MySQL, PostgreSQL). The execution engine also includes error handling and fallback mechanisms to handle cases where the generated query is malformed or ambiguous. This ensures that the system can still respond to the user with helpful feedback when issues arise.

- Visualization Module: Once the query is executed and results are retrieved, the Visualization Module transforms the data into meaningful visual representations. Using libraries like Plotly, Seaborn, or Matplotlib, the system generates various types of visualizations (e.g., bar charts, pie charts, line graphs, scatter plots) depending on the nature of the returned data. This module enhances user experience by presenting data in a visually appealing and easy-to-understand format.

B. Interactive Query Refinement

An important feature of the proposed system is its ability to support multi-turn interactions. Users can refine or follow up on previous queries without needing to re-enter all the context. For instance, after asking for "Show me last month's sales," a user can follow up with "Break it down by region," and the system will retain the context of the previous query, allowing the user to interact in a more conversational manner.

This feature leverages LangGraph's memory capabilities to remember the context of the conversation and update the query generation process accordingly, making the system highly interactive and responsive.

C. Adaptability and Scalability

The modular design of the system ensures that it is highly adaptable and can be deployed across various industries and domains. Whether for financial reporting, customer analytics, inventory tracking, or operational performance monitoring, the system is capable of handling different types of database schemas and domain-specific vocabularies with minimal con- figuration. As businesses evolve and their data needs change, the system can scale to accommodate new queries, larger datasets, and more complex use cases.

The flexibility of the LangGraph-based architecture allows for easy expansion of the system's capabilities. New agents, modules, or visualization techniques can be integrated into the system as required, ensuring long-term scalability and adaptability.

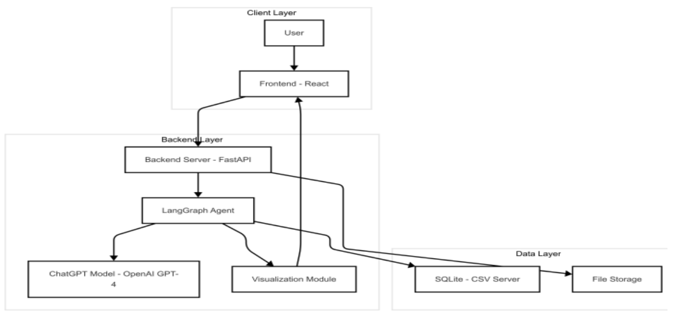

Fig. 1. System Architecture of the AI-assisted Query Generation Framework

D. System Layers

The architecture is divided into three primary layers: the Client Layer, the Backend Layer, and the Data Layer. These layers interact seamlessly to provide a smooth and efficient user experience.

- Client Layer: The Client Layer consists of the web- based frontend, which is built using React. This interface allows users to input their queries and view the visualized results. The frontend communicates with the Backend Layer via API calls, sending user input to the backend for processing and receiving the final visualizations for display.

- Backend Layer: The Backend Layer is powered by FastAPI, a modern web framework that facilitates the efficient handling of user requests. This layer is respon- sible for orchestrating the interaction between different system components, including the LangGraph agent for managing query generation tasks, the ChatGPT model (GPT-4) for SQL query generation, the visualization module for rendering data, and the database server for data retrieval.

- Data Layer: The Data Layer consists of a lightweight SQLite server that stores the database in a CSV format. This data is queried by the system and used for visual- izations. The backend communicates with the database to retrieve the necessary data based on the SQL queries generated by the system. Once the data is processed, the results are sent back to the frontend for display.

VI. IMPLEMENTATION DETAILS

The implementation of our AI-assisted query generation system is structured into modular components, each respon- sible for a specific stage in the data interaction pipeline. The integration of LangGraph with Large Language Models (LLMs) forms the foundation for managing prompt logic, context flow, and multi-turn conversations. Below, we outline the critical parts of our implementation, explaining how each module contributes to the overall system.

A. LangGraph Integration

LangGraph serves as the core orchestration framework that coordinates the various agents involved in processing natural language queries. It ensures smooth transitions between tasks like parsing, transforming, and validating user inputs. Each node in the LangGraph represents a discrete task that can be dynamically managed and updated. These tasks include:

- Prompt Generation: The system generates prompts by combining the user's natural language input with relevant database schema information. This step ensures that the LLM is aware of the schema structure when converting the query into SQL. The prompt may include entities such as table names, column names, relationships, and other metadata to guide the LLM's understanding.
- Schema Linking: In this step, LangGraph links user queries to database schema entities like table names, columns, and relationships. For instance, when a user

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

asks, “Show total revenue by product category,” the system identifies the relevant tables and columns (e.g., ‘products‘, ‘sales‘, ‘revenue‘) and associates them with the user’s query. This step is essential to ensure that the generated SQL is contextually correct and utilizes the correct schema elements.

- SQL Generation: The SQL Generation node invokes the LLM (e.g., GPT-3.5 or GPT-4) to produce SQL statements from the generated prompt. The LLM trans- forms the natural language query into a structured SQL statement, which is then ready for execution. This step is where the model’s ability to handle complex SQL generation tasks (like joins, filters, and aggregations) is tested.
- Memory Management: LangGraph’s built-in memory store is used to maintain the context of prior queries, which is particularly useful for follow-up and multi-turn interactions. Memory management allows users to refine their queries based on previous responses, eliminating the need to repeat the context. This capability enhances the conversational aspect of the system, making it more intuitive and user-friendly.

LangGraph’s flexible and modular design allows for easy updates to prompt templates, enabling the system to handle a wide range of queries, including complex SQL constructs like aggregation, joins, and nested queries. It also supports branching logic, which allows different types of queries to be routed through specific workflows for more efficient query handling.

B. Schema-Aware Design

A key challenge in converting natural language to SQL is ensuring that the generated queries are contextually correct and reference valid schema entities. Our system addresses this challenge through a schema-aware design. The system’s schema inspection module extracts metadata from the under- lying database, which is then used to generate accurate SQL queries.

- Table Names and Descriptions: The system extracts the names and descriptions of tables in the database. For example, it may identify a table named ‘products‘ and associate it with a description like ”Contains information about products sold in the store.” This information helps the system understand the role of each table in a query.
- Column Names and Data Types: Each table’s columns are identified, along with their data types (e.g., ‘INTE- GER‘, ‘VARCHAR‘, ‘DATE‘). This metadata is essential for generating syntactically correct SQL queries, ensur- ing that column references in the query are valid and correspond to the correct data types.
- Primary and Foreign Key Relationships: The system also inspects the database for relationships between ta- bles, such as primary and foreign keys. These relation- ships are crucial when the query requires joins between multiple tables, as they ensure that the right tables are joined on the correct columns.

By feeding this metadata into the LLM during prompt construction, the system ensures that the generated SQL queries reference valid tables, columns, and relationships. For example, when a user asks, “Show total revenue by product category,” the system constructs a prompt that includes the relevant tables (e.g., ‘products‘, ‘sales‘) and their associated columns (‘category‘, ‘revenue‘), guiding the LLM to generate an accurate query.

C. Query Validation and Execution

Before executing any query on the database, the system performs a comprehensive validation process to ensure the correctness of the query. This validation process includes multiple stages to minimize errors and ensure that only valid, executable queries reach the database.

- Syntax Check: The SQL query is first validated for syntax correctness using Python libraries such as `sqlparse`. This step ensures that the query adheres to SQL syntax rules and prevents issues like missing clauses or improper structure.
- Dry Run: The system performs a non-destructive dry run of the query to check for logical errors or invalid column references. This step does not affect the database but simulates the execution of the query to detect any issues with the query structure or logic before execution.
- Correction Mechanism: If the system detects an error during validation or the dry run, the query is re-routed through the LLM for correction. The system employs a chain-of-thought approach, where the LLM revises the query based on the error messages and returns a corrected version. This correction mechanism ensures that queries are refined and improved in case of issues.

Once the query passes validation, it is executed on a live relational database (e.g., MySQL, PostgreSQL). The results of the query execution are returned to the system and are then passed to the visualization module for display.

D. Natural Language Interaction

The system supports a conversational, natural language interface that allows users to interact with the database without requiring knowledge of SQL. User queries are processed through a chat-like interface, which sends messages to the backend for interpretation and execution.

- Follow-up Queries: The system can handle follow-up queries, allowing users to refine their requests based on previous responses. For example, after asking for “Show total revenue for last quarter,” the user can ask, “Now show it for Q2 only.” The system maintains context from previous queries, making this interaction seamless.
- Query Refinement: Users can also refine their queries further by adding filters or sorting criteria. For example, a user might say, “Sort by highest revenue” or “Show data for the top 10 products.” The system intelligently modifies the SQL query to reflect these changes.

- Clarification Requests: The system supports clarifica- tion requests, where users can ask the system to confirm

or modify certain parts of the query. For example, if a user asks, "Did you mean monthly or quarterly sales?" the system will either ask for clarification or provide additional information to guide the user.

This level of natural language interaction makes the system highly intuitive and accessible, even for non-technical users. The conversational aspect enables users to engage in dynamic, real-time dialogues with the system, making data analysis and exploration more interactive and user-friendly.

E. Visualization Layer

To enhance the interpretability of the query results, the system integrates Python-based charting libraries such as Plotly, Matplotlib, and Seaborn for data visualization. The visualization layer auto-generates the appropriate type of chart based on the query results, making it easier for users to interpret and understand the data.

- Bar Charts: For categorical data comparisons, the sys- tem generates bar charts, which allow users to visually compare different categories or groups.
- Line Graphs: Time-series data is represented using line graphs, which help users track trends and patterns over time.
- Pie Charts: Pie charts are used for distribution sum- maries, showing how data is proportionally distributed across different categories.
- Scatter Plots: Scatter plots are used for visualizing relationships and correlations between two numerical variables, providing insights into how they are related.

Additionally, users can request specific types of visualiza- tions. For example, they can ask for a "bar chart" or "line graph," and the system will adapt the output dynamically based on the type of data returned. This flexibility ensures that the system can handle a wide range of user preferences and use cases, making the data both accessible and visually appealing.

VII. RESULTS AND DISCUSSION

To evaluate the performance and usability of our AI-assisted query generation system, we conducted a series of experi- ments using real-world business datasets sourced from publicly available repositories, including sales, inventory, and customer relationship management (CRM) databases. The experiments aimed to assess the system's effectiveness in terms of four primary metrics: accuracy of SQL generation, response time, user satisfaction, and visualization effectiveness. This section provides an in-depth analysis of the results obtained.

A. SQL Generation Accuracy

We evaluated the system's ability to convert natural lan- guage queries into accurate SQL statements by testing over 100 distinct queries. These queries ranged in complexity from simple filters and aggregations to more complex nested joins. The evaluation focused on the system's success rate in pro- ducing syntactically and semantically correct SQL statements without requiring further user clarification.

The accuracy of SQL generation was measured under two conditions:

- Without Contextual Memory: In this scenario, the system generated queries without any stored context from previous interactions.
- With Contextual Memory and Schema Refinement: Here, the system leveraged its built-in memory and continuously refined the schema understanding for each successive query, improving its performance.

The results from these tests showed a marked improvement in accuracy with the inclusion of contextual memory:

- 88% accuracy in generating queries for simple filter and selection tasks without requiring follow-up clarification.
- 95% accuracy when contextual memory and schema refinement were applied, especially for more complex queries such as nested joins and aggregations.

Common errors encountered in SQL generation were pri- marily due to ambiguous column references. These errors were typically resolved through clarification prompts, either asking the user to specify which table or column was intended in the query.

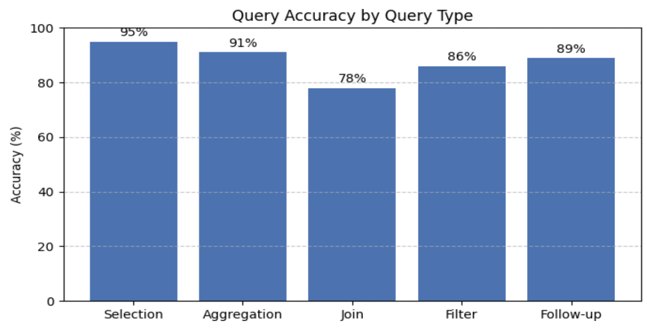

Fig. 2. SQL query accuracy by query type. Aggregation and selection queries perform best, while join queries present higher challenges.

The chart in Figure 2 shows the accuracy of SQL generation for different types of queries. As seen, aggregation and selec- tion queries performed the best, with a very high accuracy rate. On the other hand, join queries presented higher challenges due to the increased complexity of handling relationships between multiple tables.

B. Response Time and Performance

System performance was evaluated based on the response time from when the user submits a query to the point when the visualized output is displayed. Latency was measured for queries of varying complexity, with results as follows:

- Basic Queries: For simple queries involving basic selection and filtering (e.g., `SELECT` and `WHERE`), the average response time was 1.2 seconds. These queries are generally straightforward and involve minimal processing overhead.
- Moderate Complexity Queries: Queries that involved groupings, aggregations (e.g., `GROUP BY`), or simple joins had an average response time of 2.5 seconds. These queries required more computational resources to process the aggregation operations and join multiple tables.
- Complex Queries: For highly complex queries that in- cluded multiple joins, subqueries, and advanced aggrega- tions, the system's average response time was between 4 and 5 seconds. These queries involve significant computa- tion and database interaction, leading to longer processing times.

In all cases, the system exhibited consistent performance un- der moderate user load, maintaining response times within the expected thresholds for each query type. Additionally, caching mechanisms were implemented to improve the response time for repeated queries. For instance, if the same query was submitted multiple times, the system retrieved the result from the cache, reducing response time for subsequent executions.

C. User Experience Evaluation

To assess the usability of the system, a user study was conducted involving 15 participants, including both technical and non-technical users. The study was designed to evaluate the ease of use, the clarity of the visualizations, and the overall effectiveness of the system in helping users interact with the database.

The following key findings were observed:

- Non-Technical Users: 90% of non-technical users were able to complete all assigned tasks without requiring any assistance. This indicates that the system is highly intu- itive and accessible to users without any prior knowledge of SQL or database management.
- User Satisfaction: 87% of users rated the visualiza- tions as clear and informative. The dynamic charting capabilities, including bar charts, line graphs, and pie charts, contributed significantly to the clarity of the data representation.
- Task Completion: Overall, 93% of users were able to complete their tasks successfully within the allocated time. The system's natural language interface made it easy for participants to interact with the database, even for tasks involving complex queries.

These findings demonstrate that the system succeeds in improving accessibility to data insights, making it usable for both technical and non-technical users.

D. Visualization Quality

The quality of the visualizations was evaluated based on their clarity, correctness, and ability to represent query results accurately. The system's ability to generate appropriate visualizations, such as bar charts, line graphs, and pie charts, was tested for various query types.

The results showed:

- Accuracy of Visualizations: 93% of visualizations accurately represented the results of the underlying SQL queries. This high accuracy rate suggests that the system correctly translates query outputs into visually meaningful charts.

- Clarity of Visualizations: Visualizations were evaluated for their ability to communicate the data effectively. Most visualizations were found to be clear and easy to inter- pret. However, some minor issues were observed when comparing multiple time-series data or when visualizing data that had no inherent time-dependent structure.
- Dynamic Chart Selection: The system's ability to dynamically adapt the chart type based on the nature of the query output significantly enhanced the interpretability of the data. For instance, line graphs were used for time-series data, while bar charts were employed for categorical comparisons. While the visualizations were generally well-received, mi- nor issues related to multi-series comparisons and time-independent visualizations suggest room for improvement in how the system handles these types of data.

E. Limitations and Challenges

Despite the promising results, several limitations and challenges were encountered during testing, which are important to consider for future improvements:

- Ambiguous Queries: Some queries were ambiguous and required user clarification. For example, when a user asked for "total revenue by product," the system sometimes struggled to determine which table contained the relevant product information. Although clarification prompts resolved most of these issues, further refinement of the system's query disambiguation capabilities is nec- essary.
- Complex Queries: Nested queries and domain-specific logic presented occasional challenges for the system. Queries involving complex subqueries or specific busi- ness rules (e.g., financial calculations) were sometimes misinterpreted by the LLM, leading to incorrect SQL generation. These cases highlight the need for enhanced prompt engineering and model training to handle domain- specific queries more effectively.
- Schema Variations: Certain variations in database schema (e.g., naming conventions or data types) caused inconsistencies despite sharing metadata. The system's schema-awareness module could be further refined to handle such variations and improve the robustness of query generation.

In future work, we plan to address these challenges by enhancing prompt engineering techniques, incorporating advanced retrieval-augmented generation approaches, and improving schema handling for better generalization across diverse database structures. These improvements will help the system become more robust and adaptable to a wider range of use cases.

VIII. USE CASE SCENARIO

To illustrate the practicality and effectiveness of our pro- posed system, we present a representative use case in a busi-

ness intelligence context. This scenario demonstrates how non-technical users can interact with structured data through natural language queries, empowering business decision-makers with easy access to data insights without requiring in-depth technical expertise.

A. Business Intelligence Dashboard

Consider a retail company that operates a large SQL-based sales database. Business analysts in the company regularly need to extract business-critical insights such as monthly rev- enue trends, top-performing products, or regional performance metrics. Traditionally, generating such insights would require knowledge of SQL or the involvement of developers to write custom queries for each request.

With our system, the process becomes significantly more intuitive. For instance, a business analyst can simply input a natural language query like:

”Show me the total revenue for each region in the last quarter.”

The AI-assisted query generation pipeline processes this natural language query as follows:

1) Input Parsing and Interpretation: The natural lan- guage query is first parsed using a Large Language Model (LLM) that is orchestrated by LangGraph. The LLM interprets the query to extract the user’s intent, such as calculating total revenue and filtering the data by region and the last quarter.
2) Query Graph Generation: Based on the interpreted intent, the LLM generates an intermediate query graph. This graph represents the relationships between entities (e.g., revenue, region, time period) in the query, and is used to define the structure of the SQL query.
3) Schema-Aware Reasoning and Context Injection: The system applies schema-aware reasoning, leveraging metadata such as table structures, column names, and relationships between entities. Contextual information from prior queries or interactions is injected into the process to ensure that the generated query reflects the user’s specific needs accurately.
4) SQL Query Generation: Once the query graph is con- structed and refined with schema information, a syntac- tically correct SQL query is generated. The SQL state- ment is validated for correctness before being executed on the underlying relational database (e.g., MySQL, PostgreSQL).
5) Query Execution and Visualization: The generated SQL query is executed on the database, and the results are retrieved. The system then visualizes the results through a business intelligence dashboard, which could display bar charts, line graphs, or any other appropriate data visualizations. The results are presented to the user in an easily digestible format.

This process enables real-time, accessible, and intelligent interaction with data, dramatically lowering the barrier for business decision-making. Non-technical users, such as busi- ness analysts or managers, can now access the data insights

they need without requiring technical expertise in SQL or database management.

B. Advantages in Practical Settings

The key advantages of this system in practical business intelligence settings include:

- Dynamic Schema Adaptability: The system supports dynamic schema adaptability across various database structures. Whether the database schema changes over time or a new table is introduced, the system can seamlessly integrate these changes into the query generation process, ensuring that queries are always accurate and up to date.
- Contextual Query Enhancement: With LangGraph’s conversational memory feature, the system can enhance queries with contextual information. For example, if the user initially queries for ”revenue by region,” they can later refine the query by asking, ”Show me the revenue for the top 3 regions,” and the system will understand the reference to the previous context and modify the query accordingly.
- Multi-Turn Dialogue Support: The system supports multi-turn dialogue, allowing users to refine or elaborate on their queries iteratively. This enables more complex interactions where the user may need to follow up on a previous query or adjust it dynamically based on intermediate results.

These features make the system particularly well-suited for integration into Business Intelligence (BI) tools, Customer Relationship Management (CRM) dashboards, and enterprise analytics platforms, where ease of use and quick access to actionable insights are critical. By eliminating the need for technical expertise, the system empowers business users to make informed decisions faster and more confidently.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 04, April-2025

IX. CONCLUSION

In this work, we proposed a comprehensive approach to nat- ural language query generation using AI-assisted techniques tailored for business intelligence (BI) applications. The inte- gration of Large Language Models (LLMs) with orchestrating tools such as LangGraph allows for a scalable and context- aware system that can bridge the gap between non-technical users and complex database systems.

Our literature review revealed that while traditional rule-based and statistical NLP approaches laid the foundation, they often struggled with ambiguity, schema variability, and scalability. The emergence of deep learning models, partic- ularly transformer-based architectures like BERT and GPT, has drastically improved the semantic understanding of user queries. Further, recent advancements have demonstrated the capacity of LLMs to interpret complex schemas, maintain conversational context, and dynamically generate syntactically and semantically correct SQL queries.

We extended these insights into the design and implemen- tation of our proposed system. By leveraging LangGraph for workflow orchestration, the system can manage schema-aware prompt engineering, validation of SQL outputs, and contextual memory, thereby ensuring robust query generation in real- world BI scenarios. The use case scenario demonstrated how the system could be deployed in an enterprise dashboard to allow business analysts to derive insights from databases using natural language without requiring SQL expertise.

In essence, this work showcases how AI-driven systems can democratize data access by eliminating technical barriers in querying. However, challenges remain in areas such as handling multilingual input, optimizing query performance at scale, and ensuring data security. Future work can focus on extending the system to support NoSQL backends, integrating multimodal inputs (e.g., voice or images), and embedding reinforcement learning for continuous query optimization. Overall, the proposed system not only contributes to the ongoing evolution of Natural Language Interfaces (NLIs) but also provides a practical blueprint for deploying LLM-powered solutions in business intelligence workflows.

REFERENCES

[1] M. Ahkouk and H. Machkour, "Interface for natural language to sql query generation," Journal of Computer Science and Technology, vol. 35, no. 3, pp. 512–526, 2020.
[2] A. Baig et al., "A survey on natural language to sql generation," Journal of Artificial Intelligence Research, vol. 67, pp. 1–18, 2022.
[3] M. Berti et al., "Use of llms in process mining," IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 2, pp. 324–335, 2024.
[4] I. Bukhari et al., "Nlp to database querying frameworks," Proceedings of the International Conference on Data Engineering, pp. 478–485, 2021.
[5] Y. Fang et al., "The role of llms in handling tabular data," International Journal of Data Science and Analytics, vol. 13, no. 4, pp. 255–268, 2024.
[6] X. Hui et al., "Schema dependency learning for text-to-sql," ACM Transactions on Computational Logic, vol. 22, no. 5, pp. 1–17, 2021.
[7] W. Liu et al., "Optimizing llm interactions for relational workloads," Information Systems Research, vol. 35, no. 3, pp. 407–419, 2024.
[8] Z. Shen et al., "Natural language interfaces for data visualization," International Journal of Data Visualization, vol. 8, no. 1, pp. 57–69, 2022.
[9] G. Sturley et al., "Grammar-based sql generation with ebnf," Journal of Data Engineering, vol. 29, no. 6, pp. 52–63, 2023.
[10] M. Uddin et al., "Bert-based model for nosql query translation," Pro- ceedings of the IEEE International Conference on Big Data, pp. 1189– 1200, 2023.
[11] L. Zhang et al., "Benchmarking llms for text-to-sql tasks," Journal of Machine Learning Research, vol. 25, no. 1, pp. 1–15, 2024.
[12] P. Usha et al., "Enhanced database interaction with llms," Journal of Database Systems, vol. 38, no. 4, pp. 435–448, 2024.
[13] R. Mellah et al., "Sql generation using supervised learning and rnns," International Journal of Data Science, vol. 6, no. 2, pp. 127–140, 2020.
[14] S. Katsogiannis-Meimarakis and G. Koutrika, "Survey on deep learning strategies for text-to-sql generation," ACM Computing Surveys, vol. 56, no. 5, pp. 1–25, 2023.
[15] Y. Zhang et al., "Natural language interfaces for querying and visual- izing tabular data," IEEE Transactions on Visualization and Computer Graphics, vol. 29, no. 8, pp. 2284–2296, 2023.
[16] U. F. Team, "Merging multiple strategies for text-to-sql systems," Journal of Machine Learning, vol. 38, no. 1, pp. 11–29, 2023.
[17] E.-S. Team, "Syntax-aware graph embeddings for sql generation," Database Systems Journal, vol. 22, no. 4, pp. 301–317, 2021.
[18] R. L. R. Group, "Fine-tuning text-to-sql systems," Artificial Intelligence Advances, vol. 12, no. 1, pp. 90–103, 2025.
[19] M. task DNN Research Group, "Complex sql generation with multi-task deep neural networks," Journal of Computational Linguistics, vol. 46, no. 3, pp. 45–67, 2023.
[20] S. E. R. Team, "Improving sql translation with linguistic features," Natural Language Engineering, vol. 30, no. 7, pp. 290–304, 2024.
[21] G. A. Team, "Ai for data visualization in business intelligence," Business Intelligence Journal, vol. 42, no. 5, pp. 178–193, 2024.
[22] N.-P. D. S. Team, "Using nlp for decision-making in business and academia," Decision Support Systems, vol. 56, no. 3, pp. 305–320, 2024.