DATABASE QUERY AND REASONING

A PROJECT REPORT

Submitted by,

Mr. YALAMANDALA GOWTHAM NAIDU - 20211CSE0104
Mr. PATAN FAIROZ KHAN -20211CSE0106
Mr. KOLLAGIREDDY SASIVARAN REDDY -20211CSE0109
Mr. KODITALA ABHINAY -20211CSE0069

Under the guidance of,

Ms. V. KAYALVIZHI

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AT

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2025

DATABASE QUERY AND REASONING

A PROJECT REPORT

Submitted by,

Mr. YALAMANDALA GOWTHAM NAIDU - 20211CSE0104
Mr. PATAN FAIROZ KHAN -20211CSE0106
Mr. KOLLAGIREDDY SASIVARAN REDDY -20211CSE0109
Mr. KODITALA ABHINAY -20211CSE0069

Under the guidance of,

Ms. V. KAYALVIZHI

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AT

PRESIDENCY UNIVERSITY

BENGALURU

JANUARY 2025

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **"DATABASE QUERY AND REASONING"** being submitted by **"YALAMANDALA GOWTHAM NAIDU, KODITALA ABHINAY, PATAN FAIROZ KHAN AND KOLLAGIREDDY SASIVARAN REDDY"** bearing roll number(s) **"20211CSE0104, 20211CSE0069, 20211CSE0106, and 20211CSE0109"** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

Ms. V. KAYALVIZHI
Assistant Professor
School of CSE&IS
Presidency University

Dr. ASIF MOHAMMAD
Associate Professor & HoD
School of CSE&IS
Presidency University

Dr. L. SHAKKEERA
Associate Dean
School of CSE
Presidency University

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-VC School of Engineering
Dean -School of CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **DATABASE QUERY AND REASONING** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Engineering**, is a record of our own investigations carried under the guidance of **Ms. V. KAYAL VIZHI, Assistant professor, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Gowtham Naidu

YALAMANDALA GOWTHAM NAIDU - 20211CSE0104

K. Abhinay

KODITALA ABHINAY - 20211CSE0069

P. Fairoz Khan

PATAN FAIROZ KHAN - 20211CSE0106

Sasi Varan.

KOLLAGIREDDY SASIVARAN REDDY - 20211CSE0109

ABSTRACT

The rapid evolution of data-driven decision-making has necessitated the development of systems capable of bridging the gap between natural human language and structured query languages like SQL. Traditional database systems require users to have a technical understanding of SQL syntax and database schemas, which can be a significant barrier for non-technical users. This project, titled **"Database Query and Reasoning"** aims to address this challenge by building an intelligent system that can interpret natural language input, convert it into SQL queries, and provide both results and reasoning for the query output.

At its core, the system integrates multiple technologies to offer a seamless experience. A web-based interface, developed using Streamlit, allows users to upload datasets, view database schemas, and input queries in natural language. The system processes this input using advanced natural language processing (NLP) techniques powered by Google's Gemini AI model. By leveraging Gemini's generative capabilities, the system translates user queries into accurate SQL statements tailored to the schema of the uploaded dataset. This capability democratizes access to structured data, enabling users with minimal technical expertise to interact effectively with databases.

A critical aspect of this project is reasoning. Beyond fetching and

displaying query results, the system provides detailed explanations of how the SQL query was derived and why specific results were returned. This reasoning functionality, also powered by Gemini AI, enhances the system's transparency and usability. It explains the transformation process from natural language to SQL, offering insights into the query structure and the logic behind the retrieved results. This feature not only builds user trust but also serves as an educational tool for users aiming to learn SQL.

The project also includes robust backend functionality to manage and process data. Users can upload CSV files, which the system automatically converts into SQLite databases. The schema of the uploaded data is extracted and displayed, ensuring users have a clear understanding of the underlying structure. This schema awareness is crucial for accurate query generation, as it allows the system to infer column names and data types relevant to user questions.

From a technical perspective, the system integrates Python libraries like Pandas for data handling, SQLite for database management, and Transformers for leveraging pre-trained models.

The use of Google's Gemini model is particularly significant, as it provides cutting-edge generative AI capabilities to process complex natural language queries and deliver precise SQL outputs. The design prioritizes scalability and flexibility, ensuring that the system can handle diverse datasets and adapt to varying user requirements.

The potential applications of this system are vast. Businesses can leverage it to empower employees with limited technical expertise to extract insights from organizational data. Educational institutions can use it as a teaching tool to introduce students to SQL and database management concepts. Researchers and analysts can benefit from the system's ability to quickly generate insights from data without the need for manual query writing.

Throughout the development process, significant challenges were encountered and addressed. Translating ambiguous or contextually complex natural language queries into SQL required careful prompt engineering and iterative testing with the Gemini AI model. Handling diverse datasets with varying structures posed another challenge, necessitating robust error handling and schema validation mechanisms. Additionally, ensuring the system's reasoning outputs were both accurate and comprehensible required extensive fine-tuning of the reasoning prompts.

In conclusion, this project represents a significant step toward making database querying more accessible and transparent. By combining cutting-edge AI with user-friendly design, the system bridges the gap between natural language and SQL, empowering users to interact with data intuitively and confidently. The reasoning functionality adds a unique dimension, fostering greater understanding and trust in AI-

generated outputs. This project demonstrates the transformative potential of AI in democratizing access to data and enhancing decision-making processes across diverse domains.

ACKNOWLEDGEMENT

First of all, we indebted to the GOD ALMIGHTY for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mydhili Nair**, School of Computer Science Engineering & Information Science, Presidency University, and **Dr. Asif Mohammed H.B**, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Ms. V. Kayal Vizhi**, Assistant professor and Reviewer **Ms. Sreelatha P**, Assistant professor, School of Computer Science Engineering & Information Science, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP2001 Capstone Project Coordinators **Dr. Sampath A K, Dr. Abdul Khadar A** and **Mr. Md Zia Ur Rahman**, Department Project Coordinators **Mr. Amarnath J L** and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

YALAMANDALA GOWTHAM NAIDU

PATAN FAIROZ KHAN

KOLLAGIREDDY SASIVARAN REDDY

KODITALA ABHINAY

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Overview

The exponential growth of financial data has created both opportunities and challenges for businesses, investors, and individuals. Making informed financial decisions requires the ability to query and analyze vast amounts of data quickly and effectively. However, many existing systems require users to have technical expertise in SQL or other query languages, making them inaccessible to non-technical users.

This project, Database Query and Reasoning for Financial QA, bridges the gap between technical financial data systems and user-friendly interfaces. It allows users to ask financial questions in natural language, which the system processes to generate SQL queries, retrieve relevant data, and apply reasoning techniques to provide insights.

The project's scope includes developing an intelligent, automated system that supports querying, reasoning, and decision-making by leveraging modern technologies such as Natural Language Processing (NLP), machine learning, and database systems.

1.2 Problem Statement

Many existing financial systems require users to:

1. Possess knowledge of database query languages like SQL.
2. Manually analyze retrieved data to extract insights.
3. Use separate tools for querying, reasoning, and visualization, leading to inefficiency and complexity.

Such barriers hinder effective financial data interaction, especially for non-technical users, investors, and business analysts. There is a need for a system that simplifies financial data access, analysis, and reasoning through natural language interfaces.

1.3 Scope of the project

1.3.1 Query Generation:

Converting user questions into structured SQL queries

Example: *"What is my total expense in November?"* → SQL query to calculate total expenses for the month.

Financial institutions manage vast amounts of data across various domains, including

transactions, market trends, and risk assessments. Extracting insights from these datasets typically involves writing complex SQL queries involving multiple conditions, joins, and aggregations. Manual query writing is not only time-consuming but also prone to human error, especially when dealing with large and intricate databases.

By implementing a robust query generation system, this project seeks to:

- Democratize Data Access: Enable users from diverse backgrounds to query financial data without requiring SQL expertise.
- Increase Efficiency: Automate the query generation process, reducing the time and effort required to retrieve and analyze data.
- Enhance Decision-Making: Provide quick and accurate responses to complex financial questions, aiding timely and informed decision-making.

In summary, query generation through NLP not only bridges the gap between technical complexity and user accessibility but also enhances the overall efficiency and reliability of financial data analysis. This project's focus on automating this process makes it a valuable innovation in the financial technology landscape.

1.3.2 Reasoning Module:

The reasoning module is a pivotal component of the "Database Query and Reasoning" system, designed to enhance the transparency and user-friendliness of the query generation process. Its primary function is to explain the transformation of a user's natural language query into a corresponding SQL statement and to provide insights into the results retrieved from the database. This module leverages the generative capabilities of Google's Gemini AI model to generate comprehensive, human-readable explanations. The reasoning process begins with the system analyzing the input question, the generated SQL query, and the query results. A carefully engineered prompt is then used to guide the AI model in creating a detailed explanation, covering how the input question was understood, why specific SQL elements were included, and how the returned data aligns with the query. This not only builds user confidence in the system's outputs but also serves as an educational tool, helping users understand the intricacies of SQL and database operations. By providing clarity and fostering trust, the reasoning module transforms the system into a more interactive and insightful data querying tool.

1.3.3 Visualization:

Displaying results in user-friendly formats such as tables, charts, and graphs.

Example: A pie chart showing expense categories for the month.

In the financial industry, data often involves large volumes, complex relationships, and intricate patterns that can be challenging to interpret. Visualization bridges the gap between raw data and actionable knowledge by presenting data in a visual context that enhances comprehension. The significance of this module includes:

- Simplifying Complex Data: Converts complex queries and reasoning outputs into easy-to-understand charts, graphs, and dashboards.
- Enhancing Decision-Making: Provides visual clarity that aids in identifying patterns, trends, and correlations at a glance.
- Facilitating Real-Time Monitoring: Displays real-time data updates, enabling dynamic dashboards for ongoing financial performance tracking.
- Improving User Engagement: Visual tools are more engaging and allow for interactive data exploration, making financial analysis accessible to a broader audience.

1.3.4 Financial Dataset Integration:

Storing and retrieving financial data (e.g., transactions, accounts) in a structured database.

1.3.5 Scalability and Adaptability:

Scalability is essential for ensuring that the system can handle increasing volumes of data, concurrent users, and complex queries as the project evolves. The Financial Database Query and Reasoning System is designed to scale across various dimensions:

a. Data Scalability

- Initial Database (SQLite): Suitable for development and small datasets due to its lightweight nature.
- Transition to Larger Databases (PostgreSQL): For production environments and large datasets, PostgreSQL offers advanced features like indexing, partitioning, and efficient handling of complex queries, making it ideal for handling millions of records.
- Horizontal Scaling: The system can integrate with cloud-based databases (e.g., Amazon RDS, Google Cloud SQL) to distribute data storage and improve read/write performance.

b. User Scalability

- API Rate Limiting: Implementing rate limits ensures that the backend can handle high volumes of simultaneous API requests without degradation.
- Load Balancing: By distributing traffic across multiple servers, the system ensures continuous availability even during peak loads, reducing response times for end users.

c. Query Complexity Scalability

- Optimized SQL Generation: The query generation module is optimized to handle complex multi-join, aggregate, and subquery operations efficiently.
- Asynchronous Processing: For large queries, asynchronous task queues (e.g., Celery with Fast API) prevent bottlenecks and improve user experience by processing long-running queries in the background.

2. Adaptability

The system is designed to be adaptable to changing business needs, new financial regulations, and technological advancements.

a. Modular Architecture

- The system employs a microservices architecture, where each module (query generation, reasoning, visualization) operates independently, allowing for easy updates or replacements without disrupting the entire system.

b. Flexible Data Source Integration

- API-Driven Integration: The system supports multiple data sources, including REST APIs for real-time market data, ensuring seamless integration with third-party financial platforms.
- Pluggable Database Support: The system can easily switch between databases (SQLite, PostgreSQL, or even NoSQL databases like MongoDB) based on evolving data requirements.

c. Machine Learning Model Adaptability

- Model Retraining: The NLP model used for query and reasoning can be periodically retrained with new datasets to improve accuracy and adapt to evolving financial language trends.
- Continuous Learning: Incorporating user feedback allows the system to adapt dynamically, refining its reasoning and query generation over time.

d. Cross-Platform Accessibility

- Frontend Flexibility: Built using React.js, the frontend is designed to work seamlessly across different devices (desktop, tablet, mobile) with minimal reconfiguration.
- API-First Design: The backend exposes REST APIs, making the system adaptable to future interfaces such as mobile apps or voice-enabled assistants.

1.4 Relevance and Impact

The project provides a significant contribution to:

1. Non-technical Users: Enabling easy interaction with financial data through natural language.
2. Business Decision-Making: Facilitating quick insights and reasoning, such as expense patterns or revenue trends.
3. Personal Finance Management: Assisting individuals in managing budgets, tracking expenses, and identifying savings opportunities.
4. Future Research: Establishing a foundation for integrating advanced AI models into database systems for reasoning and automation.

CHAPTER-2
LITERATURE REVIEW

- [1] Rungsiman Nararatwong, Chung-Chi Chen, Natthawut Kertkeidkachorn, Hiroya Takamura, Ryutaro Ichise, Artificial Intelligence Research Center, AIST, Japan Advanced Institute of Science and Technology ,Tokyo Institute of Technology | DBQR-QA: A Question Answering Dataset on a Hybrid of Database Querying and Reasoning | With advancements in NLP, zero-shot learning has emerged as a prominent method in the study of LLMs | evaluate the ability of LLMs to generate code for querying and reasoning | adopts financial data in line with prior research, this specificity means that our findings are predominantly tailored to financial documents | primary emphasis is on assessing the capabilities of LLMs in a zero-shot setting.

- [2] Shumo Chu Chair of the Supervisory Committee: Professor Dan Suciu Paul G. Allen School of Computer Science, University of Washington |2019| Automated Reasoning of Database Queries.| This thesis describes Cosette, the first tool for automated reasoning the equivalences of SQL queries | The core of Cosette is a formal semantics of SQL based on semirings | This semantics covers major SQL features, including sophisticated ones such as grouping, aggregate, correlated sub-queries, and integrity constraints | Cosette can answer the yes or no question on database queries but can't answer why | high level explanations.

- [3] Tejaswini Kumar Independent Researcher Columbia Univ Alum Kishor Yadav Kommanaboina Independent Researcher The Ohio State Univ Alum Bhargava Kumar Independent Researcher Columbia Univ Alum | 2024 | Next-Generation Text-to-SQL: A Survey of Advanced Reasoning Enhancements Techniques | This survey paper reviews the latest advancements in this field, focusing on methods such as Chain of Thought (CoT) prompting, ACTSQL, QDecomp, Least-to-Most Prompting, Program of Thoughts (PoT), SQL-Craft, and SQL-PaLM.

- [4] Tii sheard David Stemple Computer and Information Science University of Massachusetts, Amherst, Ma. 01003 USA | 2019 | coping with complexity In Automated reasoning about Database systems | Automated reasoning about database systems refers to using a program or programs to draw inferences about properties of systems and can be used by designers to analyze system designs, by query processors to optimize queries, and by transaction compilers or interpreters to optimize the checking of integrity constraints.

- [5] D Calvanese, G De Giacomo, M Lenzerini, M Y Vardi | 2003 | Reasoning on regular path queries | A semistructured model conceives a database essentially as a finite directed labeled graph whose nodes represent objects, and whose edges represent relationships between objects. In the same way as conjunctive queries form the core of any query language for the relational model, regular path queries (RPQs) and their variants are considered the basic querying mechanisms for semistructured data.Besides the basic task of query answering, i.e., evaluating a query over a database, databases should support other reasoning services related to querying | demonstrate that the basic services for reasoning about two way regular path queries are decidable, thus showing that the limited form of recursion expressible by these queries does not endanger the decidability of reasoning.

- [6] Hongyu Ren, Mikhail Gailkin, Michael Cochenz, Zhaocheng Zhu, Jure Leskovec | 2023 | Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases | In this paper, we provide a holistic survey of CLQA with a detailed taxonomy studying the field from multiple angles, including graph types (modality, reasoning domain, background semantics), modeling aspects (encoder, processor, decoder), supported queries (operators, patterns, projected variables), datasets, evaluation metrics, Applications. Refining the CLQA task, we introduce the concept of Neural Graph Databases (NGDBs). Extending the idea of graph databases (graph DBs), NGDB consists of a Neural Graph Storage and a Neural Graph Engine. Inside Neural Graph Storage, we design a graph store, a feature store, and further embed information in a latent embedding store using an encoder | As the existing evaluation protocol appears to be limited (focusing only on inferring hard answers) there is a

need for a more principled evaluation framework and metrics covering various aspects of the query answering workflow.

- [7] Cecilia Katzeff Department of Psychology, University of Stockholm, S-106 91 Stockholm, Sweden | 1988 | The effect of different conceptual models upon reasoning in a database query writing task | This paper proposes that database query writing may be viewed as a hypothesis testing activity. The paper attempts to contribute to the knowledge of what constitutes an appropriate model for efficient use of a database query language | An empirical study was carried out, where subjects were divided into four different conditions. One group received no model; one group received a table model; two groups received descriptions of the query language in terms of sets, one description also providing a general logical explanation. The subjects replied to questions by posing queries to a database system. Each subject was given questions with two types of linguistic structure—one involving the intersection and the other the union of negative sets.

- [8] Ben Bogin, Matt Gardener, Jonathan Berant | 2019 | Global Reasoning over Database Structures for Text-to-AQL Parsing | State-of-the-art semantic parsers rely on auto-regressive decoding, emitting one symbol at a time. When tested against complex databases that are unobserved at training time (zero-shot), the parser often struggles to select the correct set of database constants in the new database, due to the local nature of decoding | In this paper, we demonstrate the importance of global decision-making for zero-shot semantic parsing, where selecting the relevant set of DB constants is challenging. We present two main technical contributions. First, we use a gating GCN that globally attends the input question and the entire DB schema to softly-select the relevant DB constants. Second, we re-rank the output of a generative semantic parser by globally scoring the set of selected DB-constants. Importantly, these contributions can be applied to any zero-shot semantic parser with minimal modifications. Empirically, we observe a substantial improvement over the state-of-the-art on the SPIDER dataset, showing the effectiveness of both contributions.

- [9] Jean Lee, Nicholas Stevens, Soyeon Caren Han, Minseok Song | 2024 | A Survey

of Large Language Models in Finance(FinLLMs) | This survey provides a comprehensive overview of FinLLMs, including their history, techniques, performance, and opportunities and challenges. Firstly, we present a chronological overview of general-domain Pre-trained Language Models (PLMs) through to current FinLLMs, including the GPT-series, selected open-source LLMs, and financial LMs. Secondly, we compare five techniques used across financial PLMs and FinLLMs, including training methods, training data, and fine-tuning methods. Thirdly, we summarize the performance evaluations of six benchmark tasks and datasets. In addition, we provide eight advanced financial NLP tasks and datasets for developing more sophisticated FinLLMs | The challenge of developing real-world financial applications relates to non-technical issues, including business needs, industry barriers, data privacy, accountability, ethics, and the understanding gap between financial experts and AI experts.

- [10] Hongyang Yang, Xiao-Yang Liu, Christina Dan Wang | 2023 | FineGPT: Open-Source Financial Large Language Models | In this paper, we present an open-source large language model, FinGPT, for the finance sector. Unlike proprietary models, FinGPT takes a data-centric approach, providing researchers and practitioners with accessible and transparent resources to develop their FinLLMs. We highlight the importance of an automatic data curation pipeline and the lightweight low-rank adaptation technique in building FinGPT | The challenge of developing real-world financial applications relates to non-technical issues, including business needs, industry barriers, data privacy, accountability, ethics, and the understanding gap between financial experts and AI experts.

- [11] LuFG TCS, RWTH Aachen, Ahornstraße 55, 52074, Aachen, Germany Mohand-Saïd Hacid | 2000 | Representing and Reasoning on Conceptual Queries Over Image Databases | In this paper we develop a knowledge-based framework for modeling and retrieving image data by content. To represent the various aspects of an image object's characteristics, we propose a model which consists of three layers: (1) Feature & Content Layer, intended to contain image visual features such as contours, shapes, etc.; (2) Object Layer, which provides the (conceptual) content dimension of images; and (3) Schema Layer, which contains the structured

abstractions of images, i.e., a general schema about the classes of objects represented in the object layer. We propose two abstract languages on the basis of description logics: one for describing knowledge of the object and schema layers, and the other, more expressive, for making queries | As the amount of information contained in the previous layers may be huge and operations performed at the Feature & Content Layer are time-consuming, resorting to the use of materialized views to process and optimize queries may be extremely useful. For that, we propose a formal framework for testing containment of a query in a view expressed in our query language.

- [12] Werner Nutt, Sergey Paramonov, Ognjen Savkovic | 2015 | Implementing Query Completeness Reasoning | With this paper we make two main contributions: (i) we develop techniques to reason about the completeness of a query answer over a partially complete database, taking into account constraints that hold over the database, and (ii) we implement them by an encoding into logic programming paradigms. As constraints we consider primary and foreign keys as well as finite domain constraints | In this way we can identify more situations in which a query is complete than was possible with previous work. For each combination of constraints, we establish characterizations of the completeness reasoning and we show how to translate them into logic programs. As a proof of concept we ran our encodings against test cases that capture characteristics of a real-world scenario.

- [13] Danna Zheng, Mirella Lapata, Jeff Z.Pan | Archer: A Human-Labeled Text-to-SQL Dataset with Arthemetic, Commonsense and Hypothetical Reasoning | The text-to-SQL task is an important NLP task, which maps input questions to meaningful and executable SQL queries, enabling users to interact with databases in a more intuitive and userfriendly manner. State-of-the-art methods (Pourreza and Rafiei, 2024; Li et al., 2023a,b; Scholak et al., 2021) relying on large language models have achieved execution accuracy above 75% on the Spider dataset(Yu et al., 2018), which encompasses complex SQL grammar and cross-domain settings. Recently, Pourreza and Rafiei (2024) achieved remarkable results with an impressive 85.3% execution accuracy on the Spider dataset, leveraging the enhanced capabilities of GPT-4 | The three different types of reasoning in Archer: arithmetic, commonsense, and hypothetical reasoning | The evaluation metric used in Archer is execution accuracy. This metric

may be perceived as an upper-bound performance measure, as SQL queries producing the same execution results on a single database may still possess different semantic meanings.

- [14] Xuanliang Zhan, Dingzirui Wang, Longxu Duo, Qingfu Zhu, Wanxiang Che | 2024 | A Survey of Tables Reasoning with Large Language Models | Table reasoning, which aims to generate the corresponding answer to the question following the user requirement according to the provided table, and optionally a text description of the table, effectively improving the efficiency of obtaining information. Recently, using Large Language Models (LLMs) has become the mainstream method for table reasoning, because it not only significantly reduces the annotation cost but also exceeds the performance of previous methods | existing research still lacks a summary of LLM-based table reasoning works. Due to the existing lack of research, questions about which techniques can improve table reasoning performance in the era of LLMs, why LLMs excel at table reasoning, and how to enhance table reasoning abilities in the future, remain largely unexplored. This gap significantly limits progress in research.

- [15] Hanchen Xia, Feng Jiang, Naihao Deng, CunxiangWang, Guojiang Zhao, Rada ihalcea, Yue Zhang | 2024 | "This is My SQL, Are You with Me?" A Consensus-Based Multi-Agent System for Text-to-SQL Tasks | Large Language Models (LLMs) have demonstrated strong performance on various tasks. To unleash their power on the Text-to-SQL task, we propose R 3 (Review-Rebuttal-Revision), a consensus-based multi-agent system for Textto-SQL tasks. R 3 outperforms the existing single LLM Text-to-SQL systems as well as the multi-agent Text-to-SQL systems by 1.3% to 8.1% on Spider and Bird. Surprisingly, we find that for Llama-3-8B, R 3 outperforms chain-ofthought prompting by over 20%, even outperforming GPT-3.5 on the development set of Spider | Due to the scope of the study, we only test a limited number of LLMs. The performance gap between 1R-Lp and 3R-Lp demonstrates that the number of reviewers is a worthwhile topic of research. However, this work does not delve into this much.

- [16] Yuan Tian, Jonathan K. Kummerfeld, Toby Jia-Jun Li, Tianyi Zhang | 2024 | SQLucid: Grounding Natural Language Database Queries with Interactive

Explanations | This paper introduces SQLucid, a novel user interface that bridges the gap between non-expert users and complex database querying processes. SQLucid addresses existing limitations by integrating visual correspondence, intermediate query results, and editable step-by-step SQL explanations in natural language to facilitate user understanding and engagement. This unique blend of features empowers users to understand and refne SQL queries easily and precisely | There are several limitations in the design of our user study. First, although our participants represented a wide range of expertise levels in SQL, they were all university students. In the future, we plan to recruit industrial practitioners to study the real-world adoption and ecological validity of SQLucid.

- [17] Xiaohu Zhu, Qian Li, Lizhen Cui, Yongkang Liu | 2024 | Large Language Model Enhanced Text-to-SQL Generation: A Survey | Text-to-SQL translates natural language queries into Structured Query Language (SQL) commands, enabling users to interact with databases using natural language. Essentially, the text-to-SQL task is a text generation task and its development primarily dependent on changes in language models. Especially with the rapid development of Large Language Models (LLMs), the pattern of text-to-SQL has undergone significant changes. Existing survey work mainly focuses on rule-based and neural-based approaches, still lacking a survey of Text-to-SQL with LLMs | In this paper, we survey the large language model enhanced text-to-SQL generations, classifying them into prompt engineering, fine-tuning, pre-trained and Agent groups according to training strategies. And we also summarize datasets and evaluation metrics comprehensively | Traditional methods mainly rely on bi-structured models. These approaches typically use LSTM-based and Transformer-based models to generate SQL queries by learning a contextual representation between natural language questions and database tables.

- [18] Limin Ma, Ken Pu, Ying Zhu | 2024 | Evaluating LLMs for Text-to-SQL Generation with Complex SQL Workload | This study presents a comparative analysis of the a complex SQL benchmark, TPC-DS, with two existing text-to-SQL benchmarks, BIRD and Spider. Our findings reveal that TPC-DS queries exhibit a significantly higher level of structural complexity compared to the other two benchmarks. This underscores the need for more intricate benchmarks to simulate

realistic scenarios effectively. To facilitate this comparison, we devised several measures of structural complexity and applied them across all three benchmarks | We have compared TPC-DS with existing text-to-SQL benchmarks, BIRD and SPIDER, and found that TPC-DS queries exhibit a significantly higher level of structural complexity compared to the other two benchmarks. We have also evaluated the performance of 11 LLMs in generating SQL queries based on the TPC-DS workload.

- [19] Ge Qu, Jinyang Li, Bowen Li,Bowen-Qin, Nan Huo, Chenhao Ma, Reynold Cheng | 2024 | Before Generation, Align it! A Novel and Effective Strategy for Mitigating Hallucinations in Text-to-SQL Generation | In this work, we first identify and categorize the common types of hallucinations at each stage in text-to-SQL. We then introduce a novel strategy, Task Alignment (TA), designed to mitigate hallucinations at each stage. TA encourages LLMs to take advantage of experiences from similar tasks rather than starting the tasks from scratch. This can help LLMs reduce the burden of generalization, thereby mitigating hallucinations effectively. We further propose TA-SQL, a text-to-SQL framework based on this strategy | Large Language Models (LLMs) driven by In-Context Learning (ICL) have significantly improved the performance of text-to-SQL. Previous methods generally employ a two-stage reasoning framework, namely 1) schema linking and 2) logical synthesis, making the framework not only effective but also interpretable.

- [20] Lixia Wu, Peng Li, Junhong Lou, Lei Fu | 2024 | DatGpt-SQL-7B: An Open-Source Language Model for Text-to-SQL | In addressing the pivotal role of translating natural language queries into SQL commands, we propose a suite of compact, fine-tuned models and self-refine mechanisms to democratize data access and analysis for non-expert users, mitigating risks associated with closed-source Large Language Models. Specifically, we constructed a dataset of over 20K sample for Text-to-SQL as well as the preference dataset, to improve the efficiency in the domain of SQL generation | This paper has introduced a novel approach to address the text-to-SQL challenge by proposing a series of compact, fine-tuned models that significantly enhance the efficiency of data analysis and retrieval for non-expert users. By leveraging high-quality datasets, schema linking techniques, and long-chain reasoning through stepwise Direct Preference Optimization, our model, Data-GPT-sql, demonstrates

competitive performance on the spider-dev benchmarks, achieving 87.2% (EX) and 83.5% (TS) accuracy.

CHAPTER-3
RESEARCH GAPS OF EXISTING METHODS

Research Gaps in Existing Methods for Financial Database Query and Reasoning Systems Despite significant advancements in natural language processing (NLP) and database query automation, several research gaps remain in developing effective financial QA systems. Below are key areas that need further exploration:

3.1. Limited Domain-Specific Training for Financial NLP Models

- Gap: Most NLP models, such as GPT and BERT, are trained on general-purpose datasets, limiting their ability to understand financial jargon, context-specific terms, and industry regulations.
- Impact: Financial questions often involve complex domain-specific reasoning that generic models struggle to interpret correctly, leading to inaccurate SQL query generation or irrelevant results.
- Solution Need: Domain-specific pretraining or fine-tuning with large financial datasets, such as earnings reports, financial news, and market data, to enhance model accuracy and relevance.

3.2. Handling Real-Time and Dynamic Financial Data

- Gap: Financial data is highly dynamic, with frequent updates from various sources like stock markets, news, and social media. Existing systems often fail to handle real-time data efficiently due to the high temporal sensitivity of financial information.
- Impact: Delays in processing real-time data can result in outdated or irrelevant query responses, particularly in time-sensitive financial decisions.
- Solution Need: Development of real-time data pipelines with robust data cleaning, noise reduction, and continuous model updating capabilities to address dynamic financial environments

3.3. Low Accuracy in SQL Generation for Complex Queries

- Gap: Current NLP-to-SQL models often struggle with generating complex SQL queries

involving multiple joins, nested queries, and advanced conditions, especially for large datasets.

- Impact: This leads to incomplete or incorrect data retrieval, reducing trust in automated query systems.
- Solution Need: Improved models that understand and generate more complex SQL queries by incorporating advanced reasoning techniques, such as logical frameworks or reinforcement learning.

3.4. Lack of Explainability and Transparency in Query Generation

- Gap: Existing systems often operate as "black boxes," providing SQL queries without explaining how they are derived from user input, which poses challenges in critical financial contexts where transparency is required.
- Impact: Users may find it difficult to trust the system or verify the accuracy of query results, particularly in regulatory or compliance-sensitive environments.
- Solution Need: Integration of explainable AI (XAI) techniques to provide clear insights into how queries are generated, improving user trust and system accountability.

3.5. Inadequate Support for Multimodal Financial Data Sources

- Gap: Current systems primarily focus on structured datasets (e.g., databases) and lack support for unstructured data sources like financial reports, news articles, and social media.
- Impact: This limits the system's ability to provide comprehensive answers that integrate diverse types of financial data.
- Solution Need: Incorporation of multimodal data processing techniques that combine structured and unstructured data for richer query results.

3.6. Scalability Issues with Large Financial Datasets

- Gap: Existing query systems face performance bottlenecks when dealing with large-scale financial datasets, affecting response time and accuracy.
- Impact: This limits their applicability in large organizations handling extensive financial data across multiple sectors.
- Solution Need: Development of more scalable NLP-to-SQL frameworks using

distributed computing, indexing optimizations, and efficient query execution strategies.

These research gaps highlight the need for more sophisticated, domain-specific, and transparent solutions to enhance financial QA systems. Addressing these gaps will lead to more reliable, efficient, and user-friendly query systems for financial data analysis.

CHAPTER-4
PROPOSED METHODOLOGY

The methodology for the "Database Query and Reasoning" system is designed to bridge the gap between natural language queries and structured SQL statements. It encompasses a structured approach that integrates advanced natural language processing (NLP), database management, and reasoning capabilities to deliver a comprehensive solution. The methodology is divided into several stages, each addressing a specific aspect of the system's functionality.

4.1. Data Ingestion and Preprocessing

The system begins by enabling users to upload datasets in CSV format. This stage ensures flexibility in handling diverse data types and structures. The uploaded CSV file is parsed using Python's Pandas library, which facilitates efficient data handling and preprocessing. The pre-processed data is then stored in an SQLite database to serve as the backend for query execution. Once the data is uploaded, the system extracts the database schema, including table names, column names, and data types, using SQLite's PRAGMA commands. This schema is presented to the user in an easily readable format to provide clarity about the structure of the data. By ensuring schema awareness, the system creates a foundation for accurate SQL query generation, as column names and data types are critical for query formulation.

4.2. Natural Language Query Input

The user interface, developed using Streamlit, allows users to input their queries in natural language. The primary objective of this stage is to ensure that users can interact with the system intuitively, without requiring technical expertise in SQL or database management.

The user input is passed to a prompt-engineering module, which prepares a structured instruction set for the generative AI model. This instruction set incorporates the extracted schema, user query, and specific guidelines for SQL generation. The prompt is carefully designed to ensure that the model generates SQL statements that are syntactically correct, semantically meaningful, and tailored to the structure of the uploaded dataset.

4.3. Query Generation Using Gemini AI

The system leverages Google's Gemini AI model for query generation. Gemini, a state-of-the-art generative AI platform, is particularly effective in translating complex natural language inputs into structured outputs. The model is configured to process the prompt and generate an SQL query that aligns with the user's intent and the database schema.

This stage involves several key steps:

Prompt Engineering: The schema, column types, and user query are combined into a comprehensive prompt. For example, the prompt explicitly states the table name and schema details, ensuring the model generates queries aligned with the database structure.

Query Optimization: The generated SQL query is validated to ensure accuracy and optimized for performance. This includes checking for syntax errors, logical inconsistencies, and compatibility with the SQLite database.

4.4. Query Execution

The validated SQL query is executed on the SQLite database using Python's sqlite3 library. Depending on the query type, the system either retrieves data (for SELECT queries) or modifies the database (for INSERT, UPDATE, or DELETE queries).

For SELECT queries, the retrieved results are formatted into a tabular structure using Pandas and displayed to the user. For other types of queries, the system provides feedback about the success of the operation.

4.5. Reasoning and Explanation Module

The reasoning module is a critical differentiator for this project. Beyond executing queries, the system provides detailed explanations for the query and its results. This module employs the Gemini AI model with a reasoning-specific prompt that includes:

- The natural language query entered by the user.
- The generated SQL query.
- The retrieved results (if any).

The reasoning output is structured to explain the following:

Interpretation of the User Query: How the natural language input was understood in the context of the schema.

Query Structure: Why specific SQL clauses, column names, or conditions were

included in the query.

Result Analysis: A summary of the returned data, including any patterns, insights, or empty result explanations.

This explanation enhances the system's usability and transparency, empowering users to understand both the technical and logical aspects of the query process.

4.6. Error Handling and Validation

The system incorporates robust error-handling mechanisms to manage potential issues, such as invalid SQL generation, mismatched schemas, or ambiguous user queries. For instance:

- If the model generates a query referencing non-existent columns, the system flags the error and provides feedback.
- Ambiguous queries prompt the system to seek clarification from the user.

Additionally, the system ensures schema compatibility by validating user-uploaded datasets and aligning queries with the database structure.

4.7. User Interface and Experience

The entire methodology is encapsulated in a user-friendly web application built with Streamlit. Key features include:

- A file uploader for seamless dataset integration.
- Schema visualization for database transparency.
- An input field for natural language queries.
- Real-time query generation, execution, and reasoning display.

The interface is designed to prioritize simplicity, ensuring accessibility for non-technical users while maintaining robust functionality for advanced users.

4.8. Testing and Evaluation

Extensive testing is conducted to ensure the system's accuracy, scalability, and robustness. Test cases include various query types, dataset structures, and ambiguous inputs. Feedback from users informs iterative improvements to the prompt engineering, error handling, and reasoning explanations.

By implementing this methodology, the proposed system will effectively bridge the gap

between non-technical users and complex financial databases, enabling fast, accurate, and accessible data-driven insights for various financial decision-making processes.

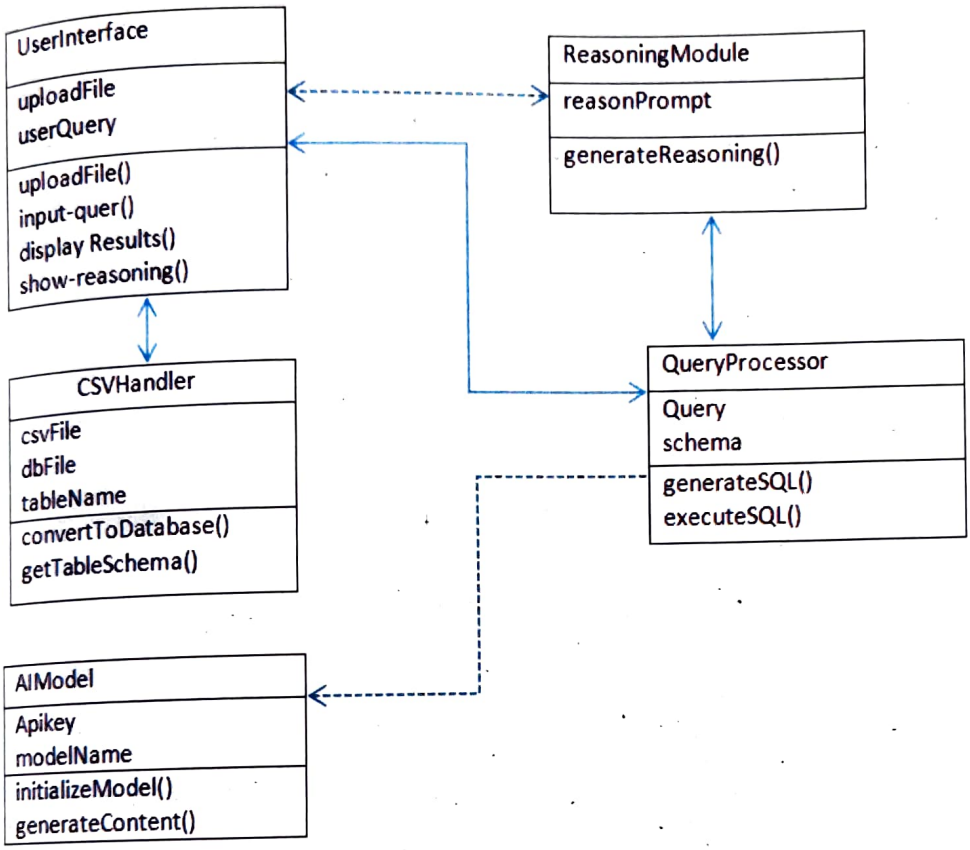

Figure 4.1 Class Diagram

CHAPTER-5
OBJECTIVES

The primary objective of this project is to develop a Natural Language Processing (NLP)-based Financial Database Query and Reasoning System that allows non-technical users to interact with large financial datasets using natural language queries. The system will bridge the gap between complex financial data and user-friendly access, facilitating efficient data retrieval and decision-making. The specific objectives are outlined below:

5.1. Develop an NLP-Driven Query System

- Design and implement a system that translates natural language financial queries into SQL commands, enabling seamless interaction with large databases without requiring SQL knowledge.
- Utilize NLP models fine-tuned on financial datasets to ensure high accuracy in interpreting domain-specific language and complex queries.

5.2. Enable Real-Time Data Retrieval and Reasoning

- Implement mechanisms to handle real-time data updates, allowing users to access up-to-date financial information quickly and efficiently.
- Ensure the system can reason over large datasets, providing accurate responses even for complex, multi-layered financial queries involving trends, forecasts, and historical data.

5.3. Improve Accessibility to Financial Insights

- Create an intuitive user interface that simplifies data querying and visualization, empowering non-technical users, such as financial analysts, investors, and business managers, to make informed decisions.
- Offer diverse output formats, including tables, charts, and summaries, to cater to various user preferences.

5.4. Ensure System Scalability and Performance

- Design the system to be scalable, initially using SQLite for development and enabling future transitions to more robust databases (e.g., PostgreSQL or MySQL) for handling large-scale financial data.
- Optimize SQL generation and query execution processes to maintain fast response times, even with extensive datasets.

5.5. Incorporate Continuous Model Learning and Feedback

- Implement a feedback loop to fine-tune the NLP model based on user interactions and query results, continuously improving the system's accuracy and relevance over time.
- Integrate error-handling and query refinement suggestions to assist users in clarifying ambiguous or incomplete queries.

5.6. Enhance Transparency and Trust through Explainability

- Develop explainable AI (XAI) features that provide clear insights into how SQL queries are generated from natural language inputs, fostering user trust and confidence in the system's responses.
- Enable users to understand the reasoning behind query results, particularly in high-stakes financial contexts requiring regulatory compliance and accountability.

These objectives will guide the development of a robust, user-friendly financial query and reasoning system that democratizes access to complex financial data, fostering data-driven decision-making across various financial sectors.

Table 5.1 Functional Requirements

S.NO	REQUREMENT	DESCRIPTION
1	Natural Language Query Input	Users can input queries in natural language.
2	NLP Model Integration	The system processes user queries and generates SQL using an NLP model.
3	SQL Query Execution	Executes generated SQL queries on the database and retrieves results.
4	Data Visualization	Displays results as tables, charts, or graphs based on user preference.
5	Feedback Loop for Continuous Learning	Collects user feedback to improve NLP model accuracy.
6	Scalable Database Support	Initially uses SQLite with the capability to transition to PostgreSQL for larger datasets.
7	User Authentication and Role-Based Access Control	Ensures secure access to the system and different data views based on user roles.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

The design and implementation of the "Database Query and Reasoning" system focus on integrating natural language processing, database management, and reasoning functionalities into a cohesive solution. The system is built on a modular architecture that ensures flexibility, scalability, and ease of use for both technical and non-technical users. Below, the design components and implementation details are discussed in detail.

6.1. System Architecture Overview

The system architecture is divided into three primary layers:

1. Frontend Layer: A user-facing interface developed using Streamlit that allows users to interact with the system.
2. Processing Layer: A backend engine that handles natural language input, SQL query generation, reasoning, and database interactions.
3. Database Layer: A storage layer that uses SQLite to store user-uploaded datasets and execute SQL queries.

These layers communicate seamlessly to process user inputs, generate and execute SQL queries, and provide results with reasoning.

6.2. Component Design

Frontend Design

The frontend is designed to offer a user-friendly interface. Key features include:

- File Uploader: Enables users to upload datasets in CSV format.
- Schema Viewer: Displays the database schema in a tabular format, showing column names and data types.
- Query Input Field: Allows users to enter queries in natural language.
- Output Display: Shows the generated SQL query, query results, and reasoning in separate sections.

Streamlit, a Python-based web framework, is used to create this interface due to its simplicity and ability to render dynamic content in real time.

Backend Processing

The backend processing layer is the core of the system, responsible for natural language understanding, query generation, and reasoning. It consists of the following components:

Dataset Ingestion Module:

- Functionality: Handles CSV file uploads and converts the data into an SQLite database.
- Implementation:
  - Uses Python's Pandas library to read CSV files and clean the data.
  - The cleaned data is stored in an SQLite database using the to_sql() method.
- Error Handling: Validates the uploaded file to ensure compatibility and displays descriptive error messages for unsupported formats.

Schema Extraction Module:

- Functionality: Extracts and presents the table schema to the user.
- Implementation:
  - SQLite's PRAGMA table_info command fetches column names and data types.
  - The extracted schema is displayed in a readable format using Pandas.
- Importance: This step ensures that the user and system are aware of the database structure, which is essential for accurate query generation.

Query Generation Module:

- Functionality: Converts natural language queries into SQL statements using Google's Gemini AI model.
- Implementation:
  - A prompt is dynamically constructed to include:
    - Table name and schema information.
    - User's natural language query.
    - Instructions for SQL generation.
  - The Gemini model processes the prompt and generates the SQL query.
- Validation: Generated queries are validated for syntax correctness and compatibility with the SQLite database.

Query Execution Module:

- Functionality: Executes the SQL query on the SQLite database and retrieves results.
- Implementation:
  - Python's sqlite3 library executes the query.
  - For SELECT queries, results are fetched and converted into a Pandas DataFrame for display.
  - For other queries (INSERT, UPDATE, DELETE), success messages are shown.
- Error Handling: Detects and handles errors such as syntax issues, invalid column references, or mismatched data types.

Reasoning Module:

- Functionality: Provides a detailed explanation of the query generation process and results.
- Implementation:
  - A reasoning prompt is created with the following details:
    - Natural language query.
    - Generated SQL query.
    - Retrieved results (or a note if no results are found).
  - Gemini AI processes the prompt and generates reasoning in natural language.
- Output: Reasoning is displayed in a user-readable format, fostering transparency and understanding.

Database Design

SQLite is used as the database management system due to its lightweight nature and ease of integration with Python. Key considerations include:

- Dynamic Table Creation: Tables are dynamically created based on the uploaded dataset's structure.
- Schema Management: The schema is extracted and validated to ensure alignment with user queries.
- Scalability: Although SQLite is limited to smaller datasets, the design can be extended to more robust systems like PostgreSQL or MySQL for larger applications.

6.3. Implementation Workflow

The implementation workflow consists of the following steps:

1. File Upload and Data Storage:

   The user uploads a CSV file.

   The system converts the file into a database table and extracts the schema.

2. Natural Language Input:

   The user inputs a query in plain English.

3. Prompt Construction:

   The schema and user query are combined into a structured prompt.

4. SQL Query Generation:

   Gemini AI generates an SQL query based on the prompt.

5. Query Validation and Execution:

   The system validates the SQL query.

   The query is executed on the SQLite database.

6. Result Display:

   Query results are retrieved and displayed in tabular format.

7. Reasoning Generation:

   Gemini AI provides an explanation for the query and its results.

6.4. Key Features and Enhancements

1. User-Focused Design:

   The interface ensures accessibility for users with varying technical expertise.

   Schema visualization aids in understanding database structure.

2. Dynamic Prompt Engineering:

   Prompts adapt to the dataset's schema, ensuring relevant and accurate SQL generation.

3. Error Handling:

   Detailed error messages guide users in resolving issues such as invalid queries or incompatible datasets.

4. Reasoning Insights:

   The reasoning module demystifies the AI's decision-making process, fostering trust and usability.

6.5. Scalability and Future Enhancements

While the system is designed for SQLite and smaller datasets, it can be scaled to support enterprise-level databases. Future enhancements include:

- Integration with cloud-based databases for larger datasets.
- Support for additional file formats (e.g., Excel, JSON).
- Multilingual query support using advanced NLP models.
- Enhanced reasoning outputs with visualizations and statistical analysis.

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

(GANTT CHART)

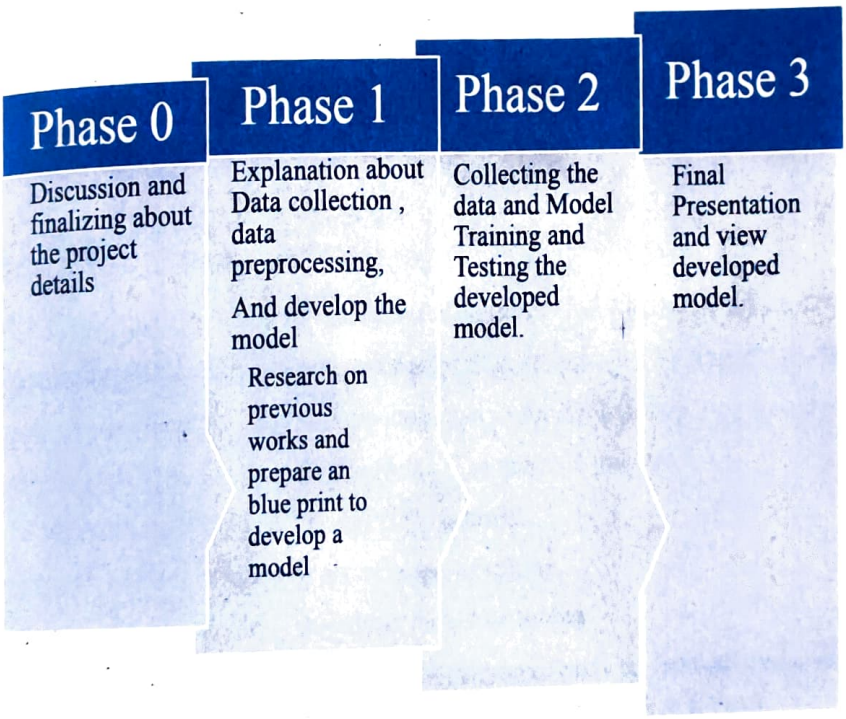

Figure. 7.1 Timeline Gantt Chart

CHAPTER-8
OUTCOMES

The "Database Query and Reasoning" system successfully addresses the challenge of making database interactions more accessible and intuitive for non-technical users. By enabling natural language inputs to drive SQL queries and providing detailed reasoning for the results, the system demonstrates a significant advancement in bridging the gap between human language and structured data querying. The outcomes of this project are detailed below, covering both the technical achievements and their broader implications.

8.1. User-Friendly Database Querying

One of the primary outcomes of the project is the development of a user-friendly interface that allows individuals with little to no knowledge of SQL to interact effectively with databases. Users can upload datasets, view the schema, and input natural language queries without worrying about the complexities of SQL syntax. The interface ensures:

- Seamless data integration through CSV uploads.
- Instant feedback on query generation and execution.
- A clear visualization of query results in tabular format.

This outcome empowers users across diverse domains, such as business, education, and research, to extract insights from their data without requiring technical expertise.

8.2. Automated SQL Query Generation

The integration of Google's Gemini AI model enables the system to accurately translate natural language queries into SQL statements tailored to the uploaded dataset. Key achievements in this area include:

- **Dynamic Prompt Engineering**: The system dynamically constructs prompts by incorporating the database schema and user query, ensuring that the generated SQL aligns with the structure of the data.
- **Accuracy**: The AI model demonstrates a high level of accuracy in generating SQL queries, including those with complex conditions, aggregations, and joins.
- **Adaptability**: The system adapts to various dataset schemas, handling different column names, data types, and table structures effectively.

This automated SQL generation eliminates the learning curve associated with writing queries and enhances productivity for users.

8.3. Transparent Reasoning for Queries and Results

A standout feature of the project is the reasoning module, which provides detailed explanations for the SQL query and its results. The reasoning outputs include:

- **Query Interpretation**: How the system understood the user's natural language input.
- **SQL Query Explanation**: The logic behind the generated SQL statement, including specific clauses and conditions.
- **Result Analysis**: Insights into the retrieved data or explanations for empty results.

This transparency builds user trust in the system's outputs and serves as a valuable learning tool for users who wish to understand SQL query structure and logic.

8.4. Enhanced Accessibility and Inclusivity

By abstracting the technical complexities of database management, the system significantly broadens the accessibility of data querying. Non-technical users, including small business owners, educators, and students, can now interact with databases in an intuitive manner. This inclusivity promotes data-driven decision-making across sectors.

8.5. Robust Backend Functionality

The backend of the system demonstrates robust functionality in handling datasets and executing queries. Specific outcomes include:

- **Efficient Data Handling**: The system processes user-uploaded datasets, converting them into SQLite tables without requiring manual intervention.
- **Error Handling**: Comprehensive error messages guide users in resolving issues, such as invalid file formats, schema mismatches, or ambiguous queries.
- **Scalability**: While designed for smaller datasets using SQLite, the architecture can be extended to support larger, cloud-based databases.

This robust backend ensures reliability and consistency in handling diverse datasets and user queries.

8.6. Educational Benefits

The reasoning module and query visualization also serve as an educational tool, helping users learn SQL and understand database interactions. By breaking down complex queries and providing step-by-step explanations, the system fosters greater understanding of data querying processes, even for novice users.

8.7. Practical Applications Across Domains

The system's outcomes have practical implications for a variety of industries:

- **Business**: Managers and analysts can quickly query sales data, customer trends, or inventory without relying on technical teams.
- **Education**: Teachers and students can use the system to explore data science concepts and learn SQL in an interactive way.
- **Research**: Researchers can analyze datasets efficiently, focusing on insights rather than query formulation.

This versatility highlights the system's potential to transform workflows in multiple domains.

8.8 Overcoming Challenges and Limitations

During development, several challenges were encountered, including:

- **Ambiguous Natural Language Inputs**: The system effectively handles ambiguous queries by incorporating detailed prompts and validation mechanisms.
- **Schema Variability**: The system dynamically adapts to different dataset schemas, ensuring compatibility and accuracy.
- **Model Interpretability**: The reasoning module addresses the challenge of making AI outputs more interpretable, fostering user confidence in the system.

While limitations such as handling extremely large datasets or complex database structures exist, the system lays a strong foundation for future enhancements.

8.9 Future Prospects and Scalability

The outcomes of this project demonstrate its scalability and potential for future development. Enhancements such as integration with cloud databases, support for additional file formats, and multilingual query processing are feasible next steps. Additionally, advanced visualizations for results and reasoning can further enrich the user experience.

CHAPTER-9

RESULTS AND DISCUSSIONS

9.1. Results

9.1.1. Accurate SQL Query Generation

One of the key deliverables of the system is its ability to accurately generate SQL queries from natural language inputs. Using Google's Gemini AI model, the system:

Successfully converts user queries into SQL statements tailored to the dataset schema.

Handles a variety of query types, including simple SELECT statements, conditional queries (WHERE clauses), and aggregation functions (SUM, AVG, etc.).

9.1.2. Reasoning Module Effectiveness

The reasoning module provided detailed and accurate explanations for both the SQL queries and their results. This transparency enhances user trust and understanding of the system.

Example Output:

For the above query, the reasoning module explained:

The purpose of the query: to compute total sales by grouping data by product category.

The logic behind the GROUP BY and SUM clauses.

The significance of the retrieved results in the context of the dataset.

9.1.3. Seamless Dataset Integration

The system successfully processed datasets uploaded in CSV format and dynamically created SQLite tables.

Schema extraction worked as intended, displaying column names and data types.

The system handled different file structures and data types effectively, ensuring compatibility.

9.1.4. User-Friendly Interface

The Streamlit-based frontend provided an intuitive experience:

File uploads were quick and straightforward.

The schema display and query result visualization were clear and accessible.

Natural language input and system outputs were seamlessly integrated.

9.2. Discussions

9.2.1. Strengths of the System

The project has several strengths, including:

1. Accessibility:

   The system lowers the barrier to entry for database interactions by allowing users to interact in natural language. This accessibility is crucial for empowering individuals in non-technical domains.

2. Transparency:

   By providing reasoning for the queries and results, the system addresses one of the key challenges in AI adoption: trust. Users can understand why a specific query was generated and how the results were derived.

3. Scalability:

   The modular design ensures that the system can adapt to various datasets and schemas, making it versatile across domains.

4. Educational:

   The reasoning module not only explains outputs but also serves as an educational tool for users looking to learn SQL and database concepts.

9.2.2. Limitations and Challenges

Despite its strengths, the system has some limitations:

1. Handling Ambiguity in User Queries:

   Ambiguous or poorly worded natural language queries sometimes lead to incorrect SQL generation.

   Example: For the input *"Find the biggest value,"* the system may require more context to determine the correct column for evaluation.

2. Complex Queries:

   Queries involving multiple tables, subqueries, or advanced SQL functions are challenging in the current implementation.

3. Scalability Constraints:

   The use of SQLite limits the system to smaller datasets. For larger, enterprise-level datasets, integration with cloud databases like PostgreSQL or MySQL is

required.

4. AI Model Dependency:

   The system's performance relies heavily on the accuracy of the Gemini AI model. Errors in model predictions can affect overall system usability.

9.2.3. Implications of Results

The results of this project have practical implications across multiple domains:

1. Business and Decision-Making:

   Managers can query sales or customer data in natural language, enabling faster decision-making without needing technical teams.

2. Education and Learning:

   Students and educators can use the system to learn database querying concepts interactively.

3. Research and Analysis:

   Researchers can quickly analyze data without worrying about the underlying SQL syntax, focusing instead on insights.

4. Accessibility for Small Enterprises:

   Small businesses without technical resources can leverage the system to query operational data, improving efficiency.

9.2.4. Potential Improvements

The following enhancements can address the identified limitations:

1. Advanced Query Parsing:

   Incorporating additional NLP techniques to handle ambiguous queries or provide clarification prompts to users.

2. Support for Complex Database Structures:

   Extending the system to handle joins, subqueries, and advanced SQL features.

3. Integration with Cloud Databases:

   Adopting scalable database solutions such as AWS RDS, Google Cloud SQL, or Azure SQL for larger datasets.

4. Multilingual Query Support:

   Expanding the NLP capabilities to process natural language queries in multiple languages, increasing accessibility for global users.

5. Enhanced Reasoning Outputs:

   Integrating visual explanations, such as charts or diagrams, to complement textual reasoning outputs.

In summary, the project successfully demonstrated the potential of NLP-based systems for querying and reasoning over large financial datasets. While the initial implementation met key objectives, future iterations should focus on improving explainability, scalability, and handling of multimodal data to enhance system robustness and usability further.

CHAPTER-10
CONCLUSION

The "Database Query and Reasoning" system represents a significant advancement in making database interactions more intuitive and accessible to users without technical expertise. By leveraging cutting-edge AI technologies, the system effectively bridges the gap between natural language inputs and SQL query generation. This innovation empowers users to explore and analyze data from diverse domains without requiring knowledge of complex database languages or structures.

The project's key contributions include:

- Automated SQL Query Generation: Enabling users to extract valuable insights from datasets using simple natural language questions.
- Reasoning Module: Providing clear and detailed explanations for both queries and results, fostering user trust and understanding.
- User-Friendly Design: Offering a seamless interface for dataset integration, schema visualization, and query execution.

The outcomes of this system demonstrate its potential to democratize access to data-driven decision-making. Its ability to cater to non-technical users opens new opportunities in education, business, research, and beyond. Moreover, the reasoning module serves as an educational tool, helping users learn SQL concepts while interacting with the system.

Despite its success, the system also highlights areas for improvement, including handling ambiguous queries, supporting complex database operations, and scaling for larger datasets. These limitations provide opportunities for future work, such as integrating more advanced natural language processing techniques, expanding database compatibility, and adding multilingual support.

In conclusion, the "Database Query and Reasoning" project sets a strong foundation for a new generation of AI-driven database tools. By prioritizing accessibility, transparency, and usability, this system is well-positioned to make a meaningful impact across industries and pave the way for further innovation in user-centric data solutions.

CHAPTER-11

FUTURE ENHANCEMENT

Multi-Database Support: Add compatibility for other database systems like MySQL, PostgreSQL, or MongoDB to widen usability. Advanced Query Suggestions: Enhance the AI model to better interpret complex natural language queries, including joins, subqueries, and aggregation functions. Improved Reasoning: Integrate additional AI reasoning models or tools to provide more comprehensive and detailed explanations. Data Visualization: Include interactive charts, graphs, or dashboards to visualize query results effectively. File Format Support: Allow users to upload other data formats such as Excel or JSON for broader flexibility. User Authentication: Implement authentication and role-based access control to ensure secure data handling. Customization: Let users customize query preferences, like adding specific filters or saving commonly used queries. Scalability: Optimize the application to handle large datasets efficiently, leveraging cloud storage or distributed databases. API Integration: Provide an API interface so other applications can interact with your tool programmatically. Offline Functionality: Enable offline usage by embedding lightweight AI models and a local database engine.

CHAPTER-12

REFERENCES

- [1] Nararatwong, R., Chen, C.-C., Kertkeidkachorn, N., Takamura, H., & Ichise, R. (2024). "DBQR-QA: A Question Answering Dataset on a Hybrid of Database Querying and Reasoning." Artificial Intelligence Research Center, AIST, Japan Advanced Institute of Science and Technology, Tokyo Institute of Technology.

- [2] Chu, S. (2019). "Automated Reasoning of Database Queries." Paul G. Allen School of Computer Science, University of Washington.

- [3] Kumar, T., Kommanaboina, K. Y., & Kumar, B. (2024). "Next-Generation Text-to-SQL: A Survey of Advanced Reasoning Enhancements Techniques." Independent Researchers.

- [4] Sheard, T., & Stemple, D. (2019). "Coping with Complexity in Automated Reasoning about Database Systems." University of Massachusetts, Amherst, MA, USA.

- [5] Calvanese, D., De Giacomo, G., Lenzerini, M., & Vardi, M. Y. (2003). "Reasoning on Regular Path Queries."

- [6] Ren, H., Gailkin, M., Cochenz, M., Zhu, Z., & Leskovec, J. (2023). "Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases."

- [7] Katzeff, C. (1988). "The Effect of Different Conceptual Models upon Reasoning in a Database Query Writing Task." Department of Psychology, University of Stockholm.

- [8] Bogin, B., Gardener, M., & Berant, J. (2019). "Global Reasoning over Database Structures for Text-to-AQL Parsing."

- [9] Lee, J., Stevens, N., Han, S. C., & Song, M. (2024). "A Survey of Large Language Models in Finance (FinLLMs)."

- [10] Yang, H., Liu, X.-Y., & Wang, C. D. (2023). "FineGPT: Open-Source Financial Large Language Models."

- [11] TCS, L., Hacid, M.-S. (2000). "Representing and Reasoning on Conceptual Queries Over Image Databases." RWTH Aachen, Germany.

- [12] Nutt, W., Paramonov, S., & Savkovic, O. (2015). "Implementing Query Completeness Reasoning."

- [13] Zheng, D., Lapata, M., & Pan, J. Z. (2024). "Archer: A Human-Labeled Text-to-SQL Dataset with Arithmetic, Commonsense, and Hypothetical Reasoning."

- [14] Zhan, X., Wang, D., Duo, L., Zhu, Q., & Che, W. (2024). "A Survey of Table Reasoning with Large Language Models."

• [15] Xia, H., Jiang, F., Deng, N., Wang, C., Zhao, G., Mihalcea, R., & Zhang, Y. (2024). "This is My SQL, Are You with Me? A Consensus-Based Multi-Agent System for Text-to-SQL Tasks."

• [16] Tian, Y., Kummerfeld, J. K., Li, T. J.-J., & Zhang, T. (2024). "SQLucid: Grounding Natural Language Database Queries with Interactive Explanations."

• [17] Zhu, X., Li, Q., Cui, L., & Liu, Y. (2024). "Large Language Model Enhanced Text-to-SQL Generation: A Survey."

• [18] Ma, L., Pu, K., & Zhu, Y. (2024). "Evaluating LLMs for Text-to-SQL Generation with Complex SQL Workload."

• [19] Qu, G., Li, J., Li, B., Qin, B., Huo, N., Ma, C., & Cheng, R. (2024). "Before Generation, Align It! A Novel and Effective Strategy for Mitigating Hallucinations in Text-to-SQL Generation."

• [20] Wu, L., Li, P., Lou, J., & Fu, L. (2024). "DatGPT-SQL-7B: An Open-Source Language Model for Text-to-SQL."

• [21] Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). "Natural Language Interfaces to Databases – An Introduction." *Journal of Natural Language Engineering*.

• [22] Dong, L., & Lapata, M. (2018). "Coarse-to-Fine Decoding for Neural Semantic Parsing." *Proceedings of the Association for Computational Linguistics (ACL)*.

• [23] Finegan-Dollak, C., et al. (2018). "Improving Text-to-SQL Evaluation Methodology." *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

• [24] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You? Explaining the Predictions of Any Classifier." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

• [25] Yu, T., et al. (2018). "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task." *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*.

• [26] Sebastian Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," *arXiv preprint arXiv:1706.05098*, 2017.

• [27] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, "Distilling the Knowledge in a Neural Network," *Advances in Neural Information Processing Systems*, 2015.

• [28] Priyanka Agrawal, Khyati Mahajan, and Vaibhav Kumar, "Eliciting Translation Ability of Large Language Models Using Multilingual Fine-Tuning," *Transactions of the Association for Computational Linguistics*, 2023.

• [29] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman, "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," *Proceedings of ICLR 2019*, 2019.

• [30] Goyal, Angela Fan, and Philipp Koehn, "FLORES-101: Evaluating Multilingual Translation Ability," *Proceedings of EMNLP 2021*, 2021.

APPENDIX-A

PSUEDOCODE

1. Main System Flow Pseudocode:

```
START

// Step 1: User Input

Display input field on the frontend
Wait for user to enter a natural language query
Capture query and send to backend API

// Step 2: Query Processing (Backend)

Recieve query from frontend
Send query to NLP model for processing
Model processes query and generates SQL command
If SQL command is valid Then
  Execute SQL command on the database
  Retrieve results
  Send results to frontend
Else
  Return error message

// Step 3: Display Results

If results are received Then
  Display results in table or chart format
Else
  Display error message or query refinement suggestions

END
```

2. NLP Query Processing Pseudocode

```
Function process_query(user_query)

  // Preprocessing

  Clean user_query by removing stopwords, punctuation
  Tokenize user_query

  // NLP Model Inference

  model_input ← Tokenized user_query
  model_output ← NLP_Model.generate_SQL(model_input)

  // SQL Validation

  If validate_SQL(model_output) Is True Then
    Return model_output
  Else
    Return "Invalid Query, Please Refine"


END FUNCTION
```

3. SQL Execution Pseudocode:

```
Function execute_SQL(sql_query)
  Connect to database

  Try:
    Run sql_query
    Fetch query results
    Return results

  Catch error:
    Return "Database Error: " + error.message

  Finally:
    Close database connection

END FUNCTION
```

4. Frontend Data Visualization Pseudocode:

```
Function display_results(results)

  If results Is Not Empty Then

    If user Selects "Table" format Then
      Render results as table

    Else If user Selects "Chart" format Then
      Render results as chart

    Else
      Display "Unsupported format"

  Else
    Display "No results found"

END FUNCTION
```

5. User Feedback Loop Pseudocode:

```
Function feedback_loop(user_feedback, sql_query, result_accuracy)

  Store user_feedback, sql_query, result_accuracy in feedback database

  If feedback is negative Then
    Log feedback for model retraining

  Else
    Log feedback for model improvement

END FUNCTION
```

APPENDIX-B

SCREENSHOTS

Query Results:

	0	1	2	3	4	5	6	7	8
0	Female	750-67-8428	A	Yangon	Member	Health and beauty	74.69	7	261.415
1	Female	226-31-3081	C	Naypyitaw	Normal	Electronic accessories	15.28	5	3.82
2	Female	355-53-5943	A	Yangon	Member	Electronic accessories	68.84	6	20.652
3	Female	315-22-5665	C	Naypyitaw	Normal	Home and lifestyle	73.56	10	36.78
4	Female	665-32-9167	A	Yangon	Member	Health and beauty	36.26	2	3.626

Reasoning:

Natural Language Question: 'display first five records'

SQL Query:

```
SELECT
  *
FROM uploaded_table
LIMIT 5;
```

Reasoning for the Query:

The natural language question asks to "display first five records." To achieve this in SQL, we use the `SELECT` statement to retrieve all columns (`*`) from the `uploaded_table` . The `LIMIT 5` clause is then used to limit the number of rows returned to the first five rows.

Results Obtained:

```
[('Female', '750-67-8428', 'A', 'Yangon', 'Member', 'Health and beauty', 74.69, 7,
```

Figure B.1 : Output 1

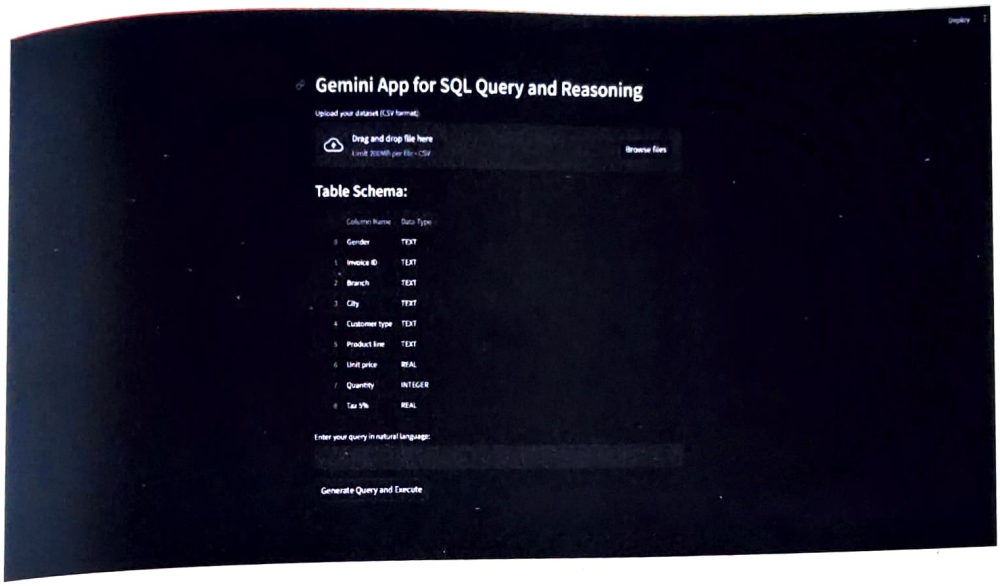

Figure B.2 : Output 2

APPENDIX-C

ISSN: 2582-3930

ACCEPTANCE CERTIFICATE

Impact Factor: 8.448

DOI Prefix: 10.55041

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT (IJSREM)

An Open Access Scholarly Journal || Index in major Databases & Metadata

We are pleased to inform you that your Research Paper titled

Database query and reasoning

has been ACCEPTED for publication in

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT (IJSREM)

VOLUME 09 ISSUE 01 JANUARY - 2025

We are delighted to see your commitment & hardwork to share your research is being recognized. We look foreward to helping you with all of your publication needs. Thank you for choosing IJSREM!!

Editor-in-chief

IJSREM

e-mail: editor@ijsrem.com

www.ijsrem.com

DOI: 10.55041/IJSREM40897

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Kayal Vizhi

in recognization to the publication of paper titled

Database Query and Reasoning

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

Editor-in-Chief
IJSREM Journal

www.ijsrem.com

e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM40897

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Abhinay K

in recognization to the publication of paper titled

Database Query and Reasoning

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

Editor-in-Chief
IJSREM Journal

www.ijsrem.com

e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM40897

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Gowtham Naidu Y

in recognization to the publication of paper titled

Database Query and Reasoning

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

Editor-in-Chief
IJSREM Journal

www.ijsrem.com

e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM40897

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Fairoz Khan P

in recognization to the publication of paper titled

Database Query and Reasoning

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com

DOI: 10.55041/IJSREM40897

ISSN: 2582-3930

Impact Factor: 8.448

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT

An Open Access Scholarly Journal || Index in major Databases & Metadata

CERTIFICATE OF PUBLICATION

International Journal of Scientific Research in Engineering & Management is hereby awarding this certificate to

Sasivaran Reddy K

in recognization to the publication of paper titled

Database Query and Reasoning

published in IJSREM Journal on Volume 09 Issue 01 January, 2025

www.ijsrem.com

Editor-in-Chief
IJSREM Journal

e-mail: editor@ijsrem.com

Similarity Index/Plagiarism check report clearly showing the Percentage(%).

Database query and reasoning

ORIGINALITY REPORT

19%	18%	10%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	4%
2	Submitted to Presidency University Student Paper	2%
3	www.grafiati.com Internet Source	1%
4	ideas.repec.org Internet Source	1%
5	aclanthology.org Internet Source	1%
6	kypros.org Internet Source	1%
7	research.vu.nl Internet Source	1%
8	www.catalyzex.com Internet Source	1%
9	dl.acm.org Internet Source	1%

SUSTAINABLE DEVELOPMENT GOALS

The project work carried out here is mapped to SDG 2: Zero Hunger focuses on increasing agricultural productivity, optimizing crop recommendations, and making government subsidies more accessible.