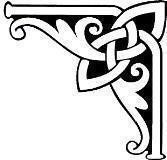
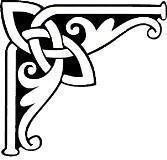
# VISVESVARAYA TECHNOLOGICAL UNIVERSITY JNANASANGAMA, BELAGAVI-590 018, KARNATAKA



## PROJECT PHASE - I REPORT

ON

## “GENERATIVE AI POWERED SYSTEM FOR CONVERTING TEXT TO SQL”

*Submitted in the partial fulfillment of requirements for the award of Degree*

**B.E. in Computer Science and Engineering**

**Submitted by**

**Adesh C Sahukar 4BD22CS004**

**Hamsa H R 4BD22CS055**

**Sahana J 4BD22CS136**

**Sanket S Gowdar 4BD22CS145**

**Submitted to**

**PROJECT GUIDE**

**Prof. Preethi B(Ph.D.) Assistant professor,**

**Dept. of CS&E**

**Bapuji Institute of Engineering and Technology Department of Computer Science and Engineering**

**Davanagere-577004 2024-2025**

# Bapuji Institute of Engineering and Technology Davangere – 577004

**Department of Computer Science and Engineering CERTIFICATE**

This is to certify that **Adesh C Sahukar, Hamsa H R, Sahana J, Sanket S Gowdar** bearing USN **4BD22CS004, 4BD22CS055, 4BD22CS136, 4BD22CS145** respectively of **Computer Science and Engineering** department have satisfactorily submitted the Final Year Project Phase - I report entitled **“GENERATIVE AI POWERED SYSTEM FOR CONVERTING TEXT TO SQL”** in the partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering (B.E.) in Computer Science and Engineering, under the VTU during the academic year 2024-25.

### Prof. Preethi B(Ph.D.) Assistant Professor Dept. of CS&E

**Guide**

**Prof. Rachana G Sunkad M. Tech.**

### Assistant Professor Dept. of CS&E Project Co-ordinator

**Dr. Nirmala C R Ph. D.**

### Head of Department CS & E

**Dr. H B Aravind Ph.D.**

**Principal**

**Date: 16-05-2025**

**Place: Davanagere**

**ACKNOWLEDGEMENT**

Salutations to our beloved and highly esteemed institute, **“BAPUJI INSTITUTE OF ENGINEERING AND TECHNOLOGY”** for having well-qualified staff and labs furnished with the necessary equipment.

We express our sincere thanks to our guide **Prof. Preethi B**, Assistant Professor, Department of Computer Science and Engineering, B.I.E.T., Davangere, for giving us the constant encouragement, support and valuable guidance throughout the course of the major project without, whose stable guidance this project would not been achieved.

We are grateful to Project Coordinators **Dr. Naresh Patel K M** Associate Professor, Department of Computer Science and Engineering, B.I.E.T and **Prof. Rachana G S,** Assistant Professor, Department of Computer Science and Engineering, B.I.E.T and **Mr. Vinayak P**, Assistant Professor, Department of Computer Science and Engineering, for their constant support and feedback throughout this project.

We express whole hearted gratitude to **Dr. Nirmala C R** who is our respectable H.O.D of Computer Science and Engineering Department. We wish to acknowledge her help who made our task easy by providing with her valuable help and encouragement

We also express our wholehearted gratitude to our respected Principal, **Dr. H B Aravind** for his moral support and encouragement.

We would like to extend our gratitude to all Teaching and Non-Teaching staff of the **Department of Computer Science and Engineering** for the help and support rendered to us. We have benefited a lot from the feedback, and suggestions given by them.

We would like to extend our gratitude to all our family members and friends especially for their advice and moral support.

### Adesh C Sahukar (4BD22CS004) Hamsa H R (4BD22CS055) Sahana J (4BD22CS0136)

**Sanket S Gowdar (4BD22CS145)**

Bapuji Educational Association (Regd.)

**Bapuji Institute of Engineering and Technology, Davangere-577004**

# Vision and Mission of the Computer Science and Engineering Department

## Vision

“To be a centre-of-excellence by imbibing state-of-the-art technology in the field of Computer Science and Engineering, thereby enabling students to excel professionally and be ethical.”

**Mission**

|  |  |
| --- | --- |
| 1. | Adapting best teaching and learning techniques that cultivates Questioning and Reasoning culture among the students. |
| 2. | Creating collaborative learning environment that ignites the critical thinking in students and leading to the innovation. |
| 3. | Establishing Industry Institute relationship to bridge skill gap and make them industry ready and relevant. |
| 4. | Mentoring students to be socially responsible by inculcating ethical and moral values. |

**Program Educational Objectives (PEOs):**

|  |  |
| --- | --- |
| PEO1 | To apply skills acquired in the discipline of computer science and engineering for solving Societal and industrial problems with apt technology intervention. |
| PEO2 | To continue their carrier ion industry /academia or pursue higher studies and research. |
| PEO3 | To become successful entrepreneurs, innovators to design and develop software products and services that meets societal, technical and business challenges. |
| PEO4 | To work in the diversified environment by acquiring leadership qualities with effective communication skills accompanied by professional and ethical values. |

**Program Specific Outcomes (PSOs):**

|  |  |
| --- | --- |
| PSO1 | Analyse and develop solutions for problems that are complex in nature but applying the knowledge acquired from the core subjects of this program. |
| PSO2 | To develop secure, scalable, resilient and distributed applications for industry and societal Requirements. |
| PSO3 | To learn and apply the concepts and contract of emerging technologies like artificial intelligence, machine learning, deep learning, big-data analytics, IOT, cloud computing etc for any real time problems. |

# CONTENTS

## CONTENTS PAGE NO.

Chapter 1: Introduction 1

* 1. Introduction 1

Chapter 2: Literature Survey 2-6

* 1. [Literature review summary 4](#_TOC_250007)
  2. [Literature gaps 4](#_TOC_250006)
  3. [Existing system 5](#_TOC_250005)
  4. [Problem statement 5](#_TOC_250004)
  5. [Proposed system 6](#_TOC_250003)
  6. [Objectives 6](#_TOC_250002)

Chapter 3: System Requirements and Specification 7-8

* 1. System requirements and specification 7-8

Chapter 4: Methodology 9-10

* 1. Methodology 9-10

[Conclusion 11](#_TOC_250001)

[References 12](#_TOC_250000)

**List of Figures**

**Sl. No. Figure No. Description Page No.**

1. 1 Architecture Diagram 8

**ABSTRACT**

AI-powered Text-to-SQL System Using Generative AI project explores the development of a system that converts natural language queries into SQL commands using Generative AI. The objective is to bridge the gap between non-technical users and structured databases by enabling intuitive, conversational interactions. Leveraging large language models (LLMs), the system interprets user input, understands the underlying database schema, and generates accurate SQL queries. The project focuses on schema adaptability, user-friendly design, and query accuracy. By automating SQL generation, this approach enhances data accessibility and simplifies database interaction for users without SQL knowledge. The solution has practical applications in business analytics, education, and customer service.

# Chapter 1

**Introduction**

The ability to interact with databases through natural language has become not only desirable but essential. As organizations across industries collect and store vast volumes of structured data, the need for accessible and user-friendly data interaction methods has grown significantly. Traditional query languages like SQL, while powerful and expressive, often require specialized knowledge and training, creating a barrier for non-technical users who still need to retrieve insights and make data-driven decisions. The development of Natural Language Interfaces to Databases (NLIDBs) - systems designed to interpret user intents expressed in plain language and translate them into precise, executable database queries.

At the forefront of this evolution are Generative AI models, particularly large language models (LLMs) such as GPT which have revolutionized the way machines understand and generate human language. These models, trained on vast corpora of text, exhibit a remarkable ability to comprehend semantic nuances, generate coherent responses, and produce structured outputs from unstructured inputs. Their application in Text-to-SQL systems—where a natural language input is automatically converted into a structured SQL query—has shown promising results. Practical tools like AI2SQL, Text2SQL Bench, and Spider exemplify how LLMs can empower users to interact with databases using everyday language, thereby democratizing data access and reducing reliance on technical personnel for routine query tasks.

However, despite these advancements, several challenges persist. Many current systems struggle with interpreting complex queries that involve nested conditions, joins across multiple tables, or domain-specific language. The issue of semantic ambiguity in natural language remains a hurdle, often leading to incorrect or suboptimal SQL generation. Ensuring that the generated queries are not only correct but also interpretable, context-aware, and secure is an ongoing area of focus.

This report delves into the current landscape of Text-to-SQL technologies, providing a comprehensive survey of existing systems, methodologies, and benchmarks. It identifies the strengths and limitations of various approaches and highlights emerging trends. Furthermore, it proposes a refined methodology aimed at enhancing both the performance and usability of natural language to SQL conversion tools. The goal is to contribute toward the development of intelligent, user-centric interfaces that bridge the gap between human language and structured data access, ultimately enabling broader and more inclusive use of organizational data assets.

# Chapter 2

**Literature Survey**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **Title** | **Authors** | **Year** | **Contribution** | **Techniques**  **Used** |
| 1. | ChatGPT for Text-to-SQL: Opportunities and Limitations | OpenAI Community (Experiments) | 2023 | Demonstrates how LLMs like ChatGPT can be prompted to generate SQL  queries. | Prompt engineering, zero-shot/few- shot learning |
| 2. | Natural SQL: Towards Natural Language Interaction  with Databases | Microsoft Research | 2022 | Focuses on real- world application and interaction with SQL-based  systems. | Fine-tuned large language models (LLMs) |
| 3. | PICARD:  Executing SQL During Decoding Improves Accuracy | Wolfgang Gatterbauer et al. | 2021 | Enhances decoding process by executing partially decoded SQL for validation. | Execution- guided decoding, autoregressive models |
| 4. | Text-to-SQL in the Wild: A Naturally- Occurring Dataset Based on Stack Exchange Data | Chenglong Wang et al. | 2021 | Provides a large-scale dataset for training models on real-world text-to-SQL cases. | Data-driven, neural semantic parsing |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5. | T5: Exploring the Limits of Transfer Learning with a Unified Text- to-Text  Transformer | Colin Raffel et al. | 2020 | General- purpose model used for text-to- SQL via fine- tuning. | Pretrained transformer (T5), text-to- text paradigm |
| 6. | TAPAS:  Weakly Supervised Table Parsing via Pretrained Language  Models | Jonathan Herzig et al. | 2020 | Converts natural language to tabular data manipulation queries. | BERT-style model for tabular QA |
| 7. | SmBoP: Semi- Autoregressive Bottom-Up Semantic  Parsing | Yuwei Lin et al. | 2020 | A semantic parser that improves compositional  generalization. | Bottom-up tree structure generation |
| 8. | RAT-SQL:  Relation- Aware Schema Encoding and Linking for Text-to-SQL  Parsers | Ben Bogin et al. | 2019 | Handles complex schemas and cross-domain tasks efficiently. | Relation- aware transformer (RAT),  schema linking |
| 9. | Spider: A Large-Scale Human- Labeled Text- to-SQL Dataset | Tao Yu et al. | 2018 | Benchmark dataset for evaluating cross-domain generalization in text-to-SQL  systems. | Multi-domain dataset, human- annotated |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10. | SQLNet:  Generating Structured Queries Without Reinforcement Learning | Xiaojun Xu et al. | 2017 | Improved accuracy over Seq2SQL  without using reinforcement learning. | Sketch-based approach, column attention |

**Table 1:** Literature Survey Table

## Literature review summary

Recent years have seen significant progress in the field of Text-to-SQL translation, driven largely by advances in neural networks and the emergence of large-scale pretrained models. Early models like SQLNet and Seq2SQL introduced neural semantic parsing approaches to map natural language to SQL queries without heavy reliance on hand-crafted rules. With the introduction of datasets like Spider and Text-to-SQL in the Wild, researchers gained access to diverse and complex training data, improving model generalization. Transformer-based models like T5 and RAT-SQL expanded on these foundations by incorporating schema encoding, relation- awareness, and multi-turn interactions. Execution-guided decoding methods such as PICARD further enhanced accuracy by validating partial SQL outputs during generation. Most recently, LLMs such as ChatGPT have shown the ability to generate SQL with minimal training through prompt engineering, underscoring the versatility and scalability of foundation models. Together, these works highlight the shift toward more intelligent, flexible, and context-aware Text-to-SQL systems.

## Literature gaps:

* + - **Lack of robust UI integration and result visualization**, which are crucial for usability and user trust.
    - **Focus is mostly on query generation and accuracy**, with limited work on the end-to- end user experience.
    - **Limited adoption of execution-guided mechanisms**, like those introduced by PICARD, for real-time validation.
    - **Inadequate handling of multi-turn, context-aware interactions**, which are essential for conversational use cases.

## Existing System

Over the past few years, significant advancements have been made in the development of text-to-SQL systems through various natural language processing and machine learning techniques. OpenAI’s 2023 exploration with **ChatGPT for Text-to-SQL** showcased the potential of large language models (LLMs) to generate SQL queries using prompt engineering and zero-shot or few-shot learning techniques, eliminating the need for extensive task-specific training.

Recent developments in natural language processing and large language models (LLMs) have significantly improved text-to-SQL systems. Similarly, Microsoft Research’s Natural SQL (2022) leveraged fine-tuned LLMs to create more intuitive and user-friendly interactions with databases, making SQL access more accessible to non-technical users.

Advancements in model architectures and decoding strategies have also contributed to higher accuracy and robustness. PICARD (2021) improved SQL generation by executing queries during decoding to catch errors early, while Text-to-SQL in the Wild (2021) introduced a large- scale, real-world dataset from Stack Exchange to train models on naturally occurring queries. Transformer-based models like T5 and TAPAS extended the use of pretrained language models for SQL and tabular data tasks, and SmBoP (2020) introduced a tree-based parsing structure to improve compositional understanding in SQL generation.

Earlier foundational systems laid the groundwork for today’s models. RAT-SQL (2019) introduced schema-aware transformers for handling complex queries across multiple domains. The Spider dataset (2018) became a key benchmark for evaluating cross-domain generalization in text-to-SQL systems. SQLNet (2017) further simplified query generation by eliminating reinforcement learning and using a sketch-based method with column attention. These systems collectively highlight the evolution of text-to-SQL technology from rule-based systems to highly adaptive and generalizable AI models.

## Problem Statement

Text-to-SQL systems face several challenges in accurately converting natural language queries into SQL. They struggle with handling complex queries, ambiguous language, and domain-specific terminology. These systems often lack interactive interfaces, visual query representation, and real-time feedback, limiting usability. Despite generating syntactically correct SQL, they frequently fail to capture the intended meaning due to limited contextual understanding. Additionally, weak support for dynamic schema linking and cross-domain generalization reduces their adaptability and reliability, especially in enterprise environments.

## Proposed System

The proposed system is an enhanced AI-driven Text-to-SQL converter built upon current advancements in natural language processing and database interfacing. It integrates a powerful language model fine-tuned specifically on SQL-related datasets and augmented with schema-awareness, contextual embeddings, and real-time query validation. Unlike generic prompt-based systems, this tool emphasizes domain-specific adaptability, advanced error correction, and support for complex queries involving nested conditions and joins. It also introduces an interactive interface where users can iteratively refine their questions and preview SQL results, improving both transparency and user confidence.

## Objectives

* + - To develop a schema linking and contextual understanding.
    - To develop SQL query generation using LLM.
    - To generate output rendering and explanation.
    - To design an interactive interface and visualization

# Chapter 3

**System Requirements and Specification**

## System Overview

* + - * The GenAI-based system will:
        + Accept natural language queries from users.
        + Generate and optimize the corresponding SQL query.
        + Execute the query on a connected database.
        + Return results or visualizations to the user.

## Functional Requirements

### Natural Language Query Interpretation

Understand and translate user queries into valid SQL statements.

### SQL Optimization

Ensure the generated queries are optimized for performance.

### Real-Time Validation

Detect syntax or semantic errors before execution.

### Support for Complex SQL Features

Handle joins, subqueries, GROUP BY, HAVING, and nested conditions.

### Multi-Database Support

Integrate with different types of databases (e.g., PostgreSQL, MySQL, Snowflake).

### User Interface

Provide a user-friendly interface for input, result display, and query history.

## Technical Requirements

### Programming Language:

Python (preferred for AI integration and backend)

### Generative AI Model:

Integration with a language model (e.g., OpenAI GPT, LLaMA, or similar)

### Hardware Requirements:

* + - * + Minimum: 8 GB RAM, multi-core CPU
        + Recommended: 16+ GB RAM, GPU for local model inference (optional)
        + Intel i5 or higher processor, 256GB storage.

### Software Requirements:

* + - * + Frontend: ReactJS or similar framework
        + Backend: Python (Flask or FastAPI)
        + NLP Engine: OpenAI GPT / Hugging Face Transformers
        + Database: PostgreSQL / MySQL
        + Libraries: Transformers, spaCy, SQLAlchemy, NLTK

TRAIN ASK

DDL, Documentation, Reference SQL Queries

Question

Generate Embedding

Generate Embedding

Store

embedding

and Metadata

Any vector database

Find related DDL, documentation, and reference SQL

Construct prompt and send to LLM

SQL

**Fig 1:** Architecture diagram

# Chapter 4

**Methodology**

The proposed Text-to-SQL system follows a modular, multi-phase pipeline designed to handle natural language inputs and convert them into accurate SQL queries. Each phase contributes to understanding, generating, and validating structured queries efficiently:

1. Natural Language Processing (NLP) and Preprocessing
   * The system first processes the user's input using standard NLP techniques such as tokenization, part-of-speech tagging, named entity recognition, and syntactic parsing.
   * This step helps identify entities (like column names or table names), intents (like SELECT, INSERT), and conditions (like WHERE clauses) from the natural language query.
2. Schema Linking and Contextual Understanding
   * The system loads the schema of the target database and maps the recognized entities from the input to actual columns, tables, or metadata in the database.
   * Relation-aware models or schema encoders (e.g., RAT-SQL style) are used to match the user query with relevant parts of the schema using attention mechanisms.
3. SQL Query Generation using LLM
   * A fine-tuned large language model (e.g., T5, Codex, or GPT-style model) is used to generate a SQL query based on the interpreted natural language and schema context.
   * The model is trained on various SQL datasets (like Spider or Text2SQL-in-the- Wild) to learn common query structures, logical forms, and syntax.
4. Intermediate Query Validation and Correction
   * The generated SQL query undergoes a validation phase where a SQL parser checks for syntax and semantic correctness.
   * Execution-guided decoding can be applied (similar to PICARD), where parts of the query are executed against the database to validate subqueries or catch execution errors early.
5. Query Optimization
   * The query is reviewed for optimization—such as adding indexes, simplifying joins, or reducing nested SELECTs—using heuristic or learned techniques.
   * In systems like AI2SQL, this can also involve suggesting improved query formulations.
6. Output Rendering
   * The final SQL query is displayed to the user, ensuring transparency and trust.
   * If enabled, the system may provide a summary of the results or allow the user to preview query output, depending on database connectivity.

# Conclusion

The integration of generative AI in Text-to-SQL systems, as exemplified by platforms like AI2SQL, demonstrates a significant step toward making database interaction more accessible and intuitive. By translating natural language into executable SQL queries, these systems reduce the technical barrier for data access and promote data-driven decision-making across industries. While current models show high potential, continuous improvements are needed in handling complex schemas, query optimization, and domain adaptability. Overall, AI- powered Text-to-SQL systems pave the way for more intelligent, user-centric database interfaces.

# References

* Tao Yu et al. (2018). Spider: A Large-Scale Human-Labeled Text-to-SQL Dataset.
* Colin Raffel et al. (2020). T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.
* Ben Bogin et al. (2019). RAT-SQL: Relation-Aware Schema Encoding for Text-to- SQL Parsers.
* Wolfgang Gatterbauer et al. (2021). PICARD: Executing SQL During Decoding Improves Accuracy.
* Microsoft Research (2022). Natural SQL: Towards Natural Language Interaction with Databases.
* OpenAI (2023). ChatGPT for Text-to-SQL: Opportunities and Limitations.