Reliable Curation of EHR Dataset via Large Language Models under Environmental Constraints

Raymond M. Xiong, Panyu Chen, Tianze Dong, Jian Lu,
Benjamin Goldstein, Danyang Zhuo, and Anru R. Zhang
Duke University

Abstract

Electronic health records (EHRs) are central to modern healthcare delivery and research; yet, many researchers lack the database expertise necessary to write complex SQL queries or generate effective visualizations, limiting efficient data use and scientific discovery. To address this barrier, we introduce CELEC, a large language model (LLM)-powered framework for automated EHR data extraction and analytics. CELEC translates natural language queries into SQL using a prompting strategy that integrates schema information, few-shot demonstrations, and chain-of-thought reasoning, which together improve accuracy and robustness. On a subset of the EHRSQL benchmark, CELEC achieves execution accuracy comparable to prior systems while maintaining low latency, cost efficiency, and strict privacy by exposing only database metadata to the LLM. CELEC also adheres to strict privacy protocols: the LLM accesses only database metadata (e.g., table and column names), while all query execution occurs securely within the institutional environment, ensuring that no patient-level data is ever transmitted to or shared with the LLM. Ablation studies confirm that each component of the SQL generation pipeline, particularly the few-shot demonstrations, plays a critical role in performance. By lowering technical barriers and enabling medical researchers to query EHR databases directly, CELEC streamlines research workflows and accelerates biomedical discovery.

Keywords: Electronic Health Records, Automated Data Analytics, Large Language Models, Text-to-SQL

1 Introduction

Electronic health records (EHRs) have become a central component of modern healthcare. Since 2009, EHR adoption has increased tenfold, and by 2020, more than 90% of primary care providers worldwide reported daily use [3,13]. When effectively implemented, EHRs improve the accuracy and continuity of patient care while enabling large-scale, cost-efficient data analysis that supports both clinical research and population health management [4, 5, 15]. Unlocking this potential requires methods that make EHR data accessible and interpretable for research, advancing both scientific discovery and healthcare systems.

In practice, however, EHR systems remain challenging for researchers to use. Extracting meaningful data often requires navigating fragmented interfaces and applying technical skills such as writing structured queries or custom analytics, which most researchers lack formal training in [15, 30]. These barriers slow down scientific progress, limit exploratory analysis, and force dependence on database specialists, leaving valuable research opportunities underutilized. Tools that allow researchers to access EHR data and conduct basic analytics without programming expertise could therefore lower entry barriers and accelerate biomedical discovery [8].

Prior work has attempted to bridge this gap using text-to-SQL methods. While these approaches have shown strong performance on EHR question-answering benchmarks [2, 7, 10, 14], it is often unclear whether the underlying database systems are exposed to cloud-based LLMs—a significant risk for sensitive medical datasets, where data leakage might compromise patient privacy.

To address the security concern, we propose **CELEC (Curation of EHR via LLMs under Environmental Constraints)**, a system that restricts LLM access to schema-level metadata only and executes all SQL queries locally, ensuring that no patient-level data are transmitted or shared inadvertently with external services. As a framework for automated EHR data extraction and analytics, CELEC translates natural language questions into SQL using a prompting strategy that combines schema information, few-shot demonstrations, and chain-of-thought reasoning to generate accurate and robust SQL outputs.

Once data are retrieved, CELEC can also generate simple visualizations directly from the extracted dataframes, providing immediate exploratory insights. Designed with real-world constraints in mind, CELEC maintains low latency and cost efficiency. CELEC also adheres to strict privacy protocols: the LLM interacts only with database metadata (e.g., table and column names), while all query execution occurs securely within the institutional environment. At no point is patient-level data transmitted to or shared with the LLM. By lowering technical barriers, CELEC empowers researchers to query EHR databases and perform preliminary analytics independently, streamlining workflows and accelerating scientific discovery.

2 Related Work

2.1 EHR system usability & data accessibility

Despite near-universal adoption of EHRs [3, 26], effective use remains limited by fragmented information, disrupted workflows, and high cognitive load—factors linked to errors, inefficiency, and burnout among both clinicians and researchers [1, 20]. Prior research has shown that extensions such as custom dashboards and enhanced visualization of clinical trends can improve usability [19, 23]. More recently, automated visualization approaches have been explored in the medical domain [33]; however, these efforts often focused on small-scale applications and electronic medical records rather than large-scale, multi-table EHR systems. CELEC advances this line of work by integrating natural language querying with flexible analytics and visualization, providing a deployment-ready framework that directly supports research-driven use cases.

Beyond usability in care delivery, secondary use of EHR data is essential for research, quality improvement, and population health management. Yet accessibility remains a barrier: healthcare professionals often lack the training required to manipulate complex database systems [15, 30]. Some prior efforts employed visual query builders grounded in fixed SQL templates [29], but these approaches sacrifice flexibility for ease of use, limiting their applicability in production environments. CELEC addresses this limitation by leveraging LLMs to generate SQL queries directly from natural language, balancing accessibility with flexibility in data extraction and analysis.

2.2 Medical text-to-SQL

General-purpose text-to-SQL models have achieved strong results on benchmarks such as Spider [32] and WikiSQL [34] [6, 18, 24]. However, performance on these datasets does not guarantee success in clinical contexts, where data complexity and privacy requirements differ substantially from business applications. Early medical-domain systems relied on templates for text-to-SQL translation, which limited their ability to address complex real-world questions [31]. The EHRSQL shared task [16,17] advanced the field by collecting natural language queries from hospital staff and aligning them to databases such as MIMIC-III, eICU, and MIMIC-IV demo. Leading teams on the benchmark explored methods including schema-aware models [10], ensemble prompting [7], and probability-based SQL verification [14]. CELEC employs a streamlined two-call design to strike a balance between accuracy and low latency, while extending its capabilities beyond question answering to support exploratory tasks, such as cohort selection and visualization. Quantitative comparisons are presented in Section 4.

2.3 LLMs in Healthcare

Most LLM applications in healthcare have centered on clinicians for decision support, summarization, and conversational interfaces [21, 25]. However, comparatively little attention has been given to applications that assist researchers with structured data analytics. Recent work has explored connecting EHR databases with Claude Desktop for LLM-powered analytics [2]. In contrast, CELEC introduces a privacy-conscious design that restricts LLM inputs to schema-level metadata, thereby minimizing exposure of raw patient data and reducing the attack surface for leakage.

3 Methods

We designed the CELEC system to be an end-to-end application (see Figure 1). The system will connect to an EHR database. In our implementation, we connect to DuckDB versions of MIMIC-III [12] and MIMIC-IV [11] databases, which are large EHR databases comprising de-identified health-related data, including demographics, measurements, laboratory test results, procedures, and medications. Users can enter a natural language (NL) question as input to specify the objective of their data extraction or analysis. Then, we utilize an LLM to translate the NL question into a SQL query that correctly extracts a succinct yet necessary

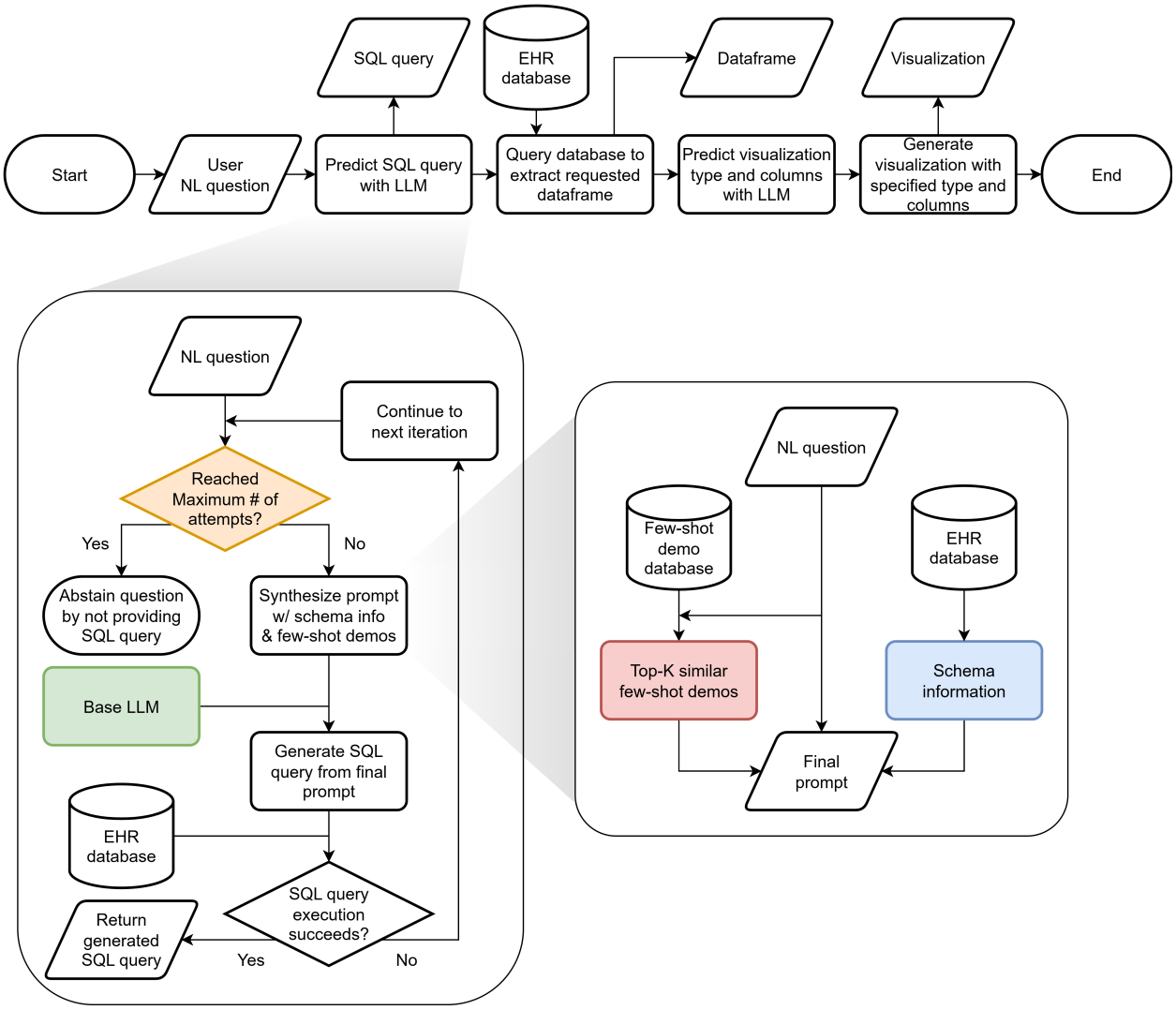

Figure 1: Overview of the CELEC system design

set of data to answer the NL question. We use prompt engineering to enhance the quality of generated SQL code (Section 3.1). Next, we connect to the EHR database and execute the generated SQL code to extract corresponding data. After the requested dataframe is retrieved, we use an LLM to create a visualization that effectively answers the NL question or enables users to better explore their data (Section 3.2).

3.1 SQL generation

We use the `o3-2025-04-16` model to generate SQL queries from NL input. To improve reliability and alignment with the target EHR database, our prompt design incorporates three key elements: (1) **schema information**, (2) **few-shot demonstrations**, and (3) **chain-of-thought (CoT)** reasoning. The complete prompt template is provided in Appendix A. Importantly, only schema-level metadata (table names, column names, and column types) are passed to the LLM; no patient-level data are ever exposed to the LLM.

Schema information. For each database, schema metadata are embedded in the prompt to inform the LLM of available tables and attributes, which reduces hallucination and prevents fabrication of nonexistent aliases. Each table is presented along with its columns, declared data types, and relevant constraints (e.g., primary keys). More details may be found in Appendix A.

Few-shot demonstrations. We include in-context few-shot demonstrations of NL questions paired with their corresponding SQL queries. Demonstrations come from two sources: (a) medical literature using MIMIC datasets, where questions were adapted from published cohort criteria and manually verified (105 demos; see Appendix C), and (b) the EHRSQL benchmark [17], from which we preprocessed the training and validation sets into 4,761 high-quality pairs (see Appendix D). These sources are combined into a single demo database; during inference, CELEC selects the top-k most similar demos ($k = 2$) based on the cosine similarity between the embeddings of the input question and the demo questions. Embeddings are computed using `all-MiniLM-L6-v2` and indexed with ChromaDB for efficient retrieval.

Chain-of-thought (CoT). To guide complex query construction, we augment the prompt with an intermediate reasoning step where the LLM first identifies potentially relevant tables before generating the final SQL. This step enhances alignment between the input question and the selected schema elements, thereby reducing unnecessary joins and extraneous columns. Crucially, the few-shot demonstrations are also formatted to include this intermediate reasoning. By showing demos that explicitly map NL questions to table-selection steps and then to final SQL, we encourage the model to follow the same structure. This not only enhances interpretability but also stabilizes SQL generation in complex queries.

Error handling. After SQL generation, the query is executed locally on the target database. If execution fails (commonly due to hallucinated aliases or function names), the error message is appended to the prompt and the LLM retries generation. By default, up to two retries are allowed, which balances robustness with latency (see ablation results in Section 5). Error messages are processed only by the LLM during retries and are not exposed to users.

3.2 Visualization generation

While existing text-to-SQL systems typically return query results as raw tables, many research and clinical workflows benefit from visual summaries that facilitate pattern interpretations. To address this, CELEC includes an LLM-powered visualization module. After the SQL query is executed and a dataframe retrieved, we provide the LLM with the input NL question along with column metadata from the resulting table. The model predicts (1) an appropriate visualization type (e.g., histogram, bar chart, line plot, scatterplot) and (2) the aesthetic mappings, such as which variables should be plotted on the horizontal and vertical axes.

The prompt includes both instructions and simple input-output examples (see Appendix B), encouraging the model to output a structured specification. Importantly, as in SQL generation, only schema-level metadata is exposed to the LLM; the underlying patient-level data remains inaccessible. Once the visualization specification is returned, CELEC renders the chart using a set of hard-coded TextScript functions. This design ensures consistency and robustness across visualizations, while protecting privacy and minimizing LLM-induced errors. By integrating visualization generation, CELEC enables users to move beyond raw tabular results and obtain exploratory insights directly, thereby lowering the barriers for both researchers and clinicians.

3.3 System integration

While Sections 3.1 and 3.2 describe SQL and visualization generation individually, their integration is key to CELEC's usability in real-world research settings. The modules are orchestrated in a unified pipeline: SQL queries are generated and executed locally, and results can either be returned directly or forwarded to the visualization module for immediate exploratory analysis. Multi-turn refinement mechanisms further ensure that users can iterate naturally, without SQL expertise.

This integration offers system-level benefits that are not captured by either module alone. First, the streamlined design, which relies on only two LLM calls per query, reduces latency compared to ensemble or verification-heavy approaches. Second, the privacy-conscious architecture ensures that at no stage do LLMs access patient-level data; only schema metadata and aggregate specifications are processed. Finally, presenting SQL and visualization under a single framework reduces the technical barrier for researchers and clinicians alike, supporting both reproducible research and practical deployment.

4 Evaluation

We evaluate CELEC on the EHRSQL-2024 benchmark [17], which contains more than 7,000 pairs of NL questions and gold SQL queries against a modified version of the MIMIC-IV demo database. The benchmark is designed to reflect real-world clinical needs, as its queries are typically complex, involving temporal conditions, group operations, and cross-table joins.

Table 1: Comparison of CELEC with teams from the EHRSQL-2024 leaderboard.

System / Team		RS(0) (%)
CELEC (Ours)		**81.05**
LG AI Research & KAIST	[10]	88.17
PromptMind	[7]	82.60
ProbGate	[14]	81.92
KU-DMIS	[22]	72.07
AIRI NLP	[27]	68.89
LTRC-IIITH	[28]	66.84
Saama Technologies	[9]	53.21

4.1 Dataset adaptation

To align the benchmark with CELEC, we filtered and modified the provided train, validation, and test splits. In particular, we removed unanswerable questions where the gold SQL did not execute successfully on the MIMIC-IV demo database (see Appendix D for full details). After adaptation, we used all training and validation points as few-shot candidates for SQL prompting, 10% of the new test set (78 queries) as validation data for hyperparameter tuning, and the remaining 90% (707 queries) for evaluation. By ensuring that all test queries are executable, this adaptation yields a more reliable measure of model capability. Although not identical to the official leaderboard split, the results remain informative for comparison with published systems.

4.2 Evaluation methods

We evaluate system performance using the RS(0) score, the official metric of the EHRSQL benchmark. RS(0) extends execution accuracy (EX) by rewarding abstention on unanswerable questions. Formally, if X is the set of all questions and $X_{ans} \subseteq X$ the subset of answerable ones, then

$$RS(0) = \frac{1}{|X|} \sum_{x \in X} \mathbb{1} \left[(x \in X_{ans} \wedge R_{gen}(x) = R_{gt}(x)) \right.$$
$$\left. \vee (x \notin X_{ans} \wedge g(x) = 0) \right].$$

Since our adapted test set excludes unanswerable questions ($X_{ans} = X$), RS(0) reduces to execution accuracy in our evaluation, but we report it as RS(0) for consistency with leaderboard scores.

4.3 Results

Table 1 compares CELEC with leading systems from the EHRSQL-2024 leaderboard. On our adapted test set, CELEC achieves 81.05% RS(0) accuracy, abstaining in only 0.14% of cases. This result is comparable to the third-place system (ProbGate, 81.92%) and exceeds the fourth-place system (KU-DMIS, 72.07%). Unlike the leaderboard-topping LG AI Research

Table 2: RS(0) score of CELEC under different parameter settings. The first row represents our default settings.

Base LLM	Schema info? (Y/N)	# of few-shot demos	Max # of attempts	RS(0) (%)
o3	**Y**	**2**	**2**	**81.05**
o4-mini	Y	2	2	**73.41**
gpt-4.1	Y	2	2	**77.93**
o3	**N**	2	2	**77.93**
o3	Y	**1**	2	**73.97**
o3	Y	**0**	2	**50.21**
o3	Y	2	**1**	**79.21**

system [10], which relied on full supervised fine-tuning, CELEC achieves this performance in a training-free setting using only few-shot prompting and two LLM calls per query.

In terms of latency, CELEC processes one NL query in an average of 6.02 to 6.11 seconds across the test set. While other systems have not disclosed inference times, CELEC's streamlined two-call design demonstrates practical feasibility for interactive use.

Finally, to safeguard privacy, we verified automatically that all generated SQL queries only reference schema-level columns and tables. No queries attempted to access fabricated or patient-level identifiers, consistent with CELEC's metadata-only exposure design.

4.4 Discussion

These results highlight CELEC's ability to approach state-of-the-art accuracy without training or complex pipelines, while maintaining low latency and strong privacy guarantees. Ablation studies in Section 5 further confirm that few-shot prompting and retry mechanisms are critical to this performance.

5 Ablation Study

We conducted ablation studies to assess the contribution of individual system components. In each experiment, we modified one component while keeping all others fixed, and report the RS(0) score on the adapted EHRSQL test set (Table 2). Results consistently show that each component makes a meaningful contribution to performance, with in-context learning yielding the largest gains.

Base LLM. The choice of the base model had a great impact on accuracy. Our default model, `o3-2025-04-16`, achieved the highest score of 81.05%. Replacing it with the smaller reasoning model `o4-mini-2025-04-16` reduced accuracy to 73.41%, whereas using `gpt-4.1-2025-04-14`, a non-reasoning model, yielded 77.93%. This suggests that both model scale and reasoning optimization are essential for handling the complex queries in EHRSQL.

Schema information. Providing schema metadata to the LLM significantly improved reliability. Without schema information, performance dropped from 81.05% to 77.93%, with frequent hallucinations of nonexistent columns or aliases. This confirms that grounding query generation in explicit schema details is critical to reducing structural SQL errors.

Few-shot demonstrations. In-context demonstrations had the most significant impact of all components. Without demonstrations (zero-shot), CELEC achieved only 50.21%. Adding a single demonstration raised performance sharply to 73.97%, and using two demonstrations, which is our default, further improved it to 81.05%. This trend underscores the crucial role of in-context learning in aligning the model with the structure and semantics of medical databases, with diminishing returns evident beyond two examples.

Maximum attempts. Allowing the model to retry once after a failed SQL execution improved accuracy from 79.21% (single attempt) to 81.05% (two attempts). Error analysis indicates that many first-pass failures stemmed from superficial issues such as hallucinated aliases or function names. Providing the database error message during a second attempt often corrected these, yielding modest but consistent gains with minimal latency overhead.

Overall, these results demonstrate that each design choice contributes to CELEC's performance, with few-shot demonstrations being the most critical factor. The combination of schema grounding, a reasoning-capable base model, and a lightweight retry mechanism together supports robust SQL generation in complex EHR environments.

6 Discussion & Conclusion

We introduced CELEC, a large language model–powered framework for automated EHR data extraction and analytics. CELEC translates natural language questions into executable SQL queries and can generate simple visualizations of query results, lowering the expertise required to access and explore EHR data. Through evaluation on the EHRSQL-2024 benchmark, CELEC achieved an RS(0) score of 81.05% on our adapted test set, comparable to strong leaderboard systems, while maintaining low latency, requiring no additional training, and adhering to a privacy-conscious design. Our ablation studies further highlighted the critical role of few-shot demonstrations, schema grounding, reasoning-optimized base models, and lightweight retry mechanisms in achieving robust SQL generation. Together, these results demonstrate that CELEC is an effective and practical approach for bridging the accessibility gap between researchers and large-scale EHR databases and has the potential to streamline workflows and accelerate discovery in healthcare research.

References

[1] Elham Asgari, Japsimar Kaur, Gani Nuredini, Jasmine Balloch, Andrew M Taylor, Neil Sebire, Robert Robinson, Catherine Peters, Shankar Sridharan, and Dominic Pimenta. Impact of Electronic Health Record Use on Cognitive Load and Burnout Among Clinicians: Narrative Review. *JMIR Med Inform*, 12:e55499, April 2024.

[2] Rafi Al Attrach, Pedro Moreira, Rajna Fani, Renato Umeton, and Leo Anthony Celi. Conversational llms simplify secure clinical data access, understanding, and analysis. *arXiv preprint arXiv:2507.01053*, 2025.

[3] Wesley Barker, Wei Chang, Jordan Everson, Meghan Gabriel, Vaishali Patel, Chelsea Richwine, and Catherine Strawley. The evolution of health information technology for

enhanced patient-centric care in the united states: Data-driven descriptive study. *J Med Internet Res*, 26:e59791, Oct 2024.

[4] Joan A. Casey, Brian S. Schwartz, Walter F. Stewart, and Nancy E. Adler. Electronic Health Records and Population Health Research. *Front Public Health Serv Sys Res*, 2016.

[5] Jawad Chishtie, Natalie Sapiro, Natalie Wiebe, Leora Rabatach, Diane Lorenzetti, Alexander A Leung, Doreen Rabi, Hude Quan, and Cathy A Eastwood. Use of Epic Electronic Health Record System for Health Care Research: Scoping Review. *J Med Internet Res*, 25:e51003, December 2023.

[6] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation, 2023.

[7] Satya Gundabathula and Sriram Kolar. PromptMind team at EHRSQL-2024: Improving reliability of SQL generation using ensemble LLMs. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 360–366, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[8] K Honeyford, P Expert, E.E Mendelsohn, B Post, A.A Faisal, B Glampson, E.K Mayer, and C.E Costelloe. Challenges and recommendations for high quality research using electronic health records. *Frontiers in Digital Health*, 4, 2022.

[9] Mohammed Jabir, Kamal Kanakarajan, and Malaikannan Sankarasubbu. Saama technologies at EHRSQL 2024: SQL generation through classification answer selector by LLM. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 655–671, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[10] Yongrae Jo, Seongyun Lee, Minju Seo, Sung Ju Hwang, and Moontae Lee. LG AI research & KAIST at EHRSQL 2024: Self-training large language models with pseudo-labeled unanswerable questions for a reliable text-to-SQL system on EHRs. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 635–643, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[11] A Johnson, L Bulgarelli, T Pollard, B Gow, B Moody, S Horng, LA Celi, and R Mark. Mimic-iv (version 3.1), 2024. *URL https://doi. org/10.13026/kpb9-mt58*.

[12] Alistair Johnson, Tom Pollard, and Roger Mark. Mimic-iii clinical database (version 1.4). *PhysioNet*, 10(C2XW26):2, 2016.

[13] G. Kerr, N. Kulshreshtha, G. Greenfield, E. Li, T. Beaney, B.W.J. Hayhoe, J. Car, A. Clavería, C. Collins, S.M. Espitia, M.J. Fernandez, G. Gusso, K. Hoedebecke,

R.D. Hoffman, G. Irving, G. Jimenez, L. Laranjo, V. Lazić, H. Lingner, E. Memarian, K. Nessler, B.G. O'Neill, D. Petek, A. Serafini, M. Ungan, A. Majeed, and A.L. Neves. Features and frequency of use of electronic health records in primary care across 20 countries: a cross-sectional study. *Public Health*, 233:45–53, 2024.

[14] Sangryul Kim, Donghee Han, and Sehyun Kim. ProbGate at EHRSQL 2024: Enhancing SQL query generation accuracy through probabilistic threshold filtering and error handling. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 687–696, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[15] Clemens Scott Kruse, Anna Stein, Heather Thomas, and Harmander Kaur. The use of Electronic Health Records to Support Population Health: A Systematic Review of the Literature. *J Med Syst*, 42(11):214, November 2018.

[16] Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. Ehrsql: A practical text-to-sql benchmark for electronic health records, 2023.

[17] Gyubok Lee, Sunjun Kweon, Seongsu Bae, and Edward Choi. Overview of the EHRSQL 2024 shared task on reliable text-to-SQL modeling on electronic health records. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 644–654, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[18] Zhishuai Li, Xiang Wang, Jingjing Zhao, Sun Yang, Guoqing Du, Xiaoru Hu, Bin Zhang, Yuxiao Ye, Ziyue Li, Rui Zhao, and Hangyu Mao. Pet-sql: A prompt-enhanced two-round refinement of text-to-sql with cross-consistency, 2024.

[19] Lukasz M. Mazur, Prithima R. Mosaly, Carlton Moore, and Lawrence Marks. Association of the Usability of Electronic Health Records With Cognitive Workload and Performance Levels Among Physicians. *JAMA Netw Open*, 2(4):e191709, April 2019.

[20] Olufisayo Olakotan, Ray Samuriwo, Hadiza Ismaila, and Samuel Atiku. Usability Challenges in Electronic Health Records: Impact on Documentation Burden and Clinical Workflow: A Scoping Review. *Evaluation Clinical Practice*, 31(4):e70189, June 2025.

[21] David Oniani, Xizhi Wu, Shyam Visweswaran, Sumit Kapoor, Shravan Kooragayalu, Katelyn Polanska, and Yanshan Wang. Enhancing large language models for clinical decision support by incorporating clinical practice guidelines, 2024.

[22] Sungho Park, Hyeonwoo Kim, Yeonsoo Kim, Minji Jeong, Minbyul Sung, Edward Choi, and Juho Lee. KU-DMIS at EHRSQL 2024: Generating SQL query via question templatization in EHR. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk

Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 462–468, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[23] Ari H. Pollack and Wanda Pratt. Association of Health Record Visualizations With Physicians' Cognitive Load When Prioritizing Hospitalized Patients. *JAMA Netw Open*, 3(1):e1919301, January 2020.

[24] Mohammadreza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction, 2023.

[25] Niroop Channa Rajashekar, Yeo Eun Shin, Yuan Pu, Sunny Chung, Kisung You, Mauro Giuffre, Colleen E Chan, Theo Saarinen, Allen Hsiao, Jasjeet Sekhon, Ambrose H Wong, Leigh V Evans, Rene F. Kizilcec, Loren Laine, Terika Mccall, and Dennis Shung. Human-algorithmic interaction using a large language model-augmented artificial intelligence clinical decision support system. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery.

[26] Yun Shen, Jiamin Yu, Jian Zhou, and Gang Hu. Twenty-Five Years of Evolution and Hurdles in Electronic Health Records and Interoperability in Medical Research: Comprehensive Review. *J Med Internet Res*, 27:e59024, January 2025.

[27] Oleg Somov, Alexey Dontsov, and Elena Tutubalina. AIRI NLP team at EHRSQL 2024 shared task: T5 and logistic regression to the rescue. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 431–438, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[28] Jerrin Thomas, Pruthwik Mishra, Dipti Sharma, and Parameswari Krishnamurthy. LTRC-IIITH at EHRSQL 2024: Enhancing reliability of text-to-SQL systems through abstention and confidence thresholding. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Danielle Bitterman, editors, *Proceedings of the 6th Clinical Natural Language Processing Workshop (ClinicalNLP)*, pages 697–702, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[29] Yuan Tian, Jonathan K. Kummerfeld, Toby Jia-Jun Li, and Tianyi Zhang. Sqlucid: Grounding natural language database queries with interactive explanations, 2024.

[30] Chen Hsi Tsai, Aboozar Eghdam, Nadia Davoody, Graham Wright, Stephen Flowerday, and Sabine Koch. Effects of electronic health record implementation and barriers to adoption and use: A scoping review and qualitative analysis of the content. *Life*, 10(12), 2020.

[31] Ping Wang, Tian Shi, and Chandan K. Reddy. Text-to-sql generation for question answering on electronic medical records, 2020.

[32] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, 2019.

[33] Haodi Zhang, Siqi Ning, Qiyong Zheng, Jinyin Nie, Liangjie Zhang, Weicheng Wang, and Yuanfeng Song. Towards visualizing electronic medical records via natural language queries, 2025.

[34] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning, 2017.

A Prompt for SQL Generation

Listing 1 shows the prompt template we use to generate SQL code from an NL question. The placeholders {`table_info`}, {`fewshot_demo`}, and {`question`} correspond to schema information, few-shot demonstrations, and the input NL question, respectively. A SQL query wrapper guides the LLM to produce structured output, enabling post-processing components to reliably extract the generated SQL query using regular expression matching.

Listing 2 shows the template used to format schema information. After the table name, the schema is presented in tabular form, where each row represents an attribute in the database. The columns `name`, `type`, `notnull`, `dflt_value`, and `pk` indicate the column name, declared data type, whether the column has a non-null constraint, its default value, and whether it is a primary key. Each table is formatted in this way, and tables are listed sequentially in lexicographical order.

Listing 3 shows the template for formatting few-shot demonstrations. After the demo NL question, we include a chain-of-thought step beginning with the phrase "Let's think step-by-step," followed by a list of relevant tables referenced in the demo SQL query. The demo SQL query itself is then presented. All few-shot demos in a prompt are displayed in this format and ordered by decreasing similarity of question embeddings, as described in the main text.

B Prompt for Visualization Generation

Listing 4 shows the prompt template we use to generate a visualization from an NL question and the corresponding extracted dataframe. The placeholders {`viz_name`}, {`columns`}, and {`question`} correspond to (1) the list of visualization types supported by our system's front end (scatterplot, bar chart, line chart, and histogram), (2) the columns of the extracted dataframe, and (3) the input NL question, respectively.

C Medical Literature–Inspired Demos

To supplement benchmark data, we created 105 few-shot demonstrations inspired by published studies that used the MIMIC-IV dataset. Candidate cohort criteria were identified by searching for "MIMIC-IV" on PubMed and reviewing approximately 20 recent papers. From these, we adapted subject selection descriptions into natural language questions. For example, a criterion such as "We selected patients that..." was paraphrased into "Select patients that...," while visualization-oriented questions were framed in formats like "Generate the distribution of ... for all patients that...." In addition to direct adaptations, some questions were modified to expand coverage across different database attributes or application scenarios (e.g., visualization as well as cohort selection).

The corresponding gold SQL queries were first generated by `gpt-4-1106-preview` and then validated against DuckDB implementations of the MIMIC databases. The correctness of the queries is manually checked by executing them on the database. Invalid queries were debugged and corrected to yield syntactically valid and semantically accurate equivalents in the DuckDB dialect.

Listing 1: Prompt template for SQL generation.

```
### You are a DuckDB expert.
Given an input question, write a syntactically correct
   DuckDB query that answers the input question.
You must write the query using only functions and syntax
   supported by DuckDB.
Even though some examples below (the few-shot demos)
   include table names prefixed with "physionet-data." and
   may use functions from other SQL dialects, your final
   query must:
- Use only DuckDB-compatible functions (for example, use
   DATEDIFF instead of TIMESTAMP_DIFF).
- Use local table names that are capitalized and do NOT
   include the "physionet-data." prefix.
- Never query for all columns from a table - only query the
   columns needed to answer the question.
- Wrap each column name in backticks (`) to denote them as
   delimited identifiers.
- Only reference column names that exist in the tables
   provided below.
- Be careful not to query for columns that do not exist,
   and ensure you are using the correct columns for each
   table.
- Use CURRENT_DATE() if the question involves "today".
- Exclude null values.
- Use aggregation functions like COUNT() and a GROUP BY
   clause if the question asks for distribution.
- Use different aliases for all tables and output columns.
Wrap the SQL query like this:
```sql
```

Here is the information about the tables:
{schema_info}

Some example pairs of questions and corresponding SQL
   queries (these examples might use functions from other
   dialects) are provided below:
{fewshot_demo}

Question: {question}
SQL: Let's think step-by-step.
```

Listing 2: Schema information for SQL generation for example table.

```
Table: admissions
Schema:
cid                 name        type  notnull dflt_value
   pk
0     0               row_id      BIGINT    False        None
   False
1     1           subject_id      BIGINT    False        None
   False
2     2              hadm_id      BIGINT    False        None
   False
3     3            admittime   TIMESTAMP    False        None
   False
4     4            dischtime   TIMESTAMP    False        None
   False
5     5       admission_type     VARCHAR    False        None
   False
6     6   admission_location     VARCHAR    False        None
   False
7     7   discharge_location     VARCHAR    False        None
   False
8     8            insurance     VARCHAR    False        None
   False
9     9             language     VARCHAR    False        None
   False
10   10       marital_status     VARCHAR    False        None
   False
11   11                  age      BIGINT    False        None
   False

Table: chartevents
Schema:
cid         name        type  notnull dflt_value      pk
0    0       row_id      BIGINT    False        None  False
...
7    7    valueuom     VARCHAR    False        None  False

Table: cost
...
...
```

Listing 3: Example few-shot demo for SQL generation.

```
##
Question: What is the minimum total hospital cost that
   involved a procedure called other enterostomy since 2100?
Answer: Let's think step-by-step.
1. Identify the relevant tables:
-- cost
-- procedures_icd
-- d_icd_procedures
2. Final SQL query:
SELECT MIN(T1.C1) FROM (SELECT SUM(cost.cost) AS C1 FROM
   cost WHERE cost.hadm_id IN (SELECT
   procedures_icd.hadm_id FROM procedures_icd WHERE
   procedures_icd.icd_code = (SELECT
   d_icd_procedures.icd_code FROM d_icd_procedures WHERE
   d_icd_procedures.long_title = 'other enterostomy')) AND
   STRFTIME(CAST(cost.chargetime AS TIMESTAMP), '%Y') >=
   '2100' GROUP BY cost.hadm_id) AS T1

##
Question: What is the maximum total hospital cost
   associated with postprocedural pneumothorax in 2100?
Answer: Let's think step-by-step.
1. Identify the relevant tables:
-- cost
-- diagnoses_icd
-- d_icd_diagnoses
2. Final SQL query:
SELECT MAX(T1.C1) FROM (SELECT SUM(cost.cost) AS C1 FROM
   cost WHERE cost.hadm_id IN (SELECT diagnoses_icd.hadm_id
   FROM diagnoses_icd WHERE diagnoses_icd.icd_code =
   (SELECT d_icd_diagnoses.icd_code FROM d_icd_diagnoses
   WHERE d_icd_diagnoses.long_title = 'postprocedural
   pneumothorax')) AND STRFTIME(CAST(cost.chargetime AS
   TIMESTAMP), '%Y') = '2100' GROUP BY cost.hadm_id) AS T1
```

Listing 4: Prompt template for visualization generation

```
### You are an expert in data visualization. Given an input
    question and the columns of a dataframe, determine the
    type of visualization and the axis columns that most
    appropriately address the question.
For the type of visualization, select only among the
    following options: {viz_names}. For the axis columns,
    select only the column names you are given.
Select only one column if the visualization type requires
    only one column of data (e.g., histogram), and select
    two columns if the visualization type requires two
    columns of data (e.g., scatterplot)
If the prompt is related to visualizing a distribution and
    there is a count column, choose the bar chart and select
    two columns instead of selecting the histogram or
    density plot and only one column.

Format the output like the following without any extra text
    or explanations (the Yaxis segment can be omitted if
    only one column is selected):
VizType: VISUALIZATION-TYPE-NO; Xaxis: HORIZONTAL-AXIS;
    Yaxis: VERTICAL-AXIS
As an example, the output "VizType: 0; Xaxis: cal_daily;
    Yaxis: bmi" means a scatterplot with the column
    "cal_daily" as the horizontal axis and the column "bmi"
    as the vertical axis.

### Here are the column names in the given dataframe.
{columns}

### Question: {question}
### Answer:
```

All demos follow the same format as described in Appendix A, including chain-of-thought reasoning steps and explicit table selection, ensuring consistency across sources. This process yielded 105 high-quality demonstrations that complement benchmark-derived examples and better reflect clinically realistic query patterns.

D EHRSQL Data Point Preprocessing

We modified the train, validation, and test sets of EHRSQL, particularly the gold SQL queries, to ensure compatibility with CELEC's design and execution environment. Since the original EHRSQL benchmark runs on a SQLite version of the MIMIC-IV demo database, while CELEC operates on DuckDB instances of MIMIC-III and MIMIC-IV, we adapted the data through a combination of automated translation and manual correction. We also filtered out unanswerable questions to focus on executable, clinically relevant queries. Below, we describe these preprocessing steps.

D.1 Translating between SQL dialects

We first applied the SQLGlot package in Python to translate gold SQL queries from SQLite into DuckDB syntax. While SQLGlot handled many standard cases automatically, it failed to account for several recurring issues. To improve compatibility, we introduced a set of targeted preprocessing rules that addressed the most frequently observed errors, as summarized below:

- **Current time.** SQLite uses the keyword `CURRENT_TIME` to return the current time of day, whereas DuckDB expects `CURRENT_TIMESTAMP`, which returns both date and time. SQLGlot left these untranslated, so we replaced all instances of `CURRENT_TIME` with `CURRENT_TIMESTAMP`.
- **Datetime expressions.** SQLite allows flexible modifiers within `datetime()` calls, such as "start of month" or "+1 day." SQLGlot did not handle these consistently, so we introduced explicit mappings:
  - `datetime(expr)` → `CAST(expr AS TIMESTAMP)`
  - `datetime(expr, 'start of X')` → `date_trunc('X', expr)`
  - `datetime(expr, '+/-N unit')` → `expr +/- INTERVAL 'N unit'`
  - `datetime(expr, '-0 year')` → no-op (left unchanged)

These issues were not isolated: each of the error types above appeared more than 500 times in the dataset. By handling them systematically, we corrected the vast majority of translation failures. Regarding other translational errors, due to their rarity and heterogeneity, we did not attempt case-level debugging and discarded them in the subsequent filtering stage.

D.2 Discarding unanswerable questions

After translation, we ensured that all splits contained only answerable queries, since detecting unanswerable ones is not a primary goal of CELEC. Unanswerable cases arose from two sources. First, the original EHRSQL benchmark deliberately included unanswerable questions as a separate evaluation task. Second, some queries became unanswerable under our setup: although answerable on the SQLite-based benchmark database, they failed on the official MIMIC-IV demo database when executed in DuckDB. Such failures included gold queries that produced execution errors or queries that executed but returned an empty dataframe. Our preprocessing identified 609 queries that consistently yielded empty dataframes; these were excluded from further use.

We discarded all such cases across the training, validation, and test sets, retaining only those queries where the gold SQL executed successfully on the official MIMIC-IV demo database and returned a non-empty dataframe. This filtering process resulted in new dataset splits comprising 3,976 training queries, 785 validation queries, and 785 test queries. For system development, we further divided the new test set into a small validation portion of 78 queries (10%) for hyperparameter tuning and 707 queries (90%) for final evaluation, as described in the main text.

This two-step preprocessing pipeline, i.e., translation followed by filtering, ensured that the datasets used in CELEC are coherent with DuckDB's execution environment and focus exclusively on answerable, clinically meaningful queries.

E User Interface Demonstration

Figure 2 illustrates the CELEC user interface, with results of running on the example question "What are the five most frequently prescribed medications for patients in their 40s since 2100?" The system enables users to formulate database queries by entering a natural language question and selecting the appropriate database and LLM backbone. Upon submission, the corresponding SQL query is automatically generated. The interface further provides modules for inspecting the resulting dataframe, visualizing query outputs, and constructing cohort selection flowcharts.

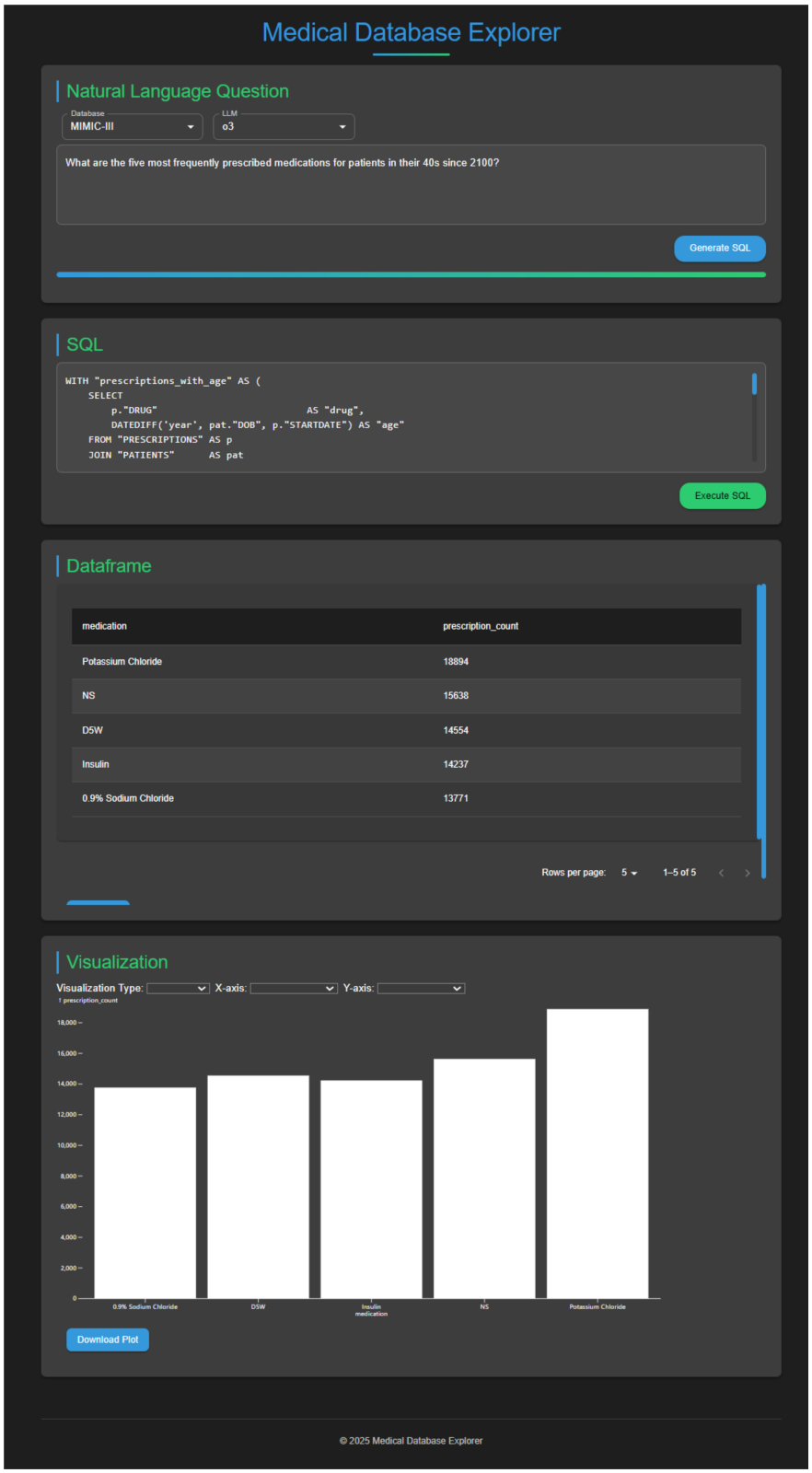

Figure 2: CELEC user interface, with results of running on an example question