algorithms

MDPI

Article

Transforming Medical Data Access: The Role and Challenges of Recent Language Models in SQL Query Automation

Nikola Tanković *, Robert Šajina and Ivan Lorencin *

Faculty of Informatics, Juraj Dobrila University of Pula, 52100 Pula, Croatia; robert.sajina@unipu.hr
* Correspondence: nikola.tankovic@unipu.hr (N.T.); ivan.lorencin@unipu.hr (I.L.)

Abstract: Generating accurate SQL queries from natural language is critical for enabling non-experts to interact with complex databases, particularly in high-stakes domains like healthcare. This paper presents an extensive evaluation of state-of-the-art large language models (LLM), including LLaMA 3.3, Mixtral, Gemini, Claude 3.5, GPT-4o, and Qwen for transforming medical questions into executable SQL queries using the MIMIC-3 and TREQS datasets. Our approach employs LLMs with various prompts across 1000 natural language questions. The experiments are repeated multiple times to assess performance consistency, token efficiency, and cost-effectiveness. We explore the impact of prompt design on model accuracy through an ablation study, focusing on the role of table data samples and one-shot learning examples. The results highlight substantial trade-offs between accuracy, consistency, and computational cost between the models. This study also underscores the limitations of current models in handling medical terminology and provides insights to improve SQL query generation in the healthcare domain. Future directions include implementing RAG pipelines based on embeddings and reranking models, integrating ICD taxonomies, and refining evaluation metrics for medical query performance. By bridging these gaps, language models can become reliable tools for medical database interaction, enhancing accessibility and decision-making in clinical settings.

Keywords: healthcare; large language model; natural language query; SQL

Academic Editor: Hao Liu

Received: 26 January 2025
Revised: 18 February 2025
Accepted: 19 February 2025
Published: 21 February 2025

Citation: Tanković, N.; Šajina, R.; Lorencin, I. Transforming Medical Data Access: The Role and Challenges of Recent Language Models in SQL Query Automation. *Algorithms* **2025**, *18*, 124. https://doi.org/10.3390/a18030124

1. Introduction

The transition from paper-based medical records to electronic health records (EHR) represents a significant technological advancement in modern healthcare. EHR systems serve as central repositories of patient information, encompassing medical histories, diagnostic results, treatment plans, and administrative data. They are essential to improve care coordination, reduce errors, and enable evidence-based decision-making [1]. Despite their fundamental role in the modernization of healthcare care, the sheer volume and diversity of data within EHR systems, ranging from structured data such as laboratory values to unstructured data such as physician notes, pose significant challenges for healthcare providers and researchers [2].

In recent years, cutting-edge technologies, especially those based on artificial intelligence (AI) and natural language processing (NLP), have revolutionized the utilization of EHR systems. Advanced NLP models such as BioBERT, ClinicalBERT, and Med-PaLM (Med-PaLM: an LLM from Google for medical domain https://sites.research.google/med-palm/ accessed on 26 January 2025) have shown impressive abilities in obtaining actionable insights from unstructured clinical text. These capabilities facilitate automated diagnosis

coding, summarizing patient encounters, and performing predictive analytics [3–5]. Moreover, integrating large-scale language models (LLMs) fine-tuned into medical corpora has opened new possibilities to improve patient safety, accelerate research, and personalize care [6].

Beyond NLP, interoperability solutions such as FHIR (fast healthcare interoperability resources) have facilitated seamless data exchange between disparate systems, addressing one of the most persistent challenges in implementing EHR [7]. Furthermore, advances in cloud computing and blockchain technologies are improving data security, scalability, and access control, ensuring compliance with stringent privacy regulations such as GDPR and HIPAA [8,9].

Despite these advancements, several challenges remain. Issues such as clinician burnout due to excessive documentation, data security and privacy concerns, and the high system integration costs continue to hinder the full potential of EHR systems [10]. Moreover, ensuring equitable access to these innovations in low-resource settings is an ongoing concern [11].

Creating precise SQL queries from natural language inputs is essential for allowing healthcare personnel to engage with intricate databases. These advancements in using LLMs for converting natural language to SQL queries align with broader trends in healthcare informatics, where user-friendly tools are becoming indispensable for navigating the vast amounts of structured and unstructured data. In the context of EHR systems, this capability has significant implications. For example, a clinician could query patient cohorts with specific comorbidities or treatment regimens directly in natural language, bypassing specialized database expertise. This democratization of database access has the potential to improve clinical decision-making, reduce administrative bottlenecks, and accelerate research.

However, the healthcare domain presents unique challenges that require customized solutions. Often characterized by abbreviations, synonyms, and context-dependent meanings, medical terminology remains a barrier to accurate query generation. Furthermore, the complexity of healthcare databases, such as those built on MIMIC-3, with their intricate schemas and interrelated tables, adds another difficulty for general-purpose models.

The paper's contributions are as follows. First, this study aims to systematically evaluate the effectiveness of state-of-the-art large language models (LLMs) in transforming medical queries into executable SQL statements, with a focus on accuracy, consistency, and cost-efficiency. To achieve this, we assess both proprietary (e.g., GPT-4o and Claude 3.5) and open-source models (e.g., LLaMA 3.3-70B and Qwen-2.5-72B) on the MIMIC-3 and TREQS datasets, utilizing a diverse set of natural language queries. Second, the paper introduces a novel prompt engineering approach tailored for healthcare databases, incorporating schema previews, one-shot examples, and data samples to optimize query accuracy and efficiency. Third, the study identifies critical trade-offs between accuracy, consistency, and token costs, offering actionable insights for selecting models based on specific use cases, such as cost-sensitive deployments or accuracy-critical medical applications. Finally, recognizing the limitations of current models in handling complex medical terminology, we propose future directions, including the integration of retrieval-augmented generation (RAG) pipelines and ICD taxonomies, to enhance query precision and adaptability in high-stakes healthcare environments.

2. Literature Overview

Recent advances in NLP have significantly impacted the healthcare sector, mainly by developing natural language query (NLQ) systems. These systems enable healthcare

professionals to interact with EHRs and clinical databases using everyday language, thus improving the accessibility and utility of data [12].

One notable implementation is the clinical NLP service within the UK's National Health Service (NHS), which utilizes platforms like MedCAT to annotate and extract meaningful information from unstructured clinical text. Such a service has facilitated the improvement of healthcare delivery and supported various clinical and operational use cases [13].

2.1. Classical Approaches and Foundational Works in Text-to-SQL

Several works have explored advances in natural language interfaces (NLIs) for databases and text-to-SQL systems. Marshan et al. [14] introduced MedT5SQL, a transformer-based large language model designed explicitly for text-to-SQL conversion in the healthcare domain, addressing domain-specific challenges. Zhang et al. [15] provided a comprehensive survey of NLIs for tabular data, including querying and visualization methods, focusing on general-purpose frameworks. Earlier foundational works, such as those of Li and Jagadish [16], concentrate on constructing interactive NLIs for relational databases.

Deep learning has played a pivotal role in the advance of text-to-SQL tasks. Katsogiannis-Meimarakis and Koutrika [17] surveyed deep learning approaches, while Iacob et al. [18] reviewed neural methods for NLIs to databases. More recent surveys, such as Qin et al. [19], highlight text-to-SQL parsing methods' evolution and future directions.

2.2. Large Language Models (LLMs) in NLP

Large language models (LLMs), such as GPT and BERT, have revolutionized the field of NLP with their ability to generate and understand human-like text. These models are pre-trained on vast datasets and fine-tuned for specific applications, including healthcare. Recent studies have explored their application in healthcare, focusing on their ability to answer medical queries in multiple languages. This cross-lingual capability is essential to ensure equitable access to healthcare information in multilingual societies [20].

LLMs have shown promise in summarizing patient records, extracting clinical insights, and even assisting medical decision-making. For example, Lee et al. [21] demonstrated using BioBERT, a domain-specific LLM, to achieve state-of-the-art performance in various biomedical text mining tasks.

2.3. Advances in Text-to-SQL for Healthcare

The study by Zhu et al. [22] provides a comprehensive review of text-to-SQL approaches. They categorize these methods based on training strategies, such as prompt engineering and fine-tuning, while summarizing key datasets and evaluation metrics. Their work highlights the importance of metrics like exact matching accuracy and execution accuracy for assessing model performance.

Building on this foundation, we evaluated state-of-the-art (SOTA) LLMs on a medical dataset, focusing on Execution Accuracy while extending the analysis to include an ablation study on prompt components. This study assesses the impact of individual prompt parts on model performance, enabling a granular understanding of prompt design for healthcare-specific tasks. In addition, we conducted a cost analysis of these models to determine their practical viability in resource-constrained healthcare settings.

Using LLMs has significantly enhanced recent advances in text-to-SQL (NL2SQL). Liu et al. [23] provide a comprehensive survey that explores the lifecycle of NL2SQL tasks, covering model development, data synthesis, evaluation, and error analysis. While their work offers valuable information about the general NL2SQL landscape, it does not present results or benchmarks specific to domain-specific applications such as the medical field. Our work addresses this gap by focusing on NL2SQL solutions tailored to the medical

domain, providing empirical results and analysis that advance the understanding and practical utility of such systems in healthcare.

Hong et al. [24] present a thorough survey of text-to-SQL systems based on large language models (LLM), emphasizing the challenges of SQL generation, understanding database schemas, and handling complex user queries. They note that "integrating LLM-based implementation can bring unique opportunities, improvements, and solutions to text-to-SQL research" while identifying challenges such as parameter constraints and the need for tailored optimization methods. However, their survey focuses on general advancements and datasets for evaluating text-to-SQL systems. It does not provide specific results or benchmarks in specialized domains, such as the medical field. Our work addresses this gap by providing empirical evidence, customized solutions, and cost-effectiveness analysis for text-to-SQL applications in the medical domain.

2.4. Retrieval-Augmented Generation (RAG) Techniques

Retrieval-augmented generation (RAG) techniques have been proposed to improve the accuracy and reliability of LLM responses in healthcare applications. RAG systems improve traditional LLMs by integrating them with external knowledge bases, ensuring the responses generated are grounded in verified content. RAG can mitigate issues such as hallucinations and inaccuracies, thus improving the trustworthiness of AI-driven health chatbots [25,26].

2.5. Challenges and Future Directions

Despite these advancements, challenges such as data privacy, interoperability, and the need for explainable AI persist. Addressing these challenges will require continued innovation, particularly in developing multimodal systems, personalized query interpretation, and ensuring fairness in healthcare AI systems [27,28].

3. Research Methodology

This study evaluates various LLMs' performance, consistency, and cost-efficiency in the language-to-SQL transformation task tailored explicitly for the medical domain. The evaluation leverages queries derived from the TREQS and MIMIC-III datasets, focusing on real-world healthcare scenarios. In the medical domain, SQL queries must be accurate and consistent to ensure reliable data retrieval, as errors can have significant consequences. Cost efficiency is also a practical consideration, especially for large-scale deployments. This methodology provides a comprehensive evaluation of these models' technical and operational aspects in the context of medical data processing.

3.1. Dataset and Query Preparation

The TREQS dataset was developed to advance text-to-SQL generation tasks, particularly within the complex and sensitive healthcare domain. This dataset is derived from the extensively documented and deidentified MIMIC-III database, encompassing a wealth of electronic medical records from over 46,520 patients. By providing carefully curated question–query pairs alongside a relational database structure, TREQS enables the creation of systems capable of translating natural language queries into SQL commands, thus facilitating direct and efficient interaction with healthcare databases.

The TREQS dataset is built around a relational database that organizes patient data into five primary tables. These tables cover demographic information, diagnostic records, procedural interventions, prescription details, and laboratory test results. For example, the demographic table contains fields such as patient ID, age, sex, admission details, and mortality status, providing a foundational overview of each patient's medical history. The diagnoses table maps the ICD-9 codes to their respective descriptions, while the

procedures table records the procedural codes and descriptions of medical treatments. Similarly, prescriptions and laboratory test tables provide information on medication administration and diagnostic results, respectively.

A key feature of the TREQS dataset is that it includes natural language question–query pairs specifically designed to train and evaluate text-to-SQL models. These pairs consist of two components. The first component comprises template questions systematically generated using predefined patterns to ensure comprehensive coverage of various database fields. The second component includes natural language questions, which are manually rephrased versions of the template questions. This rephrasing process introduces linguistic variability and improves the data set's applicability to model real-world query interactions.

For example, a template question such as "Retrieve all female patients over 60 years of age" could be rephrased as "Can you show me the list of women older than 60?" Both questions are linked to the same SQL query, enabling robust model training.

We used a total of 1000 natural language queries from the TREQS dataset, originating from the MIMIC III dataset, a widely used deidentified critical care dataset. Each query was carefully designed to reflect realistic use cases, such as retrieving patient demographics, diagnoses, or lab results. To ensure a robust evaluation, queries were executed five times per, resulting in 5000 queries per total. This repetition enabled the assessment of model consistency across multiple runs, as large language models (LLMs) are well known for their inherent variability in responses, even when presented with identical inputs. By running each query multiple times, our objective was to measure the stability of the model's outputs, identify any significant deviations, and evaluate the reliability of their performance under repeated conditions.

Table 1 illustrates some example question–query pairs from the TREQS dataset. These examples highlight the diversity in the phrasing of the questions and the corresponding SQL queries used to extract data from the relational database.

Table 1. Examples of question–query pairs from the TREQS dataset.

Natural Language Question	SQL Query
What are the details of male patients admitted in 2020?	`SELECT * FROM demographics WHERE gender='M' AND admission_year=2020;`
List all laboratory tests with abnormal results for patient ID 12345.	`SELECT test_name, result FROM lab_tests WHERE patient_id=12345 AND abnormal=1;`
Show prescriptions of antibiotics for patients under 18 years old.	`SELECT * FROM prescriptions WHERE drug_type='antibiotic' AND age<18;`
Retrieve all diagnoses for patients who underwent surgery in June 2021.	`SELECT diagnoses FROM procedures JOIN diagnoses ON procedures.patient_id = diagnoses.patient_id WHERE procedures.date BETWEEN "1 June 2021" AND "30 June 2021" AND procedures.type='surgery';`

3.2. Language Models Under Test

Table 2 provides an overview of the language models used in this study, highlighting their capabilities, parameter sizes, architectural designs, and whether they are commercial or open-source. The models studied include LLaMA 3.3-70B, Mixtral 8x22B, Gemini-1.5, Claude 3.5, GPT-4o, GPT-4o-Mini, and Qwen-2.5-72b.

The models vary widely in terms of their parameter counts and design. LLaMA 3.3-70B, for instance, is an open-source model with 70 billion parameters known for its flexibility and efficiency in token usage. Mixtral 8x22B adopts an ensemble-based approach with 176 billion parameters, leveraging multiple smaller models to enhance scalability.

Table 2. Overview of models used in the study.

Model (Commercial)	Parameter Count	Architecture and Key Features
LLaMA 3.3-70B (No)	70 Billion	Transformer decoder; open-source, highly tunable, efficient token usage for large datasets.
Mixtral 8x22B (No)	141 Billion (Ensemble)	Ensemble transformer; designed for scalability, optimized for distributed tasks with multiple smaller models.
Gemini-1.5 (Yes)	1.5 Trillion	Transformer decoder; lightweight and cost-effective, ideal for minimal token usage tasks.
Claude 3.5 (Yes)	Proprietary	Transformer decoder; human-aligned, optimized for conversational tasks and intent recognition.
GPT-4o (Yes)	Proprietary	Transformer decoder; high accuracy, general-purpose capabilities, suitable for premium use cases.
GPT-4o-Mini (Yes)	Proprietary	Transformer decoder; compact design, tuned for budget-sensitive tasks while maintaining strong performance.
Qwen-2.5-72B (Yes)	72 Billion	Transformer decoder; specialized expert models for coding and math; structured data handling.

Commercial models, such as Gemini-1.5 and Claude 3.5, are designed to balance performance and cost. Gemini focuses on lightweight applications, while Claude is optimized for human-aligned conversational tasks. GPT-4o is a highly capable commercial model with 175 billion parameters, excelling in accuracy but at a premium cost. GPT-4o-Mini provides a compact alternative that offers similar functionality in a more cost-effective package.

The architectural differences also reflect their intended applications. Transformer encoder–decoder architectures, like Gemini-1.5, are suitable for tasks requiring balanced input–output token usage. In contrast, decoder-only architectures dominate large-scale models such as GPT-4o and LLaMA, emphasizing generative capabilities.

This model diversity ensures that the study captures a broad spectrum of performance, cost, and architectural trade-offs, offering valuable insights into their application in natural language to SQL transformations.

LLaMA 3.3-70B: LLaMA (Large Language Model Meta AI) 3.3-70B represents the latest iteration in Meta's LLaMA series, designed for scalability and efficiency. With 70 billion parameters, it excels in contextual understanding and zero-shot learning tasks. Its applications span conversational AI, code generation, and multilingual text processing [29].

Mixtral 8x22B: Mixtral 8x22B is a cutting-edge sparse mixture-of-experts (SMoE) model with 141 billion parameters, of which 39 billion are active during computation. This architecture achieves high performance with reduced computational requirements and excels in large-scale and complex tasks. Its dynamic parameter activation ensures efficiency and adaptability, making it suitable for research and high-demand industry applications [30].

Gemini 1.5 Pro: Gemini Pro by Google DeepMind integrates advanced natural language understanding with Vision-Text alignment, excelling in video analysis, document intelligence, and dynamic content generation. With a focus on scalability and innovation, it serves the enterprise and research needs effectively [31].

Claude 3.5:Anthropic's Claude 3.5 emphasizes ethical principles and human-aligned AI, featuring mechanisms for reinforcement learning and interpretability. It is ideal for applications in conversational AI, educational tools, and customer support systems, adhering to rigorous safety standards [32].

GPT-4o: OpenAI's GPT-4o (optimized) is a refined version of GPT-4, offering faster processing and efficient fine-tuning for enterprise needs such as adaptive learning and real-time interactivity [33].

GPT-4o-Mini: GPT-4o-Mini delivers the core functionality of GPT-4o in a lightweight format, making it accessible for edge devices and mobile platforms, allowing greater use of advanced AI [34].

Qwen-2.5-72b: Qwen2.5 significantly improves knowledge retention, coding, and mathematical reasoning through specialized expert models. Enhanced instruction-following capabilities allow for the generation of long texts (over 8K tokens) and understanding and outputting of structured data, particularly JSON. The models exhibit increased robustness to diverse prompts, enabling advanced role-play and chatbot condition setting. Qwen2.5 supports contexts up to 128K tokens and multilingual capabilities in 29+ languages, making it a versatile tool for research and applications [35].

3.3. Prompt Structure

Each query was embedded within a standardized prompt designed to condition the models for translating natural language queries into SQL. Prompting, as a technique, frames the task and guides the model's behavior by explicitly defining its objectives and constraints. The structured prompt included the following elements:

- Job description: A clear and concise description of the role of the model, focusing on its task of converting user queries into SQL commands specifically designed for use in medical databases. This contextual setting helps the model align its outputs with the intended domain-specific requirements.
- Instructions: Detailed guidelines that outline the constraints and expectations for SQL generation. These included using SQLite-compatible functions, avoiding placeholders or overly complex expressions, and prioritizing simple, readable SQL solutions. Such explicit instructions reduce ambiguity and improve the relevance of the generated outputs.
- Schema information: A comprehensive database schema description, including table structures, field names, data types, and their purposes. This information equips the model with the necessary context to generate syntactically correct and semantically meaningful SQL queries.
- 1-shot example query: A single example demonstrating a natural language query and its corresponding SQL output. This example serves as a reference point to condition the model's behavior and guides its interpretation of subsequent inputs, particularly in aligning its outputs with the expected format and style.
- Data preview: Sample rows from key tables in the database, providing additional context about the data stored within. This contextual data allows the model to understand the relationships between tables and fields better, improving its ability to generate precise and contextually appropriate queries.

By structuring the prompt this way, we leveraged prompting as a critical method to condition the models to the specific task. This technique minimizes ambiguity and standardizes the interaction across different models, ensuring that the evaluation is consistent and fair. In addition, including domain-specific details, such as schema and data previews, reinforces the alignment of the model output with the requirements of medical database queries.

3.4. Measurements

The generated SQL queries were reviewed to assess their correctness based on predefined criteria, termed *execution accuracy* (EA). EA measures the proportion of queries that successfully achieve the expected output when executed against the database. Execution accuracy (EA) in our study was defined as the comparison between the result set produced by the LLM-generated SQL query and the ground-truth query's result set. This measure

is objective in that it strictly checks whether the generated query returns the same output as the expected query when executed against the database. A query is considered correct under EA if it satisfies all the following conditions:

- Syntactic validity: The generated SQL query must be syntactically correct and executable in an SQLite environment without errors.
- Semantic correctness: The query must return results that match the expected output for the given natural language query.
- Alignment with constraints: The query must adhere to the specified constraints, such as avoiding placeholders or using SQLite-compatible functions.
- Efficiency: While not the primary criterion, the query should avoid unnecessary complexity and redundancy where possible.

While query efficiency in terms of query performance (e.g., execution time and resource usage) could be a qualitative measure, it was not included as a criterion in our EA metric. Our evaluation strictly compared result sets for correctness, avoiding subjectivity.

The accuracy was then calculated as the proportion of queries meeting all these EA criteria out of the total number of queries. The variation in accuracy between the five repetitions of each query was also measured, providing information on the consistency of the model performance.

For cost evaluation, the following factors were considered for each model:

1. Input tokens: The number of tokens in the input prompt, including the query, schema information, instructions, and other contextual details.
2. Output tokens: The number of tokens generated by the model as part of the SQL output.
3. Pricing structure: The pricing structure specific to each model is typically defined as a cost per input and output token.

The total cost per query was calculated as the sum of the expenses associated with input and output tokens. This approach enabled a detailed comparison of each model's cost-efficiency, complementing the accuracy and consistency evaluations.

3.5. Ablation Study

An ablation study was conducted to assess the impact of two key components in the prompt: the data preview section and the inclusion of a pair of question–answer pair samples. The study systematically included or excluded these components to assess their individual and combined contributions to accuracy, creating four experimental configurations.

1. Both data preview and example question–answer are included.
2. Only the data preview is included.
3. Only the example question–answer is included.
4. Neither the data preview nor the example question–answer is included.

This setup enabled the evaluation of the accuracy and consistency of SQL generation across configurations and the identification of the significance of each component in improving model performance. Since both the data preview and the example question–answer increase the number of input tokens, their combined and individual contributions were analyzed to determine whether improvements in performance justified the additional cost. The study provided insights into the optimal prompt structure by highlighting the trade-offs between accuracy, consistency, and cost for each configuration.

4. Results

This section presents the study findings, focusing on the performance, consistency, and cost-efficiency of the evaluated language models in the context of natural language

to SQL translation for medical databases. The results highlight the models' execution accuracy (EA) across 5000 queries per model, their consistency in repeated runs, and the associated computational costs. In addition, the ablation study's results provide insight into the contribution of prompt components, such as data previews and example queries, to model performance. These findings comprehensively understand the trade-offs between technical accuracy, operational reliability, and economic viability. Some example outputs for each model are listed in Appendix B.

4.1. Performance and Efficiency

The models demonstrated varying levels of accuracy, efficiency, and consistency. The cost per query is determined by the number of tokens processed, weighted by model-specific pricing for input and output tokens. Specifically, the total cost is calculated as follows:

$$\text{Total Cost} = (\text{Tokens}_{\text{in}} \times \text{Price}_{\text{token in}}) + (\text{Tokens}_{\text{out}} \times \text{Price}_{\text{token out}}).$$

While Table 3 presents the average number of input and output tokens per model, Table 4 accounts for the pricing differences across models. The Figures 1 and 2 below further illustrate these variations, demonstrating that models with lower token usage do not necessarily have the lowest cost, as token pricing varies significantly.

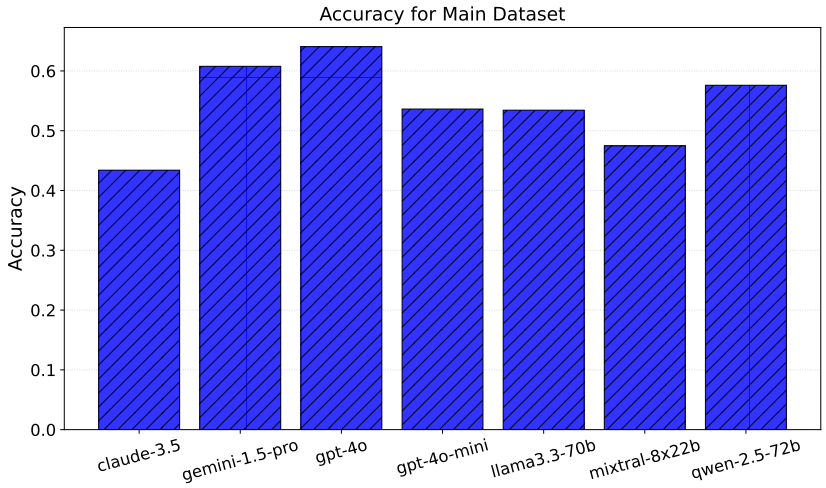

Figure 1. Model comparison with overall execution accuracy (EA).

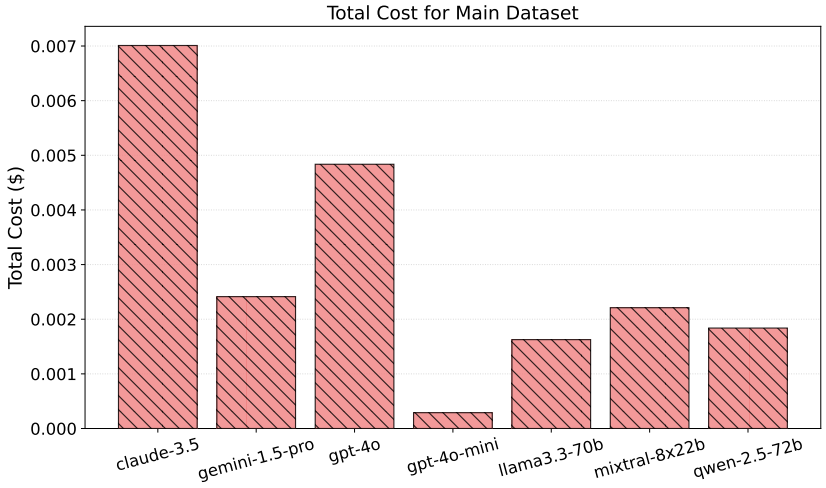

Figure 2. Model comparison with regards to token cost per query.

`GPT-4o` achieved the highest execution accuracy, with a mean of 0.641, while `Gemini-1.5-pro` followed with 0.608. Both models exhibited strong capabilities in correctly transforming inputs into SQL. In contrast, `Claude-3.5` achieved the lowest execution accuracy at 0.434, with lower variability, indicating consistent but suboptimal performance.

Token efficiency was another key factor, with `Gemini-1.5-pro` handling the minor average input token count (1716 tokens) and producing a concise SQL output. `Mixtral-8x22b`, on the other hand, required significantly more input tokens (2365 tokens on average), reflecting potentially higher computational requirements. The output token count in all models was much smaller, with `Claude-3.5` generating the most compact output (70 tokens).

Table 3. Performance and variability of models for natural language to SQL transformation.

Model	Accuracy (%)	Average Tokens In	Average Tokens Out
Claude-3.5	43.4 ± 0.1	1986.052	70.085
Gemini-1.5-pro	60.8 ± 0.4	1716.036	53.653
GPT-4o	64.1 ± 0.6	1710.875	55.880
GPT-4o-mini	53.6 ± 0.3	1710.875	55.155
Llama3.3-70b	53.4 ± 0.2	1747.020	60.461
Mixtral-8x22b	47.5 ± 0.3	2365.230	91.056
Qwen-2.5-72b	57.6 ± 0.0	1981.154	61.747

Table 4. Cost analysis of models for natural language to SQL transformation.

Model	Price/1M Input Tokens ($)	Price/1M Output Tokens ($)	Price per Query ($)
Claude-3.5	3.00	15.00	0.00701
Gemini-1.5-pro	1.25	5.00	0.00241
GPT-4o	2.50	10.00	0.00484
GPT-4o-mini	0.15	0.60	0.00029
Llama3.3-70b	0.90	0.90	0.00163
Mixtral-8x22b	0.90	0.90	0.00221
Qwen-2.5-72b	0.90	0.90	0.00184

4.2. Cost-Effectiveness and Trade-Offs

The Pareto front is a method used to identify optimal trade-offs between competing objectives: in this case, accuracy and cost. A model is considered Pareto-optimal if no other model performs better in both accuracy and cost simultaneously. This means that improving one aspect (e.g., accuracy) would come at the expense of the other (e.g., cost). Models on the Pareto front represent the best possible trade-offs in terms of the available options.

Based on the Pareto front analysis in Figure 3, the optimal model selection depends on the desired balance between accuracy and cost. For scenarios prioritizing minimal cost while maintaining reasonable performance, **GPT-4o-mini** emerges as a compelling choice, offering the lowest cost per query on the Pareto front. However, if accuracy is the primary criterion, **GPT-4o** provides the highest accuracy among the analyzed models, although at a moderately higher cost. **Gemini-1.5-pro** strikes a balance between the two extremes, offering a favorable trade-off between cost and performance. **Qwen-2.5-72b**, the only open-source option on the Pareto front, provides a competitive middle ground, with slightly higher cost but improved accuracy compared to GPT-4o-mini. Thus, the choice of model depends on the specific requirements of the use case, such as budget constraints, open-source preference, or the criticality of high accuracy in SQL query generation.

In the medical domain, where accuracy can directly impact patient care, the higher cost of `GPT-4o` or `Gemini-1.5-pro` may be justified for critical applications. However, for large-scale noncritical deployments, such as preprocessing queries for exploratory

analysis, the cost-efficiency of `GPT-4o-mini` or the open-source nature of `Qwen-2.5-72b` could be advantageous.

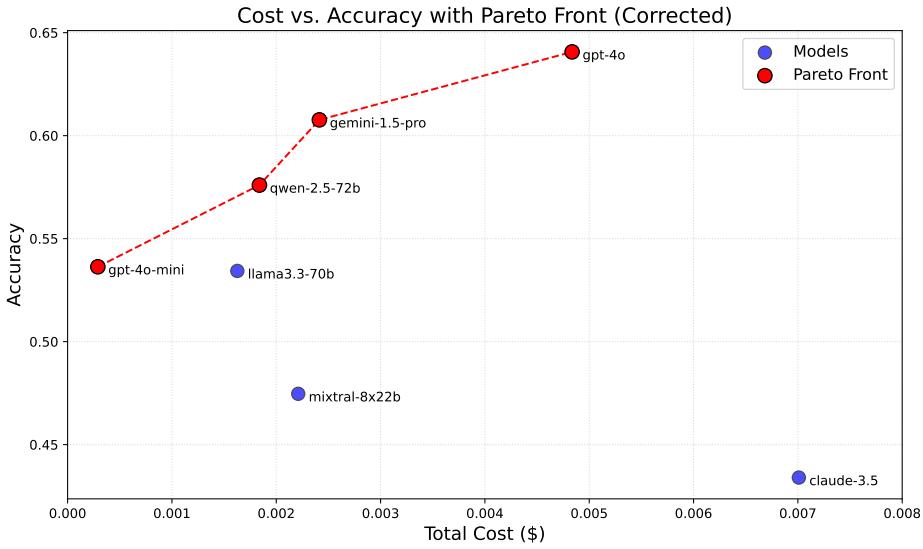

Figure 3. Model comparison with a Pareto front.

4.3. Commercial vs. Open-Source Considerations

Ethical considerations regarding proprietary (commercial) versus open-source models are vital in medicine. Commercial models such as `GPT-4o` and `Gemini-1.5-pro` offer high performance but are associated with recurring costs and potential data governance concerns. Their proprietary nature may limit transparency, which is critical in medical applications, where auditability and reproducibility are often regulatory requirements.

Open-source models like `Mixtral-8x22b` and `Qwen-2.5-72b` provide greater transparency and adaptability, potentially aligning better with medical ethics and compliance standards. However, their slightly lower accuracy and performance may require compensatory measures, such as additional validation steps or integration with domain-specific heuristics, to meet medical-grade quality benchmarks.

4.4. Recommendations

The evaluation highlights that `GPT-4o` is the most accurate model, making it ideal for tasks where performance is critical and cost is less of a concern. `Gemini-1.5-pro` provides a robust balance between accuracy and cost-effectiveness, while `GPT-4o-mini` emerges as the most economical option for large-scale applications requiring moderate accuracy. Notably, `Qwen-2.5-72b`, the only open-source model on the Pareto front, offers a compelling middle ground for those prioritizing transparency and adaptability.

The choice between commercial and open-source models for medical applications depends on the specific use case. High-stakes scenarios, such as clinical decision support, warrant investment in highly accurate commercial models like `GPT-4o`. Conversely, open-source solutions like `Qwen-2.5-72b` should be preferred in research and for systems where transparency, customization, and adherence to open-source principles are prioritized. Ultimately, model selection should be informed by assessing the trade-offs between accuracy, cost, and compliance with ethical and regulatory standards.

5. Prompt Structure Ablation Study

This ablation study investigates the impact of two key prompt design strategies, *one-shot examples* and *sample table data*, on the performance of NL2SQL tasks across multiple large language models (LLMs). A one-shot example consists of a complete input–output

pair, where the input is a natural language query and the output is its corresponding SQL query, included in the prompt to provide contextual guidance. The sample table data consist of five rows per table, presented in a textual CSV-like format, to help the model better understand the schema and structure of the database. This study aims to quantify the changes in model accuracy and token cost introduced by these strategies and evaluate their trade-offs under different configurations. The full prompt structure is provided in Appendix A.

5.1. Table Data Samples in Prompt

Including data samples from the tables in the input prompt resulted in substantial cost increases across all models, as shown in Figure 4. Cost increases ranged from 81.9% (*Claude-3.5*) to 149.5% (*Qwen-2.5-72b*), demonstrating the significant token overhead associated with this approach. However, the performance improvements were inconsistent across models. High-capacity models such as *Qwen-2.5-72b* and *GPT-4o* achieved notable accuracy gains of 26.2% and 21.4%, respectively. In contrast, *Gemini-1.5-pro* exhibited a slight decline in performance (−2.5%), and *Llama3.3-70b* showed a minimal improvement of just 2.1%.

The performance drop of *Gemini-1.5-pro* when provided with database examples may be due to overfitting, misinterpreting patterns, increased input complexity, or prompt sensitivity. The model might have prioritized example replication over generalization, leading to incorrect SQL. Additionally, longer inputs could have diluted focus, and formatting differences may have influenced interpretation.

Models such as *Claude-3.5* and *GPT-4o-mini* achieved moderate performance increases (15.9% and 17.7%, respectively). Still, the high-cost overhead suggests that the inclusion of table samples in prompts should be carefully evaluated based on the specific task requirements and budget constraints.

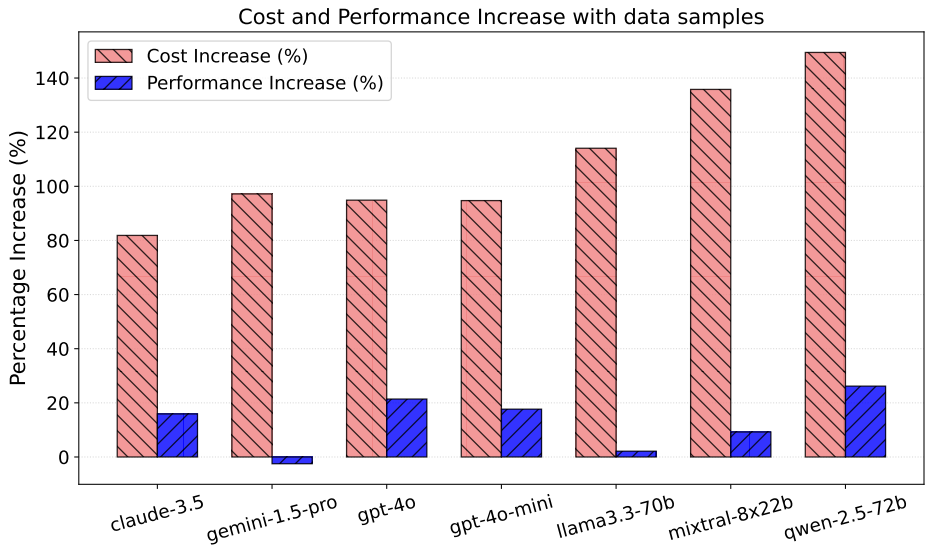

Figure 4. Cost and accuracy increase by model when using a prompt with a data example for each table.

Recommendation: Table data samples should only be included in prompts when the task requires detailed context to resolve complex SQL queries. For more straightforward questions or when using cost-sensitive models, omitting table samples is advisable. High-capacity models like *Qwen-2.5-72b* and *GPT-4o* can leverage the additional context effectively, making them suitable for this configuration. However, for models like *Gemini-*

1.5-pro and *Llama3.3-70b*, the high cost and limited or negative performance impact make this approach less practical.

5.2. One-Shot Examples in Prompt

Adding a single example query (one shot) to the prompt demonstrated a transparent performance boost in most models, as illustrated in Figure 5. The relative cost increase for one-shot examples remained modest, ranging from 5.9% for *Claude-3.5* to 12.7% for *Mixtral-8x22b*. Thus, including one-shot examples is a cost-effective strategy to improve accuracy.

The most significant performance gains were observed in high-capacity models such as *Llama3.3-70b* (75%) and *Qwen-2.5-72b* (85%), indicating that these models are particularly adept at taking advantage of contextual information provided by the example. Similarly, *GPT-4o-mini* achieved a notable 69.9% increase in precision with only a 6.7% cost increase, showcasing its efficiency in adapting to single-shot contexts. In contrast, lower-capacity models such as *Mixtral-8x22b* showed limited performance improvements (12.2%) despite incurring the highest cost increase (12.7%).

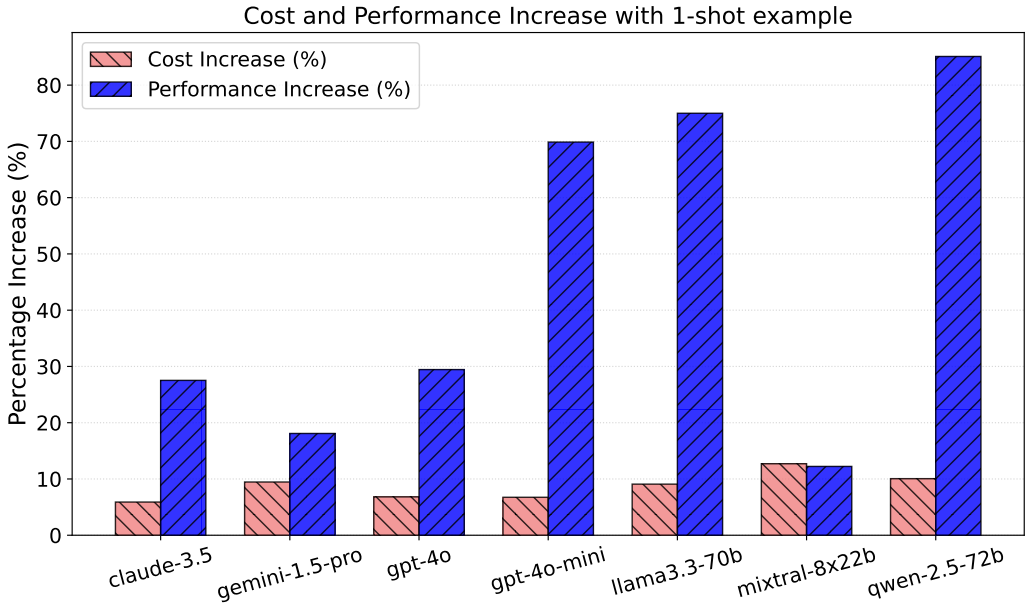

Figure 5. Cost and accuracy increase by model when using a prompt with a one-shot example.

Recommendation: The use of one-shot examples is strongly recommended for most tasks, especially when deploying high-capacity models like *Llama3.3-70b* and *Qwen-2.5-72b*, which demonstrate significant accuracy improvements with relatively low-cost overhead. For cost-sensitive scenarios or using less capable models like *Mixtral-8x22b*, omitting the one-shot example may provide better cost-performance trade-offs.

5.3. Full Prompt with Both

Combining one-shot examples and table data samples in the prompt resulted in the highest accuracy improvements across most models, as shown in Figure 6. However, this configuration also incurred the highest token costs, with cost increases ranging from 88.4% (*Claude-3.5*) to 159.0% (*Qwen-2.5-72b*). Despite the significant cost overhead, this approach yielded substantial performance gains in high-capacity models, particularly *GPT-4o-mini* (115.4%), *Qwen-2.5-72b* (109.5%), and *Llama3.3-70b* (90.8%).

Interestingly, more minor improvements were observed in models such as *Gemini-1.5-pro* (26.3%) and *Mixtral-8x22b* (29.0%), which also experienced disproportionately high-cost increases of 109.1% and 149.3%, respectively.

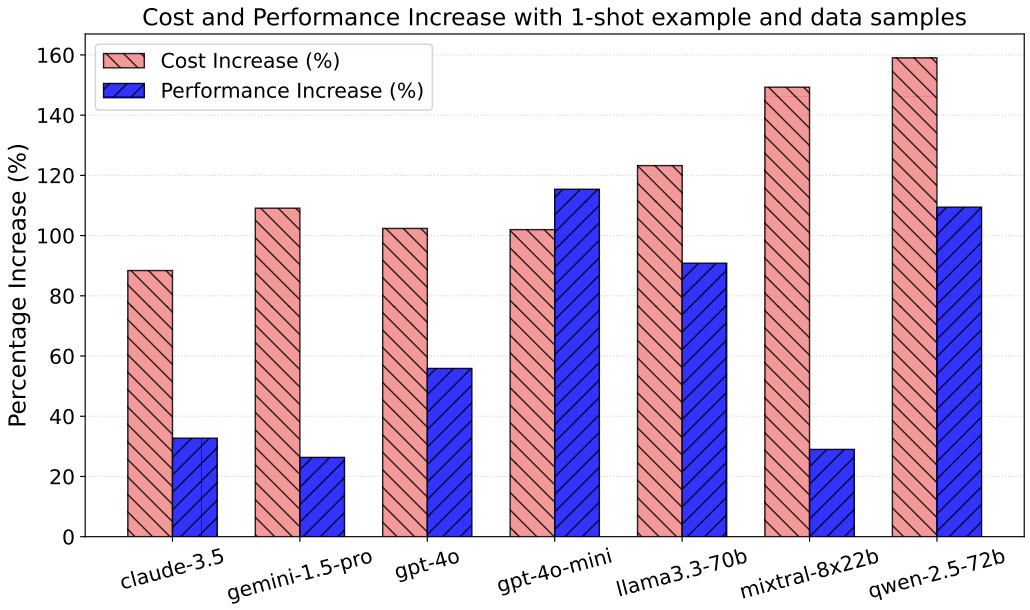

Figure 6. Cost and accuracy increase by model when using a prompt with both a one-shot example and table samples.

Recommendation: The complete prompt configuration should be reserved for tasks where accuracy is paramount and token cost is not a constraint. High-capacity models like *GPT-4o-mini*, *Qwen-2.5-72b*, and *Llama3.3-70b* benefit the most from this approach and can justify the increased token cost. For budget-sensitive applications or models with limited capacity, using only one of the two components, either table data samples or one-shot examples, provides a more cost-effective alternative.

5.4. Ablation Study Conclusions

The ablation study highlights the trade-offs between token cost and accuracy when incorporating table data samples and one-shot examples in prompts. While table data samples significantly increase token costs (often exceeding 100%), their performance gains are inconsistent and model-dependent, making them suitable only for tasks requiring detailed context in high-capacity models like *Qwen-2.5-72b*. In contrast, one-shot examples consistently deliver substantial accuracy improvements with modest cost increases (typically below 13%), particularly benefiting high-capacity models like *Llama3.3-70b*. Combining both approaches achieves the highest performance gains but incurs steep token costs, making it viable only for accuracy-critical tasks. For cost-sensitive applications, selective inclusion of table data samples or one-shot examples provides a balanced trade-off between cost and accuracy.

6. Limitations and Future Directions

Generating accurate SQL queries from natural language in the medical domain requires models to handle precise terminology, such as diseases and procedures defined by official ICD taxonomies. Although the evaluated language models showed proficiency in syntactically correct SQL generation, they struggled to align user queries with the precise medical terms necessary for accurate data retrieval. This section examines these challenges,

explores a potential solution using retrieval-augmented generation (RAG), and proposes directions for future research.

6.1. Challenges in Medical Name Resolution

A recurring issue observed during the evaluation was the model's difficulty mapping user-specified medical terms to their correct counterparts in ICD taxonomies. For example, a query about "*heart failure*" often failed to align with the official ICD9 code `428.0` ("*Congestive heart failure, unspecified*"). Instead, the models produced overly generic queries or generated outputs incompatible with the database schema. These errors highlight the disconnect between natural language expressions and the structured nomenclature required for SQL queries in medical databases.

The underlying challenges arise from two main factors. First, language models rely on pre-trained knowledge and lack direct access to comprehensive and updated medical taxonomies at inference time. Second, ambiguity in natural language adds complexity, as colloquial terms used by users often differ from the official medical terminology. For example, a term like "*heart attack*" requires resolution to "*myocardial infarction*" (`ICD9: 410.x`) to generate a valid query.

6.2. Potential of Retrieval-Augmented Generation (RAG)

Retrieval-augmented generation (RAG) offers a promising approach to address these challenges by enhancing language models with real-time access to external knowledge sources. A RAG pipeline combines a retrieval system with a generative language model, allowing it to dynamically fetch relevant information from structured knowledge bases, such as ICD taxonomies, and integrate it into its outputs.

In the context of SQL query generation, an RAG workflow would begin by analyzing the user's natural language query to identify key terms requiring resolution. The retrieval system then queries an external medical terminology database, fetching the appropriate ICD codes or related taxonomy entries. This retrieved information would inform the generation process, ensuring accurate alignment between the user's intent and the SQL schema.

This approach has several advantages. Grounding outputs in real-time knowledge retrieval reduces errors caused by the limitations of pre-trained knowledge. It also enables dynamic adaptation to updates in ICD taxonomies, ensuring consistent query accuracy over time. In addition, RAG systems can help resolve ambiguous queries by using contextual information to select the most relevant terms.

6.3. Future Directions

Implementing and evaluating RAG in generating SQL queries for medical databases opens several avenues for future research. Key areas include developing robust retrieval mechanisms that interface with ICD taxonomies and medical ontologies, ensuring the system can fetch and interpret relevant terms efficiently. Establishing metrics to evaluate the accuracy and reliability of term resolution and its overall impact on query correctness is also critical. Furthermore, conducting user studies can help refine the system's ability to disambiguate complex queries and adapt to real-world requirements. While the effort to translate natural language into medical SQL queries is commendable and beneficial for laypersons, it is likely that medical staff will derive even greater benefits from such systems once they are familiar with the domain-specific jargon (e.g., ICD taxonomies). While the study focused on general natural language to SQL translation, we acknowledge that medical professionals familiar with domain-specific jargon may benefit from a more specialized system. This has been noted as a potential direction for future research, where evaluating LLMs on translating structured medical queries into SQL could provide a more realistic assessment of their utility in clinical settings.

Additionally, expanding the dataset is a crucial aspect of future work. The limitation of using 1,000 queries from the TREQS dataset is acknowledged, and efforts to address this through the incorporation of additional datasets and the generation of synthetic queries are planned to enhance robustness and generalizability. These enhancements will provide a more comprehensive evaluation framework for SQL query generation in medical databases.

The potential of DeepSeek-R1 is also recognized, and its inclusion in future work is planned. Evaluating its performance in comparison with other state-of-the-art models will contribute to a deeper understanding of model capabilities and trade-offs in the context of SQL query generation for healthcare applications.

By addressing these challenges, retrieval-augmented generation, coupled with dataset expansion and broader model evaluations, has the potential to significantly improve the accuracy and reliability of SQL queries generated for medical databases.

6.4. Threats to Validity

The TREQS dataset, while derived from the widely documented and widely used MIMIC-III database, presents potential limitations that must be addressed to ensure generalizability and robustness. Although the dataset includes a curated collection of question–query pairs, its reliance on manually rephrased questions introduces the risk of linguistic bias. These rephrasings may inadvertently overfit the models to specific phrasing patterns, limiting their ability to generalize to diverse, real-world queries. In addition, the dataset focuses primarily on English queries, raising concerns about its applicability to multilingual healthcare systems commonly found in global contexts. Expanding the dataset to include queries in multiple languages and integrating colloquial and domain-specific medical terms would significantly enhance its relevance and usability in diverse healthcare settings.

In addition, reliance on proprietary models, such as GPT-4o and Claude 3.5, poses reproducibility and external validation challenges. These models operate as black boxes with limited access to training data, fine-tuning processes, and architectural specifics. Although we have included open-source models such as LLaMA 3.3-70B and Qwen-2.5-72B in our evaluation, the comparative emphasis on proprietary solutions could hinder broader validation efforts.

Another limitation arises from the reliance on structured tables derived from MIMIC-III, which may not fully encapsulate the complexities of real-world electronic health record (EHR) systems. Healthcare databases often exhibit variability in schema design, naming conventions, and data formats, necessitating further validation of models on additional datasets representing broader medical domains. Addressing these issues would involve developing benchmarking datasets that capture the heterogeneity and nuances of healthcare queries, thus ensuring that the proposed models perform effectively in diverse operational settings.

7. Conclusions

This research highlights the transformative potential of large language models (LLMs) in translating natural language queries into executable SQL statements, with a specific focus on the healthcare domain. Through a comprehensive evaluation of state-of-the-art models, including GPT-4o, Gemini-1.5-pro, LLaMA 3.3-70B, and others, key trade-offs between accuracy, cost, and consistency were identified. GPT-4o achieved the highest accuracy (64.1%), making it the most suitable for high-stakes applications, while GPT-4o-mini offered the most cost-effective solution (USD 0.00029 per query) for large-scale deployments requiring moderate accuracy (53.6%).

The study further demonstrated the impact of tailored prompt engineering, with techniques such as one-shot examples and data previews significantly improving model performance at different cost levels. Additionally, challenges related to medical terminology processing and schema complexity highlighted the need for domain-specific adaptations. Future research directions include integrating retrieval-augmented generation (RAG) pipelines, improving the resolution of ICD taxonomies, and expanding dataset diversity to enhance robustness and generalizability.

From a cost-effectiveness perspective, the Pareto front revealed distinct model advantages: while GPT-4o provides the highest accuracy at a moderate cost (USD 0.00484 per query), Gemini-1.5-pro presents a well-balanced alternative between accuracy (60.8%) and affordability (USD 0.00241 per query). Qwen-2.5-72B, the only open-source model on the Pareto front, offers a transparent and adaptable solution, albeit at a slightly higher cost.

The potential of DeepSeek-R1 is also recognized and should be included in future work. The current limitation of using 1000 queries from the TREQS dataset should also be addressed by incorporating additional data sets and the generation of synthetic queries to enhance robustness and generalizability.

Ultimately, while LLMs offer significant advancements in democratizing access to medical databases, their practical implementation requires careful consideration of accuracy, cost, and regulatory compliance. This work lays the foundation for further improvements in SQL query generation systems, contributing to enhanced clinical decision-making and medical research capabilities.

Author Contributions: Conceptualization, N.T. and I.L.; methodology, R.Š.; software, N.T.; validation, N.T. and I.L.; formal analysis, R.Š.; investigation, N.T. and I.L.; resources, N.T.; data curation, R.Š.; writing—original draft preparation, N.T. and I.L.; writing—review and editing, N.T. and I.L.; visualization, R.Š.; supervision, N.T. and I.L.; project administration, I.L.; funding acquisition, I.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This research was (partly) supported by the "European Digital Innovation Hub Adriatic Croatia (EDIH Adria) (project no. 101083838)" under the European Commission's Digital Europe Programme, SPIN project "INFOBIP Konverzacijski Order Management (IP.1.1.03.0120)", SPIN project "Projektiranje i razvoj nove generacije laboratorijskog informacijskog sustava (iLIS)" (IP.1.1.03.0158), and the FIPU project "Sustav za modeliranje i provedbu poslovnih procesa u heterogenom i decentraliziranom računalnom sustavu".

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Prompt Structure

This appendix provides the structured prompt used in the ablation study. The prompt is divided into different components, including the job description, the example query, the schema, and the data preview.

Job description. You are an advanced language model specialized in the generation of SQL queries for medical databases. Your role is to convert user queries into SQL statements, considering the context of medical data. Given an input question, first create a syntactically correct SQL query to run, returning only the query with no additional comments.

Guidelines:

- Generate accurate SQL queries aligned with the user's intent.
- Return SQL statements without explanations or comments.

- Prefix all selected or counted columns with the table name and wrap them in double quotes.
- Do not use placeholders (e.g., `{{table_name}}`, `{{column_name}}`).
- Wrap SQL code in ‘‘‘`sql` blocks.
- Assume an SQLite database and avoid functions unavailable there.
- Prefer simple solutions with minimal function calls or redundant boolean operations.
- Do not use table aliases.
- When matching strings, match them in lowercase using `LOWER()`.

Appendix A.1. Section `[Example]` for Ablation Study

Example Query:

```
User: Find all patients diagnosed with ICD9 9951
```

Expected Output:

```
SELECT COUNT ( DISTINCT DEMOGRAPHIC."SUBJECT_ID" )
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC.HADM_ID = DIAGNOSES.HADM_ID
WHERE DIAGNOSES."ICD9_CODE" = "9951"
```

Appendix A.2. Section `[Schema]` for Ablation Study

The following schema defines the database structure used in query generation:

```
-- demographic - patient information, count by demographic.subject_id
CREATE TABLE ‘‘demographic’’ (
‘‘SUBJECT_ID’’ INTEGER,
‘‘HADM_ID’’ INTEGER,
‘‘NAME’’ TEXT,
‘‘MARITAL_STATUS’’ TEXT,
‘‘AGE’’ INTEGER,
‘‘DOB’’ TEXT,
‘‘GENDER’’ TEXT,
‘‘LANGUAGE’’ TEXT,
‘‘RELIGION’’ TEXT,
‘‘ADMISSION_TYPE’’ TEXT,
‘‘DAYS_STAY’’ INTEGER,
‘‘INSURANCE’’ TEXT,
‘‘ETHNICITY’’ TEXT,
‘‘EXPIRE_FLAG’’ INTEGER,
‘‘ADMISSION_LOCATION’’ TEXT,
‘‘DISCHARGE_LOCATION’’ TEXT,
‘‘DIAGNOSIS’’ TEXT,
‘‘DOD’’ TEXT,
‘‘DOB_YEAR’’ INTEGER,
‘‘DOD_YEAR’’ INTEGER,
‘‘ADMITTIME’’ TEXT,
‘‘DISCHTIME’’ TEXT,
‘‘ADMITYEAR’’ INTEGER
);

-- diagnoses: patient diagnoses, use for queries related
```

```
-- to patient conditions
CREATE TABLE ‘‘diagnoses’’ (
‘‘SUBJECT_ID’’ INTEGER,
‘‘HADM_ID’’ INTEGER,
‘‘ICD9_CODE’’ TEXT,
‘‘SHORT_TITLE’’ TEXT,
‘‘LONG_TITLE’’ TEXT
);

-- lab: patient lab results
CREATE TABLE ‘‘lab’’ (
‘‘SUBJECT_ID’’ INTEGER,
‘‘HADM_ID’’ INTEGER,
‘‘ITEMID’’ INTEGER,
‘‘CHARTTIME’’ TEXT,
‘‘FLAG’’ TEXT,
‘‘VALUE_UNIT’’ TEXT,
‘‘LABEL’’ TEXT,
‘‘FLUID’’ TEXT,
‘‘CATEGORY’’ TEXT
);

-- prescriptions: patient prescriptions
CREATE TABLE ‘‘prescriptions’’ (
‘‘SUBJECT_ID’’ INTEGER,
‘‘HADM_ID’’ INTEGER,
‘‘ICUSTAY_ID’’ REAL,
‘‘DRUG_TYPE’’ TEXT,
‘‘DRUG’’ TEXT,
‘‘FORMULARY_DRUG_CD’’ TEXT,
‘‘ROUTE’’ TEXT,
‘‘DRUG_DOSE’’ TEXT
);

-- procedures: medical procedures performed on patients
CREATE TABLE ‘‘procedures’’ (
‘‘SUBJECT_ID’’ INTEGER,
‘‘HADM_ID’’ INTEGER,
‘‘ICD9_CODE’’ INTEGER,
‘‘SHORT_TITLE’’ TEXT,
‘‘LONG_TITLE’’ TEXT
);
```

Appendix A.3. Section [Data_Preview] for Ablation Study

TABLE: DEMOGRAPHIC

```
SUBJECT_ID,HADM_ID,NAME,MARITAL_STATUS,AGE,DOB,GENDER,LANGUAGE,
RELIGION,ADMISSION_TYPE,DAYS_STAY,INSURANCE,ETHNICITY,EXPIRE_FLAG,
ADMISSION_LOCATION,DISCHARGE_LOCATION,DIAGNOSIS,DOD,DOB_YEAR,DOD_YEAR,
ADMITTIME,DISCHTIME,ADMITYEAR
10026,132345,Jane Doe,SEPARATED,70,2094-03-05,F,,CATHOLIC,
```

```
EMERGENCY,8,Medicare,BLACK/AFRICAN AMERICAN,0,EMERGENCY ROOM ADMIT,
HOME HEALTH CARE,SEPSIS,2165-08-12,2094,2165,2164-10-23 21:09:00,
2164-11-01 17:15:00,2164
```

TABLE: DIAGNOSES

```
SUBJECT_ID,HADM_ID,ICD9_CODE,SHORT_TITLE,LONG_TITLE
10056,100375,2761,Hyposmolality,Hyposmolality and/or hyponatremia
10056,100375,4280,CHF NOS,Congestive heart failure, unspecified
```

TABLE: LAB

```
SUBJECT_ID,HADM_ID,ITEMID,CHARTTIME,FLAG,VALUE_UNIT,LABEL,...
10006,142345,50813,2164-10-23 17:33:00,abnormal,2.4mmol/L,Lactate,...
10006,142345,50861,2164-10-23 17:38:00,,7IU/L,Alanine (ALT),...
10006,142345,50862,2164-10-23 17:38:00,,3.8g/dL,Albumin,...
```

TABLE: PROCEDURES

```
SUBJECT_ID,HADM_ID,ICD9_CODE,SHORT_TITLE,LONG_TITLE
10056,100375,9904,Packed cell transfusion,Transfusion of packed cells
42430,100969,3891,Arterial catheterization,Arterial catheterization
42430,100969,9671,Cont inv mec ven <96 hrs,Continuous invasive
mechanical ventilation for less than 96 consecutive hours
```

TABLE: PRESCRIPTIONS

```
SUBJECT_ID,HADM_ID,ICUSTAY_ID,DRUG_TYPE,DRUG,FORMULARY_DRUG_CD,...
42458,159647,,MAIN,Pneumococcal Vac Polyvalent,PNEU25I,...
42458,159647,,MAIN,Bisacodyl,BISA5,...
42458,159647,,MAIN,Bisacodyl,BISA10R,...
```

Appendix B. Examples of Prompted SQL Queries

This appendix presents concrete examples of natural language questions, the expected SQL query (ground truth), and the corresponding outputs generated by different LLMs.

Appendix B.1. Case Number: ac54a82fadd7f1ed60b5f097dcccc668

Question: Calculate the days of hospital stay and lab test category for patient with patient id 18372.

Ground Truth SQL:

```
SELECT DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
FROM DEMOGRAPHIC
INNER JOIN LAB ON DEMOGRAPHIC.HADM_ID = LAB.HADM_ID
WHERE DEMOGRAPHIC."SUBJECT_ID" = "18372"
```

Model Responses:

- **Llama3.3-70b**:

```
SELECT DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
FROM DEMOGRAPHIC
INNER JOIN LAB ON DEMOGRAPHIC."SUBJECT_ID" = LAB."SUBJECT_ID"
WHERE DEMOGRAPHIC."SUBJECT_ID" = 18372
```

- **Mixtral-8x22b**:

```
SELECT DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
FROM DEMOGRAPHIC
INNER JOIN LAB ON DEMOGRAPHIC."SUBJECT_ID" = LAB."SUBJECT_ID"
WHERE DEMOGRAPHIC."SUBJECT_ID" = 18372
```

- **Gemini 1.5-pro**:

```
SELECT DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
FROM DEMOGRAPHIC
INNER JOIN LAB ON DEMOGRAPHIC."SUBJECT_ID" = LAB."SUBJECT_ID"
WHERE DEMOGRAPHIC."SUBJECT_ID" = 18372
```

- **Qwen-2.5-72B**:

```
SELECT "demographic"."DAYS_STAY", "lab"."CATEGORY"
FROM "demographic"
INNER JOIN "lab" ON "demographic"."HADM_ID" = "lab"."HADM_ID"
WHERE "demographic"."SUBJECT_ID" = 18372
```

- **Claude-3.5**:

```
SELECT DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
FROM DEMOGRAPHIC
LEFT JOIN LAB ON DEMOGRAPHIC."SUBJECT_ID" = LAB."SUBJECT_ID"
WHERE DEMOGRAPHIC."SUBJECT_ID" = 18372
GROUP BY DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
```

- **GPT-4o**:

```
SELECT DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
FROM DEMOGRAPHIC
INNER JOIN LAB ON DEMOGRAPHIC.HADM_ID = LAB.HADM_ID
WHERE DEMOGRAPHIC."SUBJECT_ID" = 18372
```

- **GPT-4o-mini**:

```
SELECT DEMOGRAPHIC."DAYS_STAY", LAB."CATEGORY"
FROM DEMOGRAPHIC
LEFT JOIN LAB ON DEMOGRAPHIC."HADM_ID" = LAB."HADM_ID"
WHERE DEMOGRAPHIC."SUBJECT_ID" = 18372
```

Appendix B.2. Case Number: a90ecf2fa6bc7c4af1d8fc44eb09a881

Question: Find the number of patients who are married and have been diagnosed with pyelonephritis, unspecified.

Ground Truth SQL:

```
SELECT COUNT ( DISTINCT DEMOGRAPHIC."SUBJECT_ID" )
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC.HADM_ID = DIAGNOSES.HADM_ID
WHERE DEMOGRAPHIC."MARITAL_STATUS" = "MARRIED"
AND DIAGNOSES."LONG_TITLE" = "Pyelonephritis, unspecified"
```

Model Responses:

- **Llama3.3-70b**:

```
SELECT COUNT(DISTINCT DEMOGRAPHIC."SUBJECT_ID")
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC."HADM_ID" = DIAGNOSES."HADM_ID"
WHERE LOWER(DEMOGRAPHIC."MARITAL_STATUS") = 'married'
AND LOWER(DIAGNOSES."SHORT_TITLE") = 'pyelonephritis, unspecified'
```

- **Mixtral-8x22b**:

```
SELECT COUNT(DISTINCT DEMOGRAPHIC."SUBJECT_ID")
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC."SUBJECT_ID" = DIAGNOSES."SUBJECT_ID"
WHERE LOWER(DEMOGRAPHIC."MARITAL_STATUS") = 'married'
AND LOWER(DIAGNOSES."SHORT_TITLE") = 'pyelonephritis, unspecified'
```

- **Gemini 1.5-pro**:

```
SELECT COUNT(DISTINCT DEMOGRAPHIC."SUBJECT_ID")
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC.HADM_ID = DIAGNOSES.HADM_ID
WHERE LOWER(DEMOGRAPHIC."MARITAL_STATUS") = 'married'
AND LOWER(DIAGNOSES."SHORT_TITLE") = 'pyelonephritis, unspecified'
```

- **Qwen-2.5-72B**:

```
SELECT COUNT(DISTINCT "demographic"."SUBJECT_ID")
FROM "demographic"
INNER JOIN "diagnoses" ON "demographic"."HADM_ID" = "diagnoses"."HADM_ID"
WHERE LOWER("demographic"."MARITAL_STATUS") = 'married'
AND LOWER("diagnoses"."SHORT_TITLE") = 'pyelonephritis, unspecified'
```

- **Claude-3.5**:

```
SELECT COUNT(DISTINCT DEMOGRAPHIC."SUBJECT_ID")
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC."HADM_ID" = DIAGNOSES."HADM_ID"
WHERE LOWER(DEMOGRAPHIC."MARITAL_STATUS") = 'married'
AND LOWER(DIAGNOSES."SHORT_TITLE") = 'pyelonephritis nos'
```

- **GPT-4o**:

```
SELECT COUNT(DISTINCT DEMOGRAPHIC."SUBJECT_ID")
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC."HADM_ID" = DIAGNOSES."HADM_ID"
WHERE LOWER(DEMOGRAPHIC."MARITAL_STATUS") = 'married'
AND LOWER(DIAGNOSES."SHORT_TITLE") = 'pyelonephritis, unspecified'
```

- **GPT-4o-mini**:

```
SELECT COUNT ( DISTINCT DEMOGRAPHIC."SUBJECT_ID" )
FROM DEMOGRAPHIC
INNER JOIN DIAGNOSES ON DEMOGRAPHIC.HADM_ID = DIAGNOSES.HADM_ID
WHERE DEMOGRAPHIC."MARITAL_STATUS" = "MARRIED"
AND LOWER(DIAGNOSES."SHORT_TITLE") = "pyelonephritis, unspecified"
```

References

1. Blumenthal, D. Stimulating the adoption of health information technology. *N. Engl. J. Med.* **2009**, *360*, 1477–1479. [CrossRef] [PubMed]
2. Jetley, G.; Zhang, H. Electronic health records in IS research: Quality issues, essential thresholds and remedial actions. *Decis. Support Syst.* **2019**, *126*, 113137. [CrossRef]
3. Huang, K.; Altosaar, J.; Ranganath, R. ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission. *arXiv* **2020**, arXiv:1904.05342.
4. Alsentzer, E.; Murphy, J.R.; Boag, W.; Weng, W.H.; Jin, D.; Naumann, T.; McDermott, M.B.A. Publicly Available Clinical BERT Embeddings. *arXiv* **2019**, arXiv:1904.03323.
5. Singhal, K.; Azizi, S.; Tu, T.; Mahdavi, S.S.; Wei, J.; Chung, H.W.; Scales, N.; Tanwani, A.; Cole-Lewis, H.; Pfohl, S.; et al. Large language models encode clinical knowledge. *Nature* **2023**, *620*, 172–180. [CrossRef]
6. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
7. Ayaz, M.; Pasha, M.F.; Alzahrani, M.Y.; Budiarto, R.; Stiawan, D. The Fast Health Interoperability Resources (FHIR) Standard: Systematic Literature Review of Implementations, Applications, Challenges and Opportunities. *JMIR Med. Inform.* **2021**, *9*, e21929; Erratum in *JMIR Med. Inform.* **2021**, *9*, e32869. [CrossRef]
8. Engelhardt, M. Hitching Healthcare to the Chain: An Introduction to Blockchain Technology in the Healthcare Sector. *Technol. Innov. Manag. Rev.* **2017**, *7*, 22–34. [CrossRef]
9. European Union. *General Data Protection Regulation (GDPR)*; European Union: Brussels, Belgium, 2018.
10. Shanafelt, T.D.; Dyrbye, L.N.; Sinsky, C.; Hasan, O.; Satele, D.; Sloan, J.; West, C.P. Relationship Between Clerical Burden and Characteristics of the Electronic Environment With Physician Burnout and Professional Satisfaction. *Mayo Clin. Proc.* **2016**, *91*, 836–848. [CrossRef] [PubMed]
11. Frost, M.J.; Tran, J.B.; Khatun, F.; Friberg, I.K.; Rodríguez, D.C. What Does It Take to Be an Effective National Steward of Digital Health Integration for Health Systems Strengthening in Low- and Middle-Income Countries? *Glob. Health Sci. Pract.* **2018**, *6*, S18–S28. [CrossRef] [PubMed]
12. Wang, J.; Deng, H.; Liu, B.; Hu, A.; Liang, J.; Fan, L.; Zheng, X.; Wang, T.; Lei, J. Systematic Evaluation of Research Progress on Natural Language Processing in Medicine Over the Past 20 Years: Bibliometric Study on PubMed. *J. Med. Internet Res.* **2020**, *22*, e16816. [CrossRef] [PubMed]
13. Au Yeung, J.; Shek, A.; Searle, T.; Kraljevic, Z.; Dinu, V.; Ratas, M.; Al-Agil, M.; Foy, A.; Rafferty, B.; Oliynyk, V.; et al. Natural language processing data services for healthcare providers. *BMC Med. Inform. Decis. Mak.* **2024**, *24*, 356. [CrossRef] [PubMed]
14. Marshan, A.; AlMutairi, A.N.; Ioannou, A.; Bell, D.; Monaghan, A.; Arzoky, M. MedT5SQL: A transformers-based large language model for text-to-SQL conversion in the healthcare domain. *Front. Big Data* **2024**, *7*, 1371680. [CrossRef] [PubMed]
15. Zhang, W.; Wang, Y.; Song, Y.; Wei, V.J.; Tian, Y.; Qi, Y.; Chan, J.H.; Wong, R.C.W.; Yang, H. Natural Language Interfaces for Tabular Data Querying and Visualization: A Survey. *IEEE Trans. Knowl. Data Eng.* **2023**, *36*, 6699–6718. [CrossRef]
16. Li, F.; Jagadish, H.V. Constructing an Interactive Natural Language Interface for Relational Databases. *Proc. VLDB Endow.* **2014**, *8*, 73–84. [CrossRef]
17. Katsogiannis-Meimarakis, G.; Koutrika, G. A survey on deep learning approaches for text-to-SQL. *VLDB J.* **2023**, *32*, 905–936. [CrossRef]
18. Iacob, R.; Brad, F.; Apostol, E.; Truică, C.O.; Hosu, I.A.; Rebedea, T. Neural Approaches for Natural Language Interfaces to Databases: A Survey. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 8–13 December 2020. [CrossRef]
19. Qin, B.; Hui, B.; Wang, L.; Yang, M.; Li, J.; Li, B.; Geng, R.; Cao, R.; Sun, J.; Si, L.; et al. A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions. *arXiv* **2022**, arXiv:2208.13629. [CrossRef]
20. Jin, Y.; Chandra, M.; Verma, G.; Hu, Y.; De Choudhury, M.; Kumar, S. Better to Ask in English: Cross-Lingual Evaluation of Large Language Models for Healthcare Queries. In Proceedings of the ACM Web Conference 2024, Singapore, 13–17 May 2024; Association for Computing Machinery: New York, NY, USA; pp. 2627–2638. [CrossRef]
21. Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C.H.; Kang, J. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **2020**, *36*, 1234–1240. [CrossRef] [PubMed]
22. Zhu, X.; Li, Q.; Cui, L.; Liu, Y. Large Language Model Enhanced Text-to-SQL Generation: A Survey. *arXiv* **2024**, arXiv:2410.06011.
23. Liu, X.; Shen, S.; Li, B.; Ma, P.; Jiang, R.; Zhang, Y.; Fan, J.; Li, G.; Tang, N.; Luo, Y. A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? *arXiv* **2024**, arXiv:2408.05109.
24. Hong, Z.; Yuan, Z.; Zhang, Q.; Chen, H.; Dong, J.; Huang, F.; Huang, X. Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL. *arXiv* **2024**, arXiv:2406.08426.
25. Yang, R.; Ning, Y.; Keppo, E.; Liu, M.; Hong, C.; Bitterman, D.S.; Ong, J.C.L.; Ting, D.S.W.; Liu, N. Retrieval-augmented generation for generative artificial intelligence in health care. *npj Health Syst.* **2025**, *2*, 2. [CrossRef]

26. Yang, E.; Amar, J.; Lee, J.H.; Kumar, B.; Jia, Y. The Geometry of Queries: Query-Based Innovations in Retrieval-Augmented Generation. *arXiv* **2024**, arXiv:2407.18044.
27. Holzinger, A.; Langs, G.; Denk, D.; Zatloukal, K.; Müller, H. Causability and Explainability of AI in Medicine. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *9*, e1312. [CrossRef] [PubMed]
28. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. *Adv. Neural. Inf. Process. Syst.* **2017**, *30*, 5998–6008.
29. Meta-AI. LLaMA 3.3-70B. 2024. Available online: https://www.llama.com/ (accessed on 11 December 2024).
30. Mistral AI. Mixtral 8x22B: Mixture-of-Experts Model. 2024. Available online: https://mistral.ai/news/mixtral-8x22b (accessed on 11 December 2024).
31. Google DeepMind. Gemini Pro. 2024. Available online: https://deepmind.google/technologies/gemini/pro/ (accessed on 11 December 2024).
32. Anthropic. Claude 3.5. 2024. Available online: https://www.anthropic.com/news/claude-3-5-sonnet (accessed on 11 December 2024).
33. OpenAI. GPT-4o. 2024. Available online: https://openai.com/index/hello-gpt-4o/ (accessed on 11 December 2024).
34. OpenAI. GPT-4o-Mini: Advancing Cost-Efficient Intelligence. 2024. Available online: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/ (accessed on 11 December 2024).
35. QwenLM. Qwen2.5. 2024. Available online: https://github.com/QwenLM/Qwen2.5 (accessed on 11 December 2024).