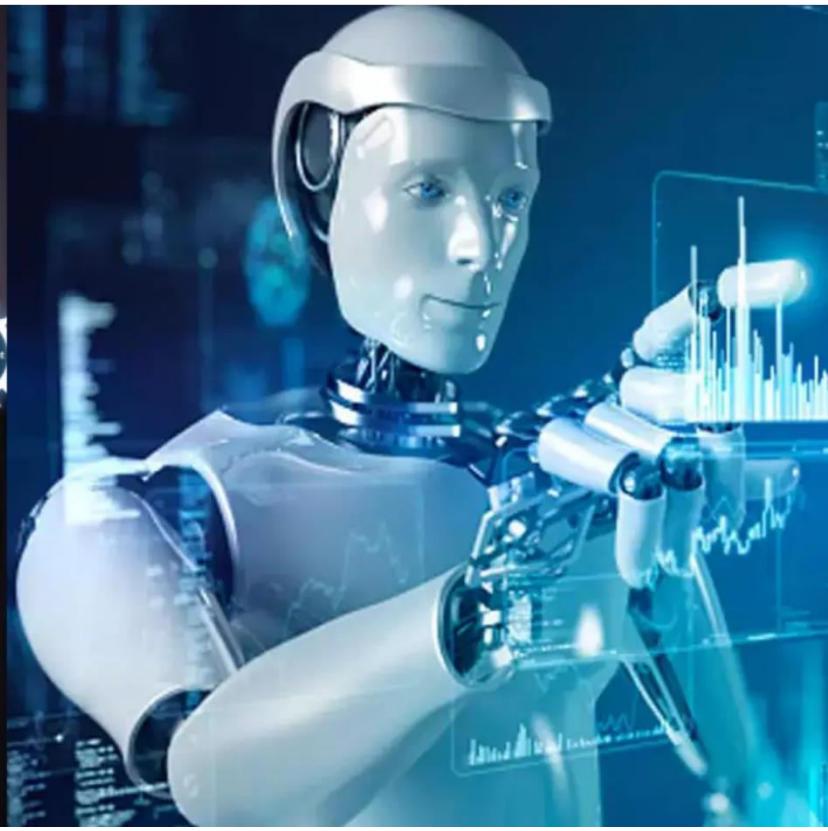
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.699

Volume 15, Issue 4, April 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

Intelligent SQL Querying with LLMs Using Gemini Pro

Mrs. Siva Tejaswi Jonna, M.Tech. (Ph.D.)[1], M. Devananda Raju[2], K. Pavani[2], S.Siva Karthik[2], Sk. Fayazuddin[2]

Assistant Professor, Kallam Haranadhareddy Institute of Technology, Guntur, Andhra Pradesh, India[1]

U.G Students, Department of CSE (Data Science), Kallam Haranadhareddy Institute of Technology, Guntur, Andhra Pradesh, India[2]

ABSTRACT: Intelligent SQL Querying with LLMs Using Gemini Pro In today's data-driven world, accessing and analyzing database information efficiently is essential. Traditional SQL querying requires users to have a strong understanding of database structures and query syntax, which can be a barrier for non-technical users. This project, IntelliSQL, aims to simplify database interaction by integrating Natural Language Processing (NLP) with advanced Large Language Models (LLMs) such as Gemini Pro. The system allows users to input queries in natural language, which are then automatically converted into structured SQL queries. By leveraging the capabilities of LLMs, the system understands user intent, context, and database schema to generate accurate and optimized SQL statements. The application is developed using modern web technologies, with backend support from frameworks like Python (Flask/Django) or Node.js, and integrates APIs for real-time query processing. Additionally, IntelliSQL enhances user experience by reducing the need for technical expertise, improving query efficiency, and minimizing errors. The system also includes features like query validation, error handling, and result visualization to provide a complete user-friendly solution. Overall, this project demonstrates how artificial intelligence can bridge the gap between human language and database systems, making data access more intuitive, efficient, and accessible to a wider audience.

KEYWORDS: IntelliSQL, Natural Language Processing (NLP), Large Language Models (LLMs) ,Gemini Pro ,SQL Query Generation ,Database Management Systems (DBMS) ,Natural Language to SQL (NL2SQL) ,Artificial Intelligence (AI) ,Machine Learning ,Query Optimization ,API Integration ,User-Friendly Interface

I. INTRODUCTION

In the modern digital era, data plays a crucial role in decision-making across various domains such as business, education, healthcare, and technology. Databases are widely used to store and manage this data efficiently. However, retrieving meaningful information from databases requires knowledge of Structured Query Language (SQL), which can be challenging for users without technical expertise. Traditional database systems rely on manually written SQL queries, where even a small mistake in syntax can lead to errors or incorrect results. This creates a barrier for non-technical users who want to interact with databases but lack programming knowledge. To overcome this limitation, there is a growing need for intelligent systems that can simplify database querying. This project, IntelliSQL, focuses on developing an intelligent SQL querying system that allows users to interact with databases using natural language. By leveraging advancements in Natural Language Processing (NLP) and Large Language Models (LLMs) such as Gemini Pro, the system can understand user queries in plain English and automatically convert them into accurate SQL statements. The proposed system bridges the gap between human language and database query languages, making data access more intuitive and user-friendly. It reduces dependency on technical skills, minimizes errors, and improves efficiency in retrieving data. The system is designed using modern technologies such as Python (Flask/Django) or Node.js for backend development and integrates APIs for seamless communication with the language model. Overall, IntelliSQL aims to transform the way users interact with databases by providing a smarter, faster, and more accessible solution for data retrieval.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

II. PROBLEM STATEMENT

In today's data-centric environment, databases are essential for storing and managing large volumes of information. However, accessing this data requires users to write Structured Query Language (SQL) queries, which demands a good understanding of database schemas, relationships, and query syntax. This creates a significant challenge for non-technical users who lack programming knowledge but still need to retrieve and analyze data. Traditional database systems do not provide an intuitive way for users to interact using natural language. As a result, users often depend on technical experts to generate queries, leading to delays, increased workload, and reduced productivity. Additionally, manual SQL query writing is prone to errors, which can result in incorrect data retrieval or system inefficiencies. Although some tools attempt to simplify database querying, they often lack accuracy, contextual understanding, and adaptability to complex queries. There is a need for a system that can accurately interpret user intent and convert natural language into efficient SQL queries. Therefore, the problem addressed in this project is to design and develop an intelligent system that enables users to interact with databases using natural language, automatically translating user inputs into accurate SQL queries, while improving accessibility, reducing errors, and enhancing overall efficiency.

III. LITERATURE SURVEY

The concept of converting natural language into SQL queries has been an active area of research in the fields of Artificial Intelligence (AI) and Natural Language Processing (NLP). Over the years, several approaches and systems have been developed to simplify database querying for non-technical users. Early systems, known as Natural Language Interfaces to Databases (NLIDB), focused on rule-based and pattern-matching techniques. These systems relied on predefined grammar rules and keyword mappings to convert user input into SQL queries. Although they were effective for simple queries, they lacked flexibility and failed to handle complex or ambiguous user inputs With the advancement of Machine Learning, statistical and supervised learning models were introduced to improve query translation. These approaches used training datasets consisting of natural language questions and their corresponding SQL queries. While they improved accuracy compared to rule-based systems, they still required large labeled datasets and struggled with domain adaptability. Recent developments in deep learning, especially with the introduction of transformer-based models, have significantly enhanced the performance of Natural Language to SQL (NL2SQL) systems. Models like sequence-to-sequence architectures and attention mechanisms enabled better understanding of context and relationships within user queries. Popular datasets such as Spider and WikiSQL have been widely used to train and evaluate these models. Furthermore, the emergence of Large Language Models (LLMs), such as Gemini Pro, GPT-based models, and others, has revolutionized the NL2SQL domain. These models can understand complex natural language queries, generate accurate SQL statements, and adapt to different database schemas with minimal training. They also provide better contextual understanding and reduce the need for extensive manual rule creation. Several modern tools and platforms integrate LLMs with database systems to provide intelligent querying capabilities. However, challenges such as ambiguity in user input, query optimization, handling complex joins, and ensuring security still remain areas of ongoing research. This project builds upon these advancements by leveraging LLM capabilities to develop an efficient and user-friendly system, IntelliSQL, that enables seamless interaction between users and databases through natural language.

3.1 Proposed Solution

The proposed system:

- Accepts natural language query
- Uses NLP model to analyze sentence
- Identifies table name, conditions, fields
- Generates optimized SQL query
- Executes query
- Shows output

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

4.3 Solution Architecture

UserInterface
↓
NLPProcessingEngine
↓
QueryGeneratorModule
↓
SQLExecutionEngine
↓
Database
↓
Results Display

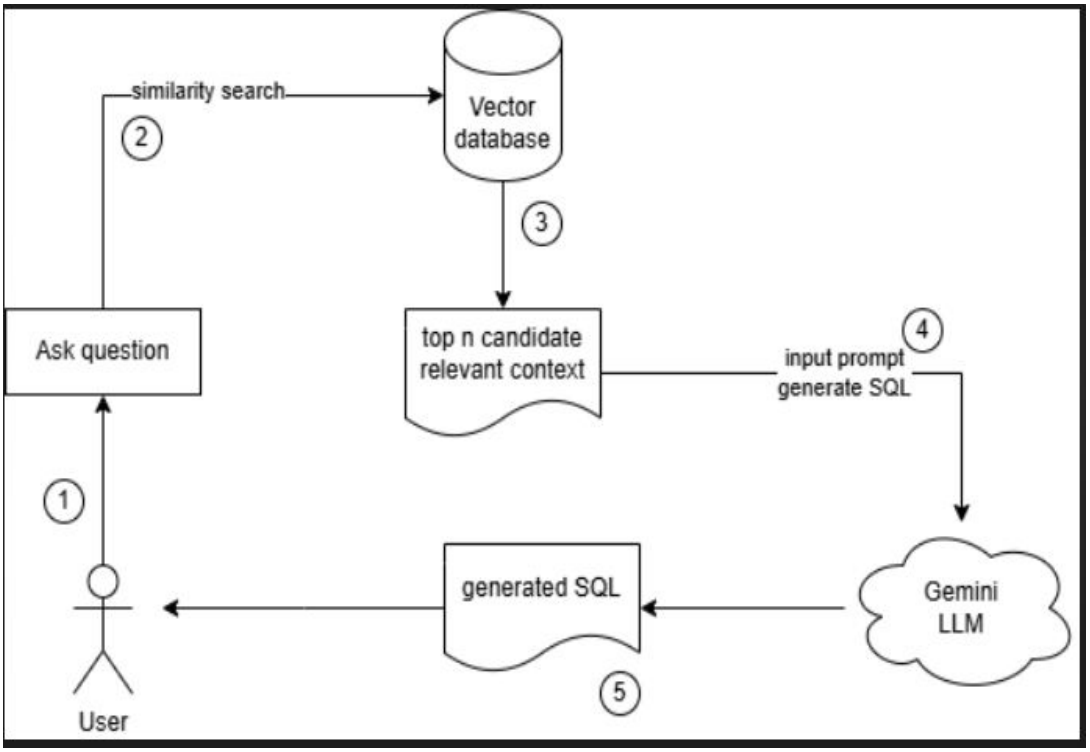


Fig: Proposed System Architecture

- The user asks a question in natural language, which is processed through a **similarity search in the vector database** to find relevant context. This context is then used to create a prompt and sent to the **Gemini LLM**, which generates the corresponding SQL query. Finally, the generated SQL is returned to the user for execution and results.

IV. PROPOSED METHODOLOGY

The proposed methodology of IntelliSQL focuses on designing an intelligent system that converts natural language queries into SQL queries using Natural Language Processing (NLP) and Large Language Models (LLMs). The system follows a step-by-step approach to ensure accurate query generation and efficient data retrieval.

1. Data Collection and Database Setup:

Create or use an existing relational database (e.g., MySQL/PostgreSQL).
Define tables, attributes, and relationships clearly.
Ensure the database schema is well-structured, as it plays a key role in query generation.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

2. User Query Input:
The user enters a query in natural language through a web-based interface.
Example: "Display all students who scored above 80 marks."
3. Preprocessing of Input:
The input query is cleaned and prepared using NLP techniques:
Tokenization
Stop-word removal
Lowercasing
This step improves the understanding of user intent.
4. Natural Language Understanding (NLU):
The processed input is passed to an LLM (e.g., Gemini Pro).
The model:
Understands user intent
Identifies entities (tables, columns, conditions)
Interprets relationships between data
5. Schema Mapping:
The system maps extracted entities from the user query to the database schema.
Example:
"students" → table name
"marks" → column name
This ensures accurate query formation.
6. SQL Query Generation:
Based on the interpreted input, the system generates a valid SQL query.
Example:
SQL
SELECT * FROM students WHERE marks > 80;
The system ensures proper syntax and logical correctness.
7. Query Validation and Optimization:
Validate the generated SQL query to avoid errors.
Apply optimization techniques (e.g., removing redundant operations) to improve performance.
8. Query Execution:
The validated SQL query is executed on the database.
The database processes the query and returns results.
9. Result Processing and Visualization:
The output is formatted into a user-friendly structure such as tables or charts.
Enhances readability and understanding.
10. Result Display:
The final results are displayed on the frontend interface.
Users can easily interpret the data without technical knowledge.

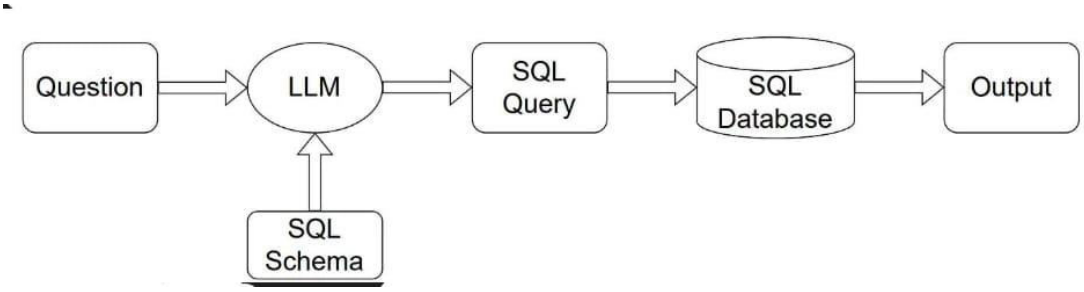


Fig: LLM-Based Natural Language to SQL Query Architecture

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

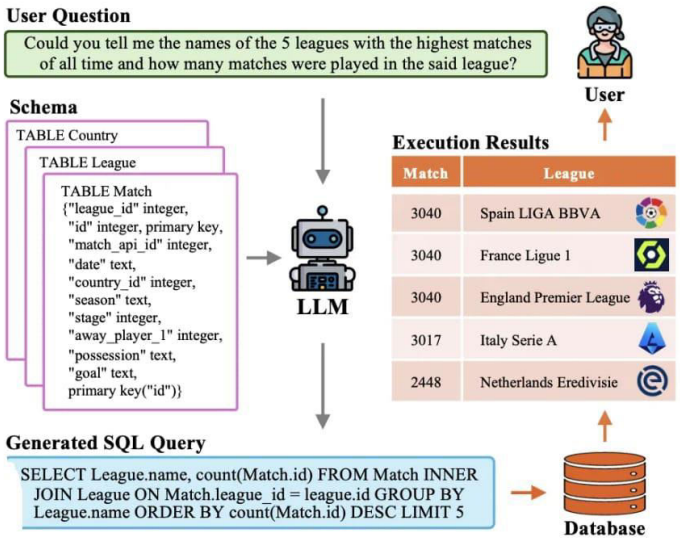


Fig: End-to-End Workflow of Intelligent SQL Query Generation using LLM

V. EXPERIMENTAL RESULTS

1. Create Table:

```
CREATE TABLE employees (
   id INT PRIMARY KEY,
   name VARCHAR(50),
   department VARCHAR(50),
   salary INT,
   join_date DATE
);
```

2. Insert Data:

```
INSERT INTO employees (id, name, department, salary, join_date) VALUES
(1, 'Fayaz', 'IT', 50000, '2022-01-10'),
(2, 'Anand', 'IT', 65000, '2021-03-15'),
(3, 'Karthik', 'IT', 70000, '2019-11-05'),
(4, 'Pavani', 'IT', 55000, '2023-06-20');
```

Input:

Show employees in IT department

Generated SQL:

```
SELECT * FROM employees WHERE department = 'IT';
```

Output:

id	name	department	salary	join_date
1	Fayaz	IT	50000	2022-01-10
2	Anand	IT	65000	2021-03-15
3	Karthik	IT	70000	2019-11-05
4	Pavani	IT	55000	2023-06-20

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

1.Query
Input:
Show employees with salary greater than 65000
SQL:
SELECT * FROM employees WHERE salary > 65000;
Output:

id	name	department	salary
2	Anand	IT	65000
3	Karthik	IT	70000

2. Query:
Input:
Show employees who joined after 2022
SQL:
SELECT * FROM employees WHERE join_date > '2022-01-01';
Output:

id	name	join_date
1	Fayaz	2022-01-10
4	Pavani	2023-06-20

3. Query
Input:
Count employees in each department
SQL:
SELECT department, COUNT(*) FROM employees GROUP BY department;
Output:

department	count
IT	4

4.Query
Input:
Show top 2 highest paid employees
SQL:
SELECT * FROM employees ORDER BY salary DESC LIMIT 2;
Output:

name	salary
Karthik	70000
Anand	65000

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

VI. ALGORITHM

Algorithm: IntelliSQL (Natural Language to SQL Query System)
Input:
User's natural language query (in English)
Output:
Correct SQL query
Query result from the database
Step-by-Step Algorithm
Start
User Input
Accept natural language query from the user
(**Example:** "Show all employees with salary greater than 50,000")
Preprocessing
Convert text to lowercase
Remove unnecessary symbols/punctuation
Perform tokenization (split into words)
Intent Recognition
Identify the type of operation:
SELECT / INSERT / UPDATE / DELETE
Detect keywords like:
"show", "list" → SELECT
"add", "insert" → INSERT
Entity Extraction
Identify:
Table name (e.g., employees)
Columns (e.g., salary, name)
Conditions (e.g., > 50000)
Schema Mapping
Match extracted entities with database schema
Validate table names and column names
Query Generation using LLM
Send processed input + schema to LLM (Gemini Pro)
Generate SQL query
Example:
SELECT * FROM employees WHERE salary > 50000;
Query Validation
Check syntax correctness
Prevent SQL injection or invalid queries
Execution
Execute SQL query on the database
Result Retrieval
Fetch results from database
Response Generation
Convert result into user-friendly format (table/text)
Display Output
Show results to user
End

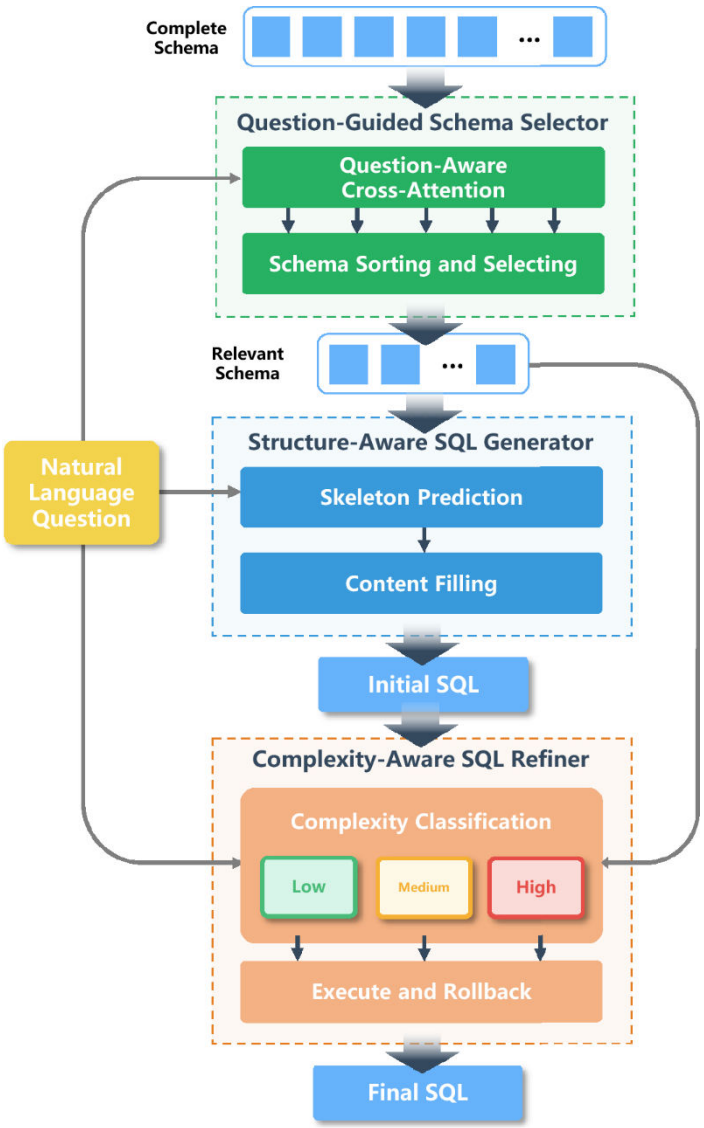
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|



VII. DISCUSSION

The IntelliSQL system demonstrates how natural language processing combined with large language models can simplify database interactions for non-technical users. Traditional SQL querying requires knowledge of syntax, database schema, and logical structuring, which can be a barrier for many users. This project addresses that limitation by allowing users to interact with databases using simple natural language. One of the key observations from this project is that integrating an LLM (such as Gemini Pro) significantly improves the accuracy of SQL query generation. The model is capable of understanding user intent, identifying relevant entities, and generating syntactically correct queries. This reduces the need for manual query writing and minimizes human errors. During implementation, it was observed that schema awareness plays a crucial role in improving results. When the system is provided with database schema details (table names, column names, relationships), the generated SQL queries are more precise and contextually relevant. Without schema guidance, the model may produce ambiguous or incorrect queries. Another important aspect is query validation and security. The system must ensure that generated queries are safe to execute. Mechanisms such as syntax checking and restricting harmful operations (e.g., DROP or DELETE without conditions) are necessary to prevent unintended data loss or security risks. The experimental results show that the system performs well for simple to moderately complex queries, such as filtering, sorting, and basic joins. However, performance may decrease for highly complex queries involving nested subqueries or multiple joins. This highlights an area for further

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

improvement. Additionally, the system improves user experience and productivity by reducing the time required to retrieve information from databases. Users without SQL knowledge can easily access and analyze data, making the system useful in domains like education, business analytics, and administration.

VIII. LIMITATIONS

The system heavily depends on the accuracy of the Large Language Model (LLM); incorrect interpretation can lead to wrong SQL queries. It may struggle with ambiguous or unclear user inputs, resulting in incorrect or incomplete results. Performance decreases when handling complex queries such as nested queries, subqueries, and multiple joins. Requires proper database schema information; without it, the system may generate invalid queries. The system relies on internet connectivity/API access for LLM integration, which may affect availability. There is a risk of generating unsafe queries (e.g., DELETE or DROP) if proper validation is not implemented. Limited ability to handle domain-specific terminology unless trained or fine-tuned accordingly. May not always optimize queries for performance, leading to slower execution on large datasets. Handling of multi-language queries is limited if the model is not properly configured. Requires additional effort for security measures like preventing SQL injection attacks. Debugging incorrect queries can be difficult due to lack of transparency in LLM decision-making.

IX. CONCLUSION

The IntelliSQL project successfully demonstrates how natural language processing and large language models can be used to simplify database querying. By allowing users to interact with databases using plain English, the system removes the need for deep knowledge of SQL syntax and makes data access more user-friendly. The integration of LLMs enables accurate understanding of user intent and efficient generation of SQL queries. This improves productivity, reduces errors, and saves time compared to traditional query-writing methods. The system performs effectively for simple and moderately complex queries, proving its practical usability. However, the project also highlights certain challenges such as handling ambiguous queries, dependence on model accuracy, and limitations with complex database operations. These areas provide opportunities for further enhancement. Overall, IntelliSQL presents a promising approach toward intelligent database systems. With future improvements in accuracy, security, and performance, it has the potential to become a powerful tool for both technical and non-technical users across various domains.

X. ACKNOWLEDGMENT

We would like to express our sincere gratitude to our project guide Mrs. SIVA TEJASWINI for providing valuable guidance, support, and encouragement throughout the development of this research work. His expert suggestions and continuous support helped us successfully complete this project.

We also thank the management and faculty members of Kallam Haranadha Reddy Institute of Technology, Department of Data Sceince, for providing the necessary facilities, computational resources, and academic environment to carry out this research work.

Finally, we extend our appreciation to all those who directly or indirectly contributed to the completion of this study.

REFERENCES

1. Google AI, *Gemini Pro: Advanced Multimodal AI Model*, Google DeepMind, 2024.
2. Brown, T. B., et al., *Language Models are Few-Shot Learners*, NeurIPS, 2020.
3. OpenAI, *GPT-4 Technical Report*, OpenAI, 2023.
4. Rajkumar, P., et al., *Evaluating the Text-to-SQL Capabilities of Large Language Models*, arXiv, 2022.
5. Zhong, V., et al., *Seq2SQL: Generating Structured Queries from Natural Language*, arXiv, 2017.
6. Yu, T., et al., *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task*, EMNLP, 2018.
7. Google Cloud, *BigQuery Documentation*, cloud.google.com/bigquery
8. MySQL Documentation, *MySQL Reference Manual*, dev.mysql.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.699

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 4, April 2026

|DOI: 10.15680/IJIRSET.2026.1504100|

9. SQLite Documentation, *SQLite Database Engine*, sqlite.org
10. Pandas Documentation, *Python Data Analysis Library*, pandas.pydata.org
11. NumPy Documentation, numpy.org
12. Scikit-learn Documentation, scikit-learn.org
13. Hugging Face, *Transformers Library Documentation*, huggingface.co/docs
14. LangChain Documentation, *Building Applications with LLMs*, langchain.com
15. Flask Documentation, *Flask Web Framework*, flask.palletsprojects.com
16. FastAPI Documentation, fastapi.tiangolo.com
17. XGBoost Documentation, xgboost.readthedocs.io
18. World Bank, *Data-Driven Decision Making and Analytics*, worldbank.org
19. Kaggle, *Text-to-SQL and Database Query Datasets*, kaggle.com
20. Chen, M., et al., *Evaluating Large Language Models Trained on Code*, arXiv, 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details