Patient-Similarity Cohort Reasoning in Clinical Text-to-SQL

Yifei Shen[*W] **Yilun Zhao**[* Y] **Justice Ou**[Y] **Tinglin Huang**[Y] **Arman Cohan**[Y]
[W] University of Washington [Y] Yale University

Abstract

Real-world clinical text-to-SQL requires reasoning over heterogeneous EHR tables, temporal windows, and patient-similarity cohorts to produce executable queries. We introduce CLINSQL, a benchmark of 633 expert-annotated tasks on MIMIC-IV v3.1 that demands multi-table joins, clinically meaningful filters, and executable SQL. Solving CLINSQL entails navigating schema metadata and clinical coding systems, handling long contexts, and composing multi-step queries beyond traditional text-to-SQL. We evaluate 22 proprietary and open-source models under Chain-of-Thought self-refinement and use rubric-based SQL analysis with execution checks that prioritize critical clinical requirements. Despite recent advances, performance remains far from clinical reliability: on the test set, GPT-5-mini attains 74.7% execution score, DeepSeek-R1 leads open-source at 69.2% and Gemini-2.5-Pro drops from 85.5% on Easy to 67.2% on Hard. Progress on CLINSQL marks tangible advances toward clinically reliable text-to-SQL for real-world EHR analytics.

Data yifeis02/ClinSQL
Code Barryshen1/ClinSQL

1 Introduction

Automating clinical data analysis requires bridging natural-language questions from clinicians to executable queries over complex electronic health record (EHR) databases. While large language models (LLMs) have recently excelled at text-to-SQL and database reasoning on general-domain benchmarks (Yu et al., 2018; Wei et al., 2024; Yang et al., 2025), real-world clinical analysis presents distinct challenges: specialized medical terminology, fine-grained temporal reasoning across heterogeneous tables, and cohort-level clinical reasoning

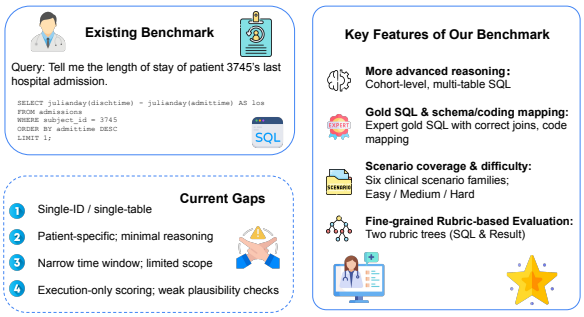

Figure 1: Overview of the CLINSQL benchmark.

that goes beyond point retrieval to compare *similar* patients under clinically meaningful constraints (Yu et al., 2018; Li et al., 2023; Wei et al., 2024). These requirements are not merely larger versions of the classic text-to-SQL problem; they demand workflows that integrate domain knowledge, temporal windows, coding systems, and outcome-aware analytics over longitudinal data (Johnson et al., 2023).

Foundational text-to-SQL evaluations (*e.g.,* WikiSQL, Spider 1.0, BIRD) catalyze progress on cross-domain parsing and database generalization (Zhong et al., 2017; Yu et al., 2018; Li et al., 2023). Recent enterprise-style benchmarks (*i.e.,* Spider 2.0) further expose challenges from large schemas, diverse SQL dialects, and multi-step workflows (Wei et al., 2024). However, clinical settings introduce additional, domain-specific hurdles: temporal abstractions (*e.g.,* first 24/48/72 hours), clinical ranges/units, ICD/medication coding, and cohort construction for outcome comparison. Prior clinical text-to-SQL datasets, notably MIMIC-SQL (Wang et al., 2020) and EHRSQL (Lee et al., 2022), demonstrate feasibility on EHR schemas but predominantly emphasize single-patient or statistical summaries and seldom require *patient-similarity* cohort reasoning central to real-world clinical decision making.

To bridge this gap, we introduce CLINSQL, a

* Equal Contributions. Correspondence: Yilun Zhao (yilun.zhao@yale.edu)

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1367–1412

Dataset	Task	Source	Data Construction	Patient_id Optional
		General Text-to-SQL Benchmarks		
WikiSQL (Zhong et al., 2017)	Single-table Text-to-SQL	Wikipedia and SQL tables	Crowdsourcing	-
Spider (Yu et al., 2018)	Cross-domain, multi-table Text-to-SQL	Diverse real DB schemas	Expert annotation	-
Spider 2.0 (Wei et al., 2024)	Real-world enterprise workflows	Enterprise-scale DBs	Expert + synthetic	-
KaggleDBQA (Lee et al., 2021)	Realistic DBs from Kaggle	Real-world multi-table DBs	Author-written Qs	-
BIRD (Li et al., 2023)	Large-scale Text-to-SQL	95 DBs across 37 domains	Crowdsourcing + expert review	-
LiveBench (White et al., 2024)	Contamination-limited evaluation	Mixed sources incl. DB tasks	Expert-authored, verifiable	-
		Healthcare Benchmarks		
PubMedQA (Jin et al., 2019a)	Biomedical QA	PubMed abstracts	Heuristic generation + manual labels	-
MedQA (Jin et al., 2021)	Exam-style multiple-choice QA	Medical board exam questions	Exam scrape	-
MedMCQA (Pal et al., 2022)	Broad medical MCQ QA	Multi-subject exam questions	Exam scrape	-
MedExQA (Kim et al., 2024b)	Medical QA w/ explanations	Mock tests & online exams	Manual collection/cleaning	-
MedXpertQA (Zhang et al., 2025)	Expert-level multimodal medical QA	Specialty board Qs; multimodal clinical info	Collection + filtering + synthesis; expert review	-
emrQA (Pampari et al., 2018)	Template-driven clinical QA	De-identified clinical notes i2b2	Template generation (i2b2)	✗
DrugEHRQA (Wang et al., 2022)	Medication-centric QA	EHR notes + structured meds	Template generation + sample human check	✗
EHRXQA (Bae et al., 2023)	Multi-modal EHR QA	Notes + chest X-ray images	Derived from MIMIC-CXR-VQA & EHRSQL; curated	✗
EHRNoteQA (Kweon et al., 2024)	Discharge-summary QA	Real EHR discharge summaries	GPT-4 generation + clinician review	✗
DischargeQA (Ou et al., 2025)	Discharge-related clinical QA	EHR discharge summaries	Generated from discharge data	✗
RadQA (Soni et al., 2022)	Radiology report QA	Radiology reports	Physician-authored Qs + span annotation	✗
		EHR Text-to-SQL Benchmarks		
MIMICSQL (Wang et al., 2020)	NL → SQL clinical	MIMIC-III structured tables	Auto-generated Qs + crowdsourcing filter	✓
EHRSQL (Lee et al., 2022)	Practical NL → SQL	Hospital EHR schemas	Hospital-staff utterances + manual SQL annotation	✓
EHRSQL-ST (Lee et al., 2024)	Reliable Text-to-SQL evaluation	Same family of EHR schemas	Organizer-curated evaluation splits	✓
EHR-SeqSQL (Ryu et al., 2024)	NL → SQL EHR	Institutional EHR DB	Decomposition of EHRSQL into sequential tasks	✓
CLINSQL	Text-to-SQL with advanced reasoning	EHR tables	Expert annotation + validation; fine-grained eval rubrics	✓

Table 1: Comparison of CLINSQL with existing Text-to-SQL and Healthcare Benchmarks. The "Patient_id Optional" column indicates whether a benchmark supports supplying an optional de-identified anchor patient identifier (*e.g.,* MIMIC `subject_id`/`hadm_id`) alongside the question to ground patient-similarity or patient-specific queries. ✓: supported; ✗: not supported; "-": not applicable.

benchmark of 633 expert-annotated clinical text-to-SQL tasks on the MIMIC-IV v3.1 database (Johnson et al., 2023). A high-level benchmark overview appears in Figure 1, and Figure 2 details the construction pipeline: we design six scenario types to reflect real clinical settings. Each example is grounded in a concrete scenario and requires composing multi-table, temporally aware SQL with *patient-similarity* cohort construction. Difficulty is stratified by SQL and clinical reasoning complexity. We adopt rubric-based evaluation with critical-first aggregation and execution checks that verify result format and clinical plausibility while allowing equivalent formulations.

We evaluate 22 proprietary and open-source models with Chain-of-Thought self-refinement and find that CLINSQL remains challenging: Gemini-2.5-Pro drops from 85.5% execution on Easy to 67.2% on Hard; GPT-5-mini leads overall test execution at 74.7%, and DeepSeek-R1 tops open-source models at 69.1%. Even for these models, Hard split execution remains below 70%, underscoring the difficulty of CLINSQL. Our error analysis reveals that most failures stem from cohort specification drift (*e.g.,* relaxed ICD/item constraints), schema or output mismatches, and mis-specified clinical aggregations, even for top-performing models. Guided by these findings, we further study a schema-hinted inference setting that foregrounds clinically validated filters and expected outputs, yielding consistent execution gains, especially on medium and hard cases.

We summarize our contributions as follows:

- We introduce a clinically grounded text-to-SQL benchmark that requires *patient-similarity* cohort construction and multi-step temporal reasoning over heterogeneous EHR tables.
- We curate six families of realistic clinical scenarios and provide a rubric-structured evaluation for reliable automated evaluation.
- We benchmark 22 proprietary and open-source models and release a rubric-based error taxonomy that highlights the challenges that future clinical text-to-SQL systems must address.

2 Related Work

General Text-to-SQL Benchmarks. General-domain Text-to-SQL has evolved through successive foundational benchmarks (Zhong et al., 2017; Yu et al., 2018; Lee et al., 2021; Li et al., 2023; Wei et al., 2024; White et al., 2025). However, healthcare Text-to-SQL presents fundamental challenges that distinguish it from general-domain applications, requiring medical terminology, complex temporal relationships, and clinical reasoning that extends beyond standard database operations to

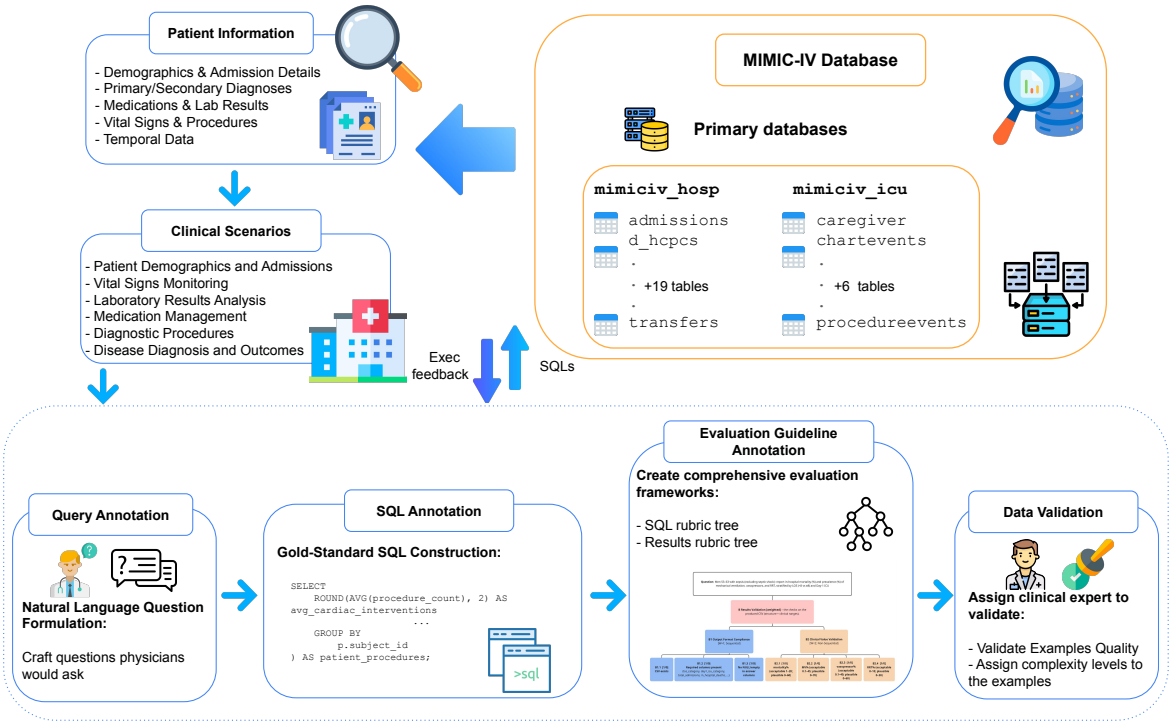

Figure 2: Overview of CLINSQL construction pipeline. The process begins with scenario design and patient selection, followed by question authoring. Annotators then perform database analysis and schema mapping on MIMIC-IV, write executable gold SQL, and construct tree-structured rubrics for SQL and results validation.

incorporate medical decision-making logic (Lee et al., 2022; Wang et al., 2020). Most critically, the patient similarity reasoning paradigm central to CLINSQL represents a fundamental departure from general Text-to-SQL evaluation, as healthcare queries require identifying patient cohorts based on multi-dimensional similarity criteria rather than simple retrieval or aggregation operations.

Healthcare NLP/ML Benchmarks. Healthcare NLP benchmarks have evolved from general medical knowledge QA (Jin et al., 2019b; Hendrycks et al., 2021; Jin et al., 2021; Pal et al., 2022; Singhal et al., 2022; Wang et al., 2024; Kim et al., 2024b) to sophisticated clinical QA tasks with expert-level capabilities and real clinical data utilization (Pampari et al., 2018; Saleh and Pecina, 2019; Suominen et al., 2020; Bardhan et al., 2022; Bae et al., 2023; Kweon et al., 2024; Kim et al., 2024b; Zhang et al., 2025; Chen et al., 2025; Ou et al., 2025). However, as shown in Table 1, existing clinical Text-to-SQL datasets (Wang et al., 2020; Lee et al., 2022; Ryu et al., 2024; Lee et al., 2024; Sivasubramaniam et al., 2024; Kim et al., 2024a) predominantly emphasize statistical analyses rather than addressing authentic clinical questions encountered in real-world practice, and consistently assume queries target specific patients with known identifiers, thereby representing only a limited subset of actual clinical analysis scenarios. Our work addresses these limitations by centering on patient-similarity cohort reasoning over MIMIC-IV v3.1, requiring models to define cohorts, apply temporal and phenotyping logic, and compute stratified cohort-level outcomes that mirror real clinical workflows.

3 Benchmark Construction

CLINSQL is designed to comprehensively evaluate Text-to-SQL capabilities within realistic clinical scenarios. Our benchmark, built upon MIMIC-IV v3.1 (Johnson et al., 2023), incorporates complex analytical scenarios that require sophisticated clinical reasoning and the multi-step integration of diverse clinical data. Figure 2 provides an overview of our benchmark construction pipeline. Table 2 presents the six core clinical scenarios that reflect real-world healthcare data analysis needs and clinical decision-making. Concrete scenario examples and rubric trees are provided in Appendix B. In the following sections, we detail the query annotation, SQL annotation, evaluation guideline annotation, and data validation. The expert annotator biographies are summarized in Appendix A, and

Clinical Scenarios	Example Question Provided in Appendix B.
Patient Demographics and Admissions Analysis of patient demographics and administrative data (admissions, length of stay), testing foundational SQL skills and understanding of clinical administrative workflow.	For an 81-year-old female: among female Medicare patients aged 76–86 transferred from another hospital with principal AMI (ICD-9 410*/ICD-10 I21*), report 30-day readmission rate; median index LOS for readmitted vs not; percent index stays $>$ 4 days. *Complexity: Hard* (Appendix B.1)
Vital Signs Monitoring Temporal analysis of vital signs (e.g., blood pressure, heart rate), designed to test time-series reasoning, understanding of clinical normal ranges, and trend identification capabilities.	I have a 60-year-old man in the ICU. In male ICU patients aged 55–65 with HFNC within 24 hours versus condition-matched ICU controls, what are the instability score median and p25/p75/p95, tachycardia and hypotension burden, ICU LOS and mortality? *Complexity: Medium* (Appendix B.2)
Laboratory Results Analysis Analysis of trends in laboratory results, designed to test knowledge of medical terminology, unit conversions, and the ability to correlate lab values with clinical conditions.	I have a 51-year-old female with suspected ACS. Among female ACS admissions age 46–56, what are counts, percentages, and mean hospital length of stay for first hs-TnT: Normal, Borderline, Myocardial Injury? *Complexity: Medium* (Appendix B.3)
Medication Management Analysis of medication regimens (prescriptions, dosing, interactions), designed to test complex temporal reasoning and pharmacological knowledge to ensure medication safety.	I have a 64-year-old female inpatient. Among females aged 59–69, what's the IQR of single inpatient amiodarone prescription durations (days)? *Complexity: Easy* (Appendix B.4)
Diagnostic Procedures Temporal sequencing of diagnostic procedures and interventions, designed to evaluate the understanding of clinical workflows, procedural relationships, and care coordination.	Evaluating an 88-year-old man: among male patients aged 83–93 with sepsis on their first ICU stay, stratify first-72-hour diagnostic intensity (distinct procedures) into quartiles and report mean procedure counts, mean ICU LOS in days, and mortality (%) per quartile. *Complexity: Hard* (Appendix B.5)
Disease Diagnosis and Outcomes Analysis of diagnoses (ICD-9/10 codes), comorbidities, and clinical outcomes, designed to test knowledge of medical coding, integrated clinical reasoning, and the ability to assess treatment effectiveness and patient prognosis.	I have a 75-year-old female inpatient with pulmonary embolism. For female inpatients aged 70–80 with PE, stratify into risk-score quintiles and report per quintile: 90-day mortality, general 70–80 female 90-day mortality (comparison), AKI and ARDS rates, and median survivor LOS. *Complexity: Hard* (Appendix B.6)

Table 2: Definition of clinical scenario types in CLINSQL.

the annotation interface is shown in Appendix I.

3.1 Query Annotation

Clinical Scenario Development. Each annotator is assigned one of six scenario types and selects a representative patient from MIMIC-IV v3.1 (Johnson et al., 2023). Sampling is stratified across five dimensions: **(1) Age** uses scenario-specific ranges spanning 25–85 years; **(2) Clinical condition** covers major categories (*e.g.,* cardiovascular, respiratory, metabolic, infectious, post-operative); **(3) Healthcare utilization** varies admission type (*e.g.,* emergency, elective, urgent), insurance (*e.g.,* Medicare, Medicaid, commercial), and length of stay (*e.g.,* 2–15 days); **(4) Acuity** distinguishes settings (*e.g.,* ward vs ICU) with risk strata and monitoring intensity; and **(5) Temporal windows** include early windows (*e.g.,* first 24/48/72 hours), the full hospitalization, and procedure-specific periods. To prevent data contamination and ensure benchmark integrity, all information related to the selected patients is removed from the database prior to model evaluation.

Natural Language Question Formulation. Annotators craft natural language questions that physicians would realistically ask given the provided patient information and scenario type. Each question must require database querying and cannot be answered through simple observation (*e.g.,* questions requiring temporal analysis or aggregation across multiple records). Questions incorporate appropriate medical terminology while maintaining clarity and clinical authenticity.

3.2 SQL Annotation

Our SQL annotation process is divided into two steps: database analysis and schema mapping, and gold-standard SQL construction. Each clinical question undergoes comprehensive database analysis by annotators, followed by the development of gold-standard SQL to produce high-quality executable queries.

Database Analysis and Schema Mapping. For each natural language clinical question, annotators first conduct a systematic database analysis to identify the required MIMIC tables and establish the necessary relationships between clinical entities. This process includes locating relevant database tables (e.g., patients, admissions, diagnoses_icd), identifying key features including specific columns and clinical values (e.g., gender='M', icd_code LIKE '410%'), and mapping clinical concepts to database schema elements while considering temporal constraints and data integrity requirements. A concise schema reference for MIMIC-IV is provided in Appendix P.

Gold-Standard SQL Construction. Following the database analysis phase, expert annotators develop comprehensive gold-standard SQL queries

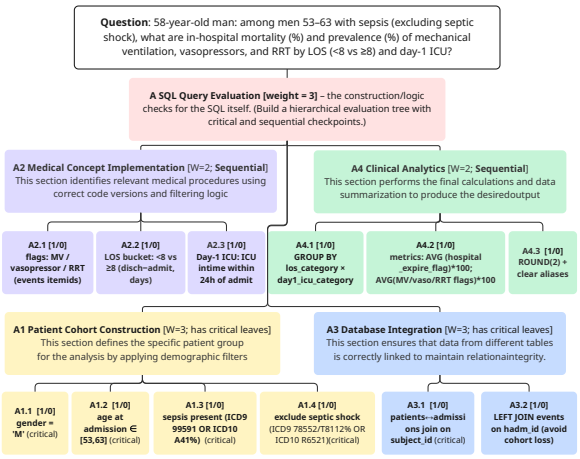

Figure 3: Example of a SQL evaluation rubric tree.

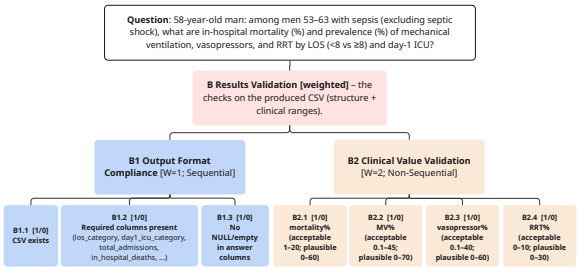

Figure 4: Example of an executed result rubric tree.

that accurately translate clinical questions into executable database operations. Each SQL implementation undergoes rigorous development processes including multi-table join construction with proper foreign key relationships to ensure data consistency, temporal constraint implementation using appropriate date functions and time-based filtering (*e.g.,* DATE_DIFF for length of stay calculations, charttime-based temporal analysis), clinical value range validation incorporating medical domain knowledge and normal physiological parameters (*e.g.,* hemoglobin levels between 7-18 g/dL, age calculations using anchor_age and anchor_year), edge case handling for common clinical database issues including null value management and data quality constraints, and query optimization to maintain computational efficiency while preserving clinical accuracy.

3.3 Evaluation Guideline Annotation

To support reliable automated evaluation, each clinical question is accompanied by a guideline comprising two rubric trees: one for SQL evaluation and one for executed results.

Algorithm 1 Critical-First Score Aggregation

Require: rubric tree T, node weights W, critical flags C, sequential flags S
function EVALUATENODE($node$)
  if $node$ is leaf **then**
    return LLM judge evaluation score $\{0, 1\}$
  end if
  $critical\_children \leftarrow \{c \in children : C[c] = \text{true}\}$
  $noncritical\_children \leftarrow \{c \in children : C[c] = \text{false}\}$
  for $c \in critical\_children$ **do**
    $score[c] \leftarrow$ EVALUATENODE(c)
    if $score[c] \neq 1$ **then return** 0
    end if
  end for
  if $noncritical\_children = \emptyset$ **then return** 1
  end if
  $sum \leftarrow 0, total\_weight \leftarrow 0$
  for $c \in noncritical\_children$ **do**
    $score[c] \leftarrow$ EVALUATENODE(c)
    $sum \leftarrow sum + W[c] \times score[c], total\_weight \leftarrow total\_weight + W[c]$
    if $S[c] = \text{true} \wedge score[c] = 0$ **then break**
    end if
  end for
  return $\frac{sum}{total\_weight}$
end function
$final\_score \leftarrow$ EVALUATENODE($root$)
return $final\_score$

Rubric Design Principles. Our evaluation framework employs tree-structured rubrics that hierarchically decompose complex evaluation tasks into granular, verifiable criteria (Gou et al., 2025). Each rubric tree consists of internal nodes representing high-level evaluation aspects and leaf nodes defining specific binary verification criteria. The SQL evaluation rubric (Figure 3) assesses query construction across four primary dimensions: Patient Cohort Construction, Medical Concept Implementation, Database Integration, and Clinical Analytics, while the results rubric (Figure 4) focuses on output validation and clinical value assessment. Following practices in automated evaluation (Starace et al., 2025), we implement three key structural components: (1) *Critical vs. Non-Critical Nodes*, where critical nodes represent essential requirements whose failure immediately causes parent failure, while non-critical nodes allow partial scoring; (2) *Sequential Dependencies*, where sequential nodes indicate dependencies and earlier failures short-circuit subsequent evaluations; (3) *Weighted Scoring*, where each node is assigned a weight from 1 to 3 based on importance: 1 denotes basic supportive criteria, 2 indicates standard requirements, and 3 marks critical elements that are essential to validity and require substantial domain expertise.

Our scoring system employs Critical-First Scor-

Statistics	Easy	Med.	Hard
Total Examples	190	254	189
Avg. Question Length	22.86	36.04	45.52
Avg. SQL Tokens	158.79	452.53	615.62
Evaluation Trees			
SQL (Nodes/Depth)	14.80 / 3.05	18.27 / 3.06	19.03 / 3.11
SQL Avg. Words	15.55	16.97	17.74
Results (Nodes/Depth)	10.01 / 4.00	19.62 / 4.02	23.54 / 4.04
Results Avg. Words	6.52	8.21	7.98

Table 3: Basic statistics of CLINSQL.

ing adapted from recent agentic evaluation frameworks (Gou et al., 2025), detailed in Algorithm 1.

3.4 Data Validation

Each annotated example undergoes comprehensive validation by an expert annotator within the medical research field. The validation framework examines four critical aspects: clinical question assessment evaluates real-world relevance, medical terminology accuracy, and linguistic quality; SQL implementation review verifies correct MIMIC-IV database (Johnson et al., 2023) standards and technical execution; output verification confirms structural integrity and medical plausibility; evaluation framework review ensures comprehensive component coverage and unambiguous scoring standards. Validators revise examples with minor issues or reject those with significant problems. Upon passing all validation checks, examples receive "Validated" status for dataset inclusion. To assess problem difficulty and provide fine-grained evaluation of model capabilities, we stratify our dataset into three difficulty levels based on SQL complexity and clinical reasoning requirements: (1) *Easy* (30%): Few-table queries with basic filtering and clinical concepts; (2) *Medium* (40%): Multi-table joins with temporal filtering and moderate reasoning; (3) *Hard* (30%): Complex multi-join queries with nested subqueries and advanced logic. Table 3 presents the data statistics of CLINSQL.

4 Evaluation Protocol

4.1 SQL Analysis using Evaluation Guideline

SQL evaluation employs a rubric-driven process that decomposes each candidate query into clinically relevant checks. The guideline distinguishes critical from supportive requirements and encodes sequential dependencies such that subsequent reasoning is evaluated only if prerequisite steps are satisfied. Candidates must construct an appropriate cohort and map clinical concepts to schema and codes; subsequently, table relationships, join keys, type handling, grouping, and aggregation are verified, followed by task-specific interpretation. Leaf criteria receive binary decisions. Scores are aggregated with a critical-first rule: any failed critical node collapses its parent, whereas non-critical checks contribute via weighted averaging only after critical prerequisites are met. Consistency is ensured by employing GPT-5 as the judge, which leverages strong instruction following and long-context capacity to compare rubric text, gold SQL, and schema hints. Concise rationales are recorded for each leaf decision to support error analysis.

4.2 SQL Execution Result Evaluation

The execution-level evaluation examines CSV outputs produced by executing the candidate SQL. Format compliance is first enforced, including file presence, exact column names, absence of nulls, and basic type checks; clinical plausibility is then assessed using per-column value ranges grounded in the cohort and task definition. Plausible bands admit clinically equivalent answers, whereas acceptable bands tighten tolerance; these criteria support equivalence across alternative but valid formulations. Sequential gating prevents downstream clinical judgments from obscuring upstream format defects. Leaf decisions are binary and are aggregated under the same critical-first rule. The same GPT-5 judge issues decisions and concise rationales by comparing the CSV with the rubric and gold references, yielding interpretable discrepancies in schema adherence, unit handling, rounding, and cohort-conditioned statistics.

The LLM-as-Judge prompts are provided in Appendix O. We provide the reliability and reconciliation analyses of our proposed rubric-based evaluation in Appendix K and L, with SQL-execution divergence statistics summarized in Appendix M.

5 Experiment

This section discusses the experiment setup and our experiment results and analysis.

5.1 Experiment Setup

We evaluate all models on CLINSQL using rubric-based metrics specifically designed for clinical text-to-SQL tasks. Our primary evaluation metrics are the **SQL Score** and the **Execution Score**. We consider the following categories of models: (1)

Model	Test Set						Avg. Validation		Avg. Test	
	Easy		Medium		Hard					
	SQL	Exec	SQL	Exec	SQL	Exec	SQL	Exec	SQL	Exec
Proprietary Models										
GPT-5-mini	54.07	81.31	37.12	73.46	38.94	69.69	42.30	75.16	42.72	74.67
Gemini-2.5-Pro	53.15	85.46	46.34	69.89	42.71	67.22	45.31	76.14	47.28	73.73
GPT-5	58.56	73.79	41.29	66.94	39.58	65.08	42.62	66.52	45.93	68.42
GPT-4.1	59.54	71.20	42.30	68.55	38.25	63.39	44.69	69.92	46.23	67.79
Gemini-2.5-Flash	55.86	72.64	45.61	64.05	41.67	58.72	47.06	66.75	47.48	65.01
OpenAI o4-mini	54.62	67.96	36.59	58.37	34.13	51.68	39.41	59.07	41.23	59.22
Grok-4-Fast-Reason.	49.47	70.82	40.34	53.18	39.40	53.23	42.67	56.58	42.78	58.46
GPT-5-nano	40.83	55.42	34.11	54.67	34.17	45.14	33.18	51.58	36.13	52.03
Mistral-Medium	42.61	56.05	32.81	42.50	31.47	37.68	32.71	43.81	35.33	45.10
Grok-4-Fast-Non-Reason.	54.16	39.31	32.33	29.80	35.69	22.39	34.77	30.10	39.85	30.41
Open-source Models										
DeepSeek-R1	45.32	75.73	45.91	68.43	43.16	63.59	42.63	69.79	44.91	69.15
DeepSeek-V3.1	57.54	71.63	38.78	58.38	34.83	52.99	38.90	61.46	43.19	60.71
Qwen3-235B-A22B-Ins.	38.14	71.85	37.72	54.48	32.77	51.05	36.24	58.15	36.36	58.63
Qwen3-Coder-480B-A35B-Ins.	48.97	64.58	36.00	56.04	33.27	54.69	35.54	60.51	39.05	58.18
Qwen3-Next-80B-A3B-Ins.	48.34	67.85	29.93	45.83	26.81	35.38	34.48	43.41	34.48	49.26
Qwen3-235B-A22B-Think.	34.71	55.51	43.41	44.72	34.71	46.67	38.08	51.11	38.20	48.54
Llama-4-Maverick-17B-128E-Ins.	39.82	59.14	27.23	47.42	20.98	36.01	29.36	51.63	29.11	47.49
Llama-4-Scout-17B-16E-Ins.	40.79	36.57	24.04	28.69	20.23	26.55	26.88	31.44	27.89	30.40
Qwen3-Next-80B-A3B-Think.	46.07	27.73	37.30	30.66	31.25	21.36	37.77	29.06	38.13	27.01
Baichuan-M2-32B	36.09	23.10	30.44	13.24	23.17	10.11	26.60	11.40	29.97	15.27
MedGemma-27B	32.66	6.52	16.60	3.12	14.92	2.65	21.03	4.46	20.92	4.00
SQLCoder-7B-2	6.65	0.00	6.50	0.00	2.30	0.00	3.99	0.00	5.29	0.00

Table 4: SQL score and execution score (%) on CLINSQL validation and test sets using CoT prompting with self-refinement. Scenario-level scores are presented in Appendix H.

Open-source general-purpose LLMs: DeepSeek-R1 (DeepSeek-AI et al., 2025a), DeepSeek-V3.1 (DeepSeek-AI et al., 2025b), Qwen3-Coder series, Qwen3-Instruct series, Qwen3-Thinking series (Qwen Team, 2025), Llama-4 series (Meta AI, 2024). (2) **Proprietary models:** GPT-5 series (OpenAI, 2025b), Gemini-2.5 series (Comanici et al., 2025), GPT-4.1 (OpenAI, 2025a), o4-mini (OpenAI, 2025c), Grok-4-Fast series (xAI, 2025b,a), and Mistral-Medium (Mistral AI, 2025). (3) **Text-to-SQL models:** SQLCoder-7B-2 (Defog.ai, 2024). (4) **Medical-domain LLMs:** MedGemma-27B (Google DeepMind, 2025) and Baichuan-M2-32B (Baichuan AI, 2025). For open-source models, we perform inference using `vLLM` pipeline (Kwon et al., 2023), while proprietary models are accessed through official APIs.

We evaluate models under two prompting regimes: Direct Output and Chain-of-Thought (CoT). In both regimes, the model must return a single executable BigQuery query. If execution fails, we run up to two *self-refinement* rounds that feed the question, the prior SQL, and the BigQuery error back to the model with minimal-edit instructions. We then extract the final fenced SQL block and execute it. We apply *self-refinement* to both regimes because many models have low first-pass execution success, and a single correction round is insufficient; Appendix J reports the attempt-wise success rates that motivate this choice. Prompt templates for both regimes and the refinement procedure are provided in Appendix D. Parameter settings and model configurations appear in Appendix E.

5.2 Main Findings

Table 4 presents SQL and execution scores on CLINSQL. We highlight the following findings:

CLINSQL presents substantial challenges for current foundation models. While GPT-5-mini achieves the best average execution score, performance on the Hard split remains modest: leading proprietary models stay under 70% (*e.g.,* GPT-5-mini 69.7% and Gemini-2.5-Pro 67.2%). Gemini-2.5-Pro also drops by 18.24% from Easy to Hard.

Open-sourced models performance. DeepSeek-R1 attains 69.2% average test execution with a 44.9% SQL score, and it surpasses several proprietary baselines, including o4-mini and both Grok-4 variants. However, open-source models still lag the strongest proprietary models: the best proprietary model (GPT-5-mini) reaches 74.7% execu-

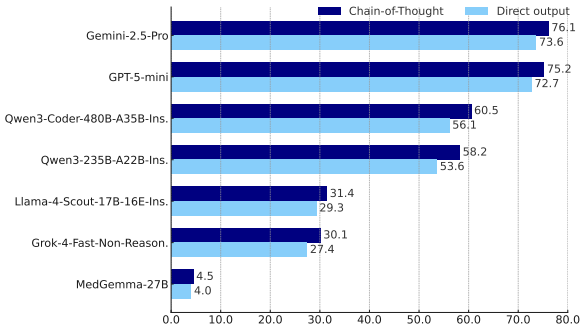

Figure 5: Execution score comparison on the validation set for representative models. Full SQL and execution comparisons for all models are provided in Appendix G.

tion, about 5.5 points higher than DeepSeek-R1 and roughly 14 to 16 points ahead of DeepSeek-V3.1 (60.7%) and Qwen3-Coder-480B-A35B-Instruct (58.2%). Even so, these results show the gap is narrowing as techniques mature.

CoT reasoning generally improves model performance compared to directly outputting the final SQL. As shown in Figure 5, the extent of improvement varies across models. Qwen3-235B-A22B-Instruct increases from 53.6% to 58.2% and Qwen3-Coder-480B-A35B-Instruct from 56.1% to 60.5%, while Llama-4-Scout-17B-16E-Instruct rises only from 29.3% to 31.4% and Grok-4-Fast-Non-Reasoning from 27.4% to 30.1%. Strong proprietary models see modest gains (Gemini-2.5-Pro from 73.6% to 76.1%, GPT-5-mini from 72.7% to 75.2%). Even lower baselines benefit (MedGemma-27B from 4.0% to 4.5%).

5.3 Error Analysis and Case Study

To better characterize failure modes, we randomly select 10 cases from each of six validation scenarios generated by GPT-5-mini and analyze rubric feedback from our judge model. We observe three common error types: **Cohort specification & coding** (54%): in which explicit ICD or `itemid` constraints are replaced by keyword heuristics or key joins are relaxed, broadening cohorts; **Output schema & formatting** (24%): omitted required columns, invalid values, or naming mismatches that trigger schema checks; and **Aggregation & clinical statistics** (14%): mis-specified denominators or missing normalization leading to implausible rates. **Other observed errors** include occasional temporal boundary mistakes and pattern-specific issues seen in GPT-5-mini outputs, such as using quartiles instead of percentiles in `APPROX_QUANTILES` (*e.g.,* employing quartile buckets rather than 100-quantile offsets), which yields incorrect reported statistics. Examples for each error type are provided in Appendix C.

Setting	Easy	Medium	Hard
SQL Score			
Baseline CoT	54.78	36.40	37.90
Schema-hinted CoT	58.27$^{\uparrow 3.49}$	41.09$^{\uparrow 4.69}$	42.04$^{\uparrow 4.14}$
Execution Score			
Baseline CoT	79.96	75.63	69.83
Schema-hinted CoT	82.74$^{\uparrow 2.78}$	85.85$^{\uparrow 10.21}$	77.03$^{\uparrow 7.20}$

Table 5: Validation performance of GPT-5-mini under baseline and *schema-hinted* Chain-of-Thought configurations. Superscripts denote absolute percentage-point gains of the schema-hinted setting over the baseline.

5.4 Schema-Hinted Inference Analysis

The preceding failure analysis underscores cohort drift and schema mismatches as dominant error sources. This motivates our exploration of a Schema-Hinted inference configuration designed to mitigate these common failures. The setting augments the standard Chain-of-Thought prompting and self-refinement schedule with schema hints, foregrounding clinically validated ICD filters and expected result columns. Full setup details are provided in Appendix F. We evaluate this configuration with GPT-5-mini on the validation split, and the results show consistent gains over the baseline. As summarised in Table 5, SQL and execution accuracy improve across all difficulty tiers. The most pronounced execution gains are observed on medium and hard queries, where providing clinically validated ICD filters and expected result columns better constrains the inference process.

6 Conclusion

We propose CLINSQL, a benchmark for realistic clinical text-to-SQL analytics. It captures core challenges of real EHR practice, including heterogeneous tables, temporal windows, and patient-similarity cohort construction. We assess a broad set of models using the developed rubric-based evaluation protocols and observe that, despite recent advances, performance remains well short of clinically reliable operation: execution frequently exceeds SQL correctness and errors cluster around cohort specification, schema/formatting, and aggregation/clinical statistics. CLINSQL establishes a rigorous, domain-grounded target for clinical research and advance trustworthy EHR analytics.

Limitations

While CLINSQL advances clinically grounded text-to-SQL evaluation, several limitations remain. First, the current benchmark is built on MIMIC-IV v3.1—data from a single health system—and targets a single SQL environment, which may limit transferability to other EHR ecosystems, data models, and database backends. Second, CLINSQL depends on substantial domain-expert involvement for scenario specification, gold-standard SQL authoring, and rubric-aligned leaf rationales. Although this expert curation provides high-fidelity supervision, the associated training, annotation, and review burden reduces throughput and makes it difficult to scale to substantially larger datasets without additional tooling or alternative supervision strategies.

References

Seongsu Bae, Sunjun Kweon, Taewoong Jang, and Edward Choi. 2023. Ehrxqa: A multi-modal question answering dataset for electronic health records with chest x-ray images. *arXiv preprint arXiv:2310.18652*.

Baichuan AI. 2025. Baichuan-M2-32B. https://huggingface.co/baichuan-inc/Baichuan-M2-32B. Model card.

Jayetri Bardhan, Anthony Colas, Kirk Roberts, and Daisy Zhe Wang. 2022. DrugEHRQA: A question answering dataset on structured and unstructured electronic health records for medicine related queries. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1083–1097, Marseille, France. European Language Resources Association.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025. Benchmarking large language models on answering and explaining challenging medical questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3563–3599, Albuquerque, New Mexico. Association for Computational Linguistics.

Gheorghe Comanici, many others, and the Gemini Team. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025a. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang,

Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025b. Deepseek-v3 technical report.

Defog.ai. 2024. SQLCoder-7B-2. `https://huggingface.co/defog/sqlcoder-7b-2`. Model card.

Google DeepMind. 2025. MedGemma 27B Text-only (medgemma-27b-text-it). `https://huggingface.co/google/medgemma-27b-text-it`. Model card.

Boyu Gou, Zanming Huang, Yuting Ning, Yu Gu, Michael Lin, Weijian Qi, Andrei Kopanev, Botao Yu, Bernal Jiménez Gutiérrez, Yiheng Shu, Chan Hee Song, Jiaman Wu, Shijie Chen, Hanane Nour Moussa, Tianshu Zhang, Jian Xie, Yifei Li, Tianci Xue, Zeyi Liao, Kai Zhang, Boyuan Zheng, Zhaowei Cai, Viktor Rozgic, Morteza Ziyadi, Huan Sun, and Yu Su. 2025. Mind2web 2: Evaluating agentic search with agent-as-a-judge.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019a. Pubmedqa: A dataset for biomedical research question answering. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019b. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. 2023. Mimic-iv, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1–9.

Zong Ke, Yuqing Cao, Zhenrui Chen, Yuchen Yin, Shouchao He, and Yu Cheng. 2025a. Early warning of cryptocurrency reversal risks via multi-source data. *Finance Research Letters*, page 107890.

Zong Ke, Jiaqing Shen, Xuanyi Zhao, Xinghao Fu, Yang Wang, Zichao Li, Lingjie Liu, and Huailing Mu. 2025b. A stable technical feature with gru-cnn-ga fusion. *Applied Soft Computing*, page 114302.

Hajung Kim, Chanhwi Kim, Hoonick Lee, Kyochul Jang, Jiwoo Lee, Kyungjae Lee, Gangwoo Kim, and Jaewoo Kang. 2024a. KU-DMIS at EHRSQL 2024 : Generating SQL query via question templatization in EHR. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 672–686, Mexico City, Mexico. Association for Computational Linguistics.

Yunsoo Kim, Jinge Wu, Yusuf Abdulle, and Honghan Wu. 2024b. MedExQA: Medical question answering benchmark with multiple explanations. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, pages 167–181, Bangkok, Thailand. Association for Computational Linguistics.

Sunjun Kweon, Jiyoun Lee, Joon Sung Yi, and Edward Choi. 2024. Ehrnoteqa: An llm benchmark for real-world clinical practice using discharge summaries. *arXiv preprint arXiv:2402.16040*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. Kaggledbqa: Realistic evaluation of text-to-sql parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. 2022. Ehrsql: A practical text-to-sql benchmark for electronic health records. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Gyubok Lee, Sunjun Kweon, Seongsu Bae, and Edward Choi. 2024. Overview of the EHRSQL 2024 shared task on reliable text-to-SQL modeling on electronic health records. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 644–654, Mexico City, Mexico. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sql evaluation. *Advances in Neural Information Processing Systems*, 36.

Meta AI. 2024. Introducing llama 4: Advancing multimodal intelligence.

Mistral AI. 2025. Mistral medium: Model overview and documentation. Product Documentation.

OpenAI. 2025a. Gpt-4.1 and gpt-4.1 mini release notes. Release notes (14 May 2025) announcing GPT-4.1 and GPT-4.1 mini; GPT-4.1 excels at coding and precise instruction following, while GPT-4.1 mini is a fast, efficient model with a 1 M-token context window.

OpenAI. 2025b. Introducing GPT-5 for developers. Announces API model sizes including `gpt-5-mini`.

OpenAI. 2025c. Openai o4-mini: reasoning model release. Blog post (16 Apr 2025) announcing the o3 and o4-mini models; o4-mini is a small model optimized for fast, cost-efficient reasoning.

Justice Ou, Tinglin Huang, Yilun Zhao, Ziyang Yu, Peiqing Lu, and Rex Ying. 2025. Experience retrieval-augmentation with electronic health records enables accurate discharge qa.

Kaichen Ouyang, Zong Ke, Shengwei Fu, Lingjie Liu, Puning Zhao, and Dayu Hu. 2024. Learn from global correlations: Enhancing evolutionary algorithm via spectral gnn. *arXiv preprint arXiv:2412.17629*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. *Proceedings of the Conference on Health, Inference, and Learning*, pages 248–260.

Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. 2018. emrqa: A large corpus for question answering on electronic medical records. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2357–2368.

Qwen Team. 2025. Qwen3 technical report.

Jaehee Ryu, Seonhee Cho, Gyubok Lee, and Edward Choi. 2024. EHR-SeqSQL : A sequential text-to-SQL dataset for interactively exploring electronic health records. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16388–16407, Bangkok, Thailand. Association for Computational Linguistics.

Shadi Saleh and Pavel Pecina. 2019. An extended CLEF ehealth test collection for cross-lingual information retrieval in the medical domain. In *Advances in Information Retrieval: 41st European Conference on Information Retrieval (ECIR 2019)*, volume 11438 of *Lecture Notes in Computer Science*, pages 188–195. Springer.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. 2022. Large language models encode clinical knowledge.

Sithursan Sivasubramaniam, Cedric Osei-Akoto, Yi Zhang, Kurt Stockinger, and Jonathan Fuerst. 2024. SM3-text-to-query: Synthetic multi-model medical text-to-query benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Siddharth Soni, Kirk Roberts, Daqing Wang, Arvind Subburathinam, and Sivaramakrishnan Long. 2022. Radqa: A question answering dataset to improve comprehension of radiology reports. In *Proceedings of the Language Resources and Evaluation Conference*, pages 6567–6577.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. 2025. Paperbench: Evaluating ai's ability to replicate ai research.

Hanna Suominen, Liadh Kelly, Lorraine Goeuriot, and Martin Krallinger. 2020. Clef ehealth evaluation lab 2020. In *Advances in Information Retrieval: 42nd European Conference on Information Retrieval (ECIR 2020), Part II*, volume 12036 of *Lecture Notes in Computer Science*, pages 587–594. Springer.

Logesh Kumar Umapathi Wang, Ankit Pal, and Malaikannan Sankarasubbu. 2022. Drugehrqa: A

question answering dataset on structured and unstructured electronic health records. In *Proceedings of the Language Resources and Evaluation Conference*, pages 2789–2797.

Xiaoxuan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2020. Mimicsql: Text-to-sql generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020*, pages 2506–2516.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Fangyu Wei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. 2024. Livebench: A challenging, contamination-limited llm benchmark. *arXiv preprint arXiv:2406.19314*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. Livebench: A challenging, contamination-free LLM benchmark. In *The Thirteenth International Conference on Learning Representations*.

xAI. 2025a. Grok 4 fast (non-reasoning) — model overview. xAI Documentation.

xAI. 2025b. Grok 4 fast (reasoning) — model overview. xAI Documentation/News.

Zheyuan Yang, Lyuhao Chen, Arman Cohan, and Yilun Zhao. 2025. Table-r1: Inference-time scaling for table reasoning tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20605–20624, Suzhou, China. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiko Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Yifan Zhang, Jingqing Chen, Qiao Yuan, Zhihao Ding, Huaishao Luo, Kaixiong Pan, Mengzhuo Zhang, Haiyang Yu, Qingyun Chen, Xiangru Wang, et al. 2025. Medxpertqa: Benchmarking expert-level medical reasoning and understanding. *arXiv preprint arXiv:2501.18362*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. In *arXiv preprint arXiv:1709.00103*.

A CLINSQL Benchmark Construction

ID	Year	Major	Assigned Scenario	Author?
1	3rd-year PhD	Health Informatics	Patient Demographics & Admissions	✗
2	—	—	—	✓
3	5th-year PhD	Biochemistry	Laboratory Results Analysis	✗
4	3rd-year PhD	Medicine	Medication Management	✗
5	4th-year PhD	Biomedical Engineering	Diagnostic Procedures	✗
6	—	—	—	✓

Table 6: Overview of the 6 expert annotators who contributed to the CLINSQL Benchmark Construction. Two annotators (IDs 2 and 6) are paper authors; to preserve confidentiality, their rows omit year, major, and scenario details.

B Example Clinical Questions

B.1 Patient Demographics Example

Query For an 81-year-old female: among female Medicare patients aged 76–86 transferred from another hospital with principal AMI (ICD-9 410*/ICD-10 I21*), report 30-day readmission rate; median index LOS for readmitted vs not; percent index stays > 4 days.

SQL As Figure 6

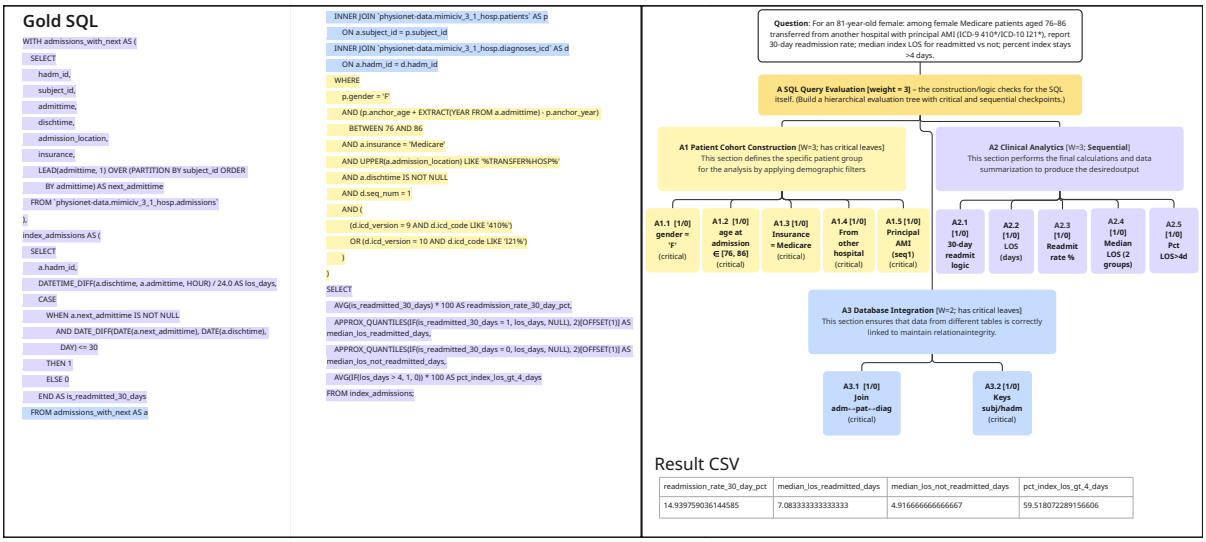

Figure 6: SQL evaluation rubric tree and Result for a CLINSQL sample: Patient Demographics.

B.2 Vital Signs Monitoring Example

Query I have a 60-year-old man in the ICU. In male ICU patients aged 55–65 with HFNC within 24 hours versus condition-matched ICU controls, what are the instability score median and p25/p75/p95, tachycardia and hypotension burden, ICU LOS and mortality?

SQL As Figure 7

B.3 Laboratory Results Analysis Example

Query I have a 51-year-old female with suspected ACS. Among female ACS admissions age 46–56, what are counts, percentages, and mean hospital length of stay for first hs-TnT: Normal, Borderline, Myocardial Injury?

SQL As Figure 8

B.4 Medication Management Example

Query I have a 64-year-old female inpatient. Among females aged 59–69, what's the IQR of single inpatient amiodarone prescription durations (days)?

SQL As Figure 9

B.5 Diagnostic Procedures Example

Query Evaluating an 88-year-old man: among male patients aged 83–93 with sepsis on their first ICU stay, stratify first-72-hour diagnostic intensity (distinct procedures) into quartiles and report mean procedure counts, mean ICU LOS in days, and mortality (%) per quartile.

SQL As Figure 10

B.6 Disease Diagnosis and Outcomes Example

Query I have a 75-year-old female inpatient with pulmonary embolism. For female inpatients aged 70–80 with PE, stratify into risk-score quintiles and report per quintile: 90-day mortality, general 70–80 female 90-day mortality (comparison), AKI and ARDS rates, and median survivor LOS.

SQL As Figure 11

Gold SQL

```
WITH
 base_cohort AS (
  SELECT
   icu.subject_id,
   icu.hadm_id,
   icu.stay_id,
   icu.intime,
   icu.outtime,
   DATETIME_DIFF(icu.intime, DATETIME(pat.anchor_year, 1, 1, 0, 0, 0),
    YEAR) + pat.anchor_age AS age_at_icustay,
   adm.hospital_expire_flag,
   DATETIME_DIFF(icu.outtime, icu.intime, HOUR) / 24.0 AS icu_los_days
  FROM
   `physionet-data.mimiciv_3_1_icu.icustays` AS icu
  INNER JOIN
   `physionet-data.mimiciv_3_1_hosp.patients` AS pat
   ON icu.subject_id = pat.subject_id
  INNER JOIN
   `physionet-data.mimiciv_3_1_hosp.admissions` AS adm
   ON icu.hadm_id = adm.hadm_id
  WHERE
   pat.gender = 'M'
   AND (
    DATETIME_DIFF(icu.intime, DATETIME(pat.anchor_year, 1, 1, 0, 0, 0),
      YEAR)
    + pat.anchor_age
   ) BETWEEN 55 AND 65
 ),
 hfnc_stays AS (
  SELECT DISTINCT
   ce.stay_id
  FROM
   `physionet-data.mimiciv_3_1_icu.chartevents` AS ce
  INNER JOIN
   base_cohort AS cohort
   ON ce.stay_id = cohort.stay_id
  WHERE
   ce.itemid = 227287
   AND DATETIME_DIFF(ce.charttime, cohort.intime, HOUR) <= 24
 ),
 vitals_first_24h AS (
  SELECT
   ce.stay_id,
   CASE
    WHEN ce.itemid = 220045
    THEN 'HR'
    WHEN ce.itemid IN (220052, 220181)
    THEN 'MAP'
   END AS vital_name,
   ce.valuenum AS value
  FROM
   `physionet-data.mimiciv_3_1_icu.chartevents` AS ce
  INNER JOIN
   base_cohort AS cohort
   ON ce.stay_id = cohort.stay_id
  WHERE
   ce.itemid IN (220045, 220052, 220181)
   AND ce.valuenum IS NOT NULL AND ce.valuenum > 0
   AND DATETIME_DIFF(ce.charttime, cohort.intime, HOUR) <= 24),
 stay_level_metrics AS (
  SELECT
   stay_id,
   (SAFE_DIVIDE(
     STDDEV_SAMP(IF(vital_name = 'HR', value, NULL)),
     AVG(IF(vital_name = 'HR', value, NULL)))
   ) + (
    SAFE_DIVIDE(STDDEV_SAMP(IF(vital_name = 'MAP', value, NULL)),
     AVG(IF(vital_name = 'MAP', value, NULL)))
   ) AS instability_score,
   SAFE_DIVIDE(
    COUNTIF(vital_name = 'HR' AND value > 100),
    COUNTIF(vital_name = 'HR')
   ) AS tachycardia_burden,
   SAFE_DIVIDE(
    COUNTIF(vital_name = 'MAP' AND value < 65),
    COUNTIF(vital_name = 'MAP')
   ) AS hypotension_burden
  FROM
   vitals_first_24h
  GROUP BY
   stay_id
  HAVING
   COUNTIF(vital_name = 'HR') > 2 AND COUNTIF(vital_name = 'MAP') > 2
 ),
 final_cohort_data AS (
  SELECT
   bc.stay_id,
   bc.icu_los_days,
   bc.hospital_expire_flag,
   CASE
    WHEN hs.stay_id IS NOT NULL
    THEN 'HFNC_Target'
    ELSE 'Control'
   END AS cohort_group,
   slm.instability_score,
   slm.tachycardia_burden,
   slm.hypotension_burden
  FROM
   base_cohort AS bc
  LEFT JOIN
   hfnc_stays AS hs
   ON bc.stay_id = hs.stay_id
  INNER JOIN
   stay_level_metrics AS slm
   ON bc.stay_id = slm.stay_id
 )
SELECT
 cohort_group,
 COUNT(DISTINCT stay_id) AS number_of_stays,
 AVG(instability_score) AS avg_instability_score,
 APPROX_QUANTILES(instability_score, 100) [OFFSET(25)] AS instability_score_p25,
 APPROX_QUANTILES(instability_score, 100) [OFFSET(50)] AS instability_score_median,
 APPROX_QUANTILES(instability_score, 100) [OFFSET(75)] AS instability_score_p75,
 APPROX_QUANTILES(instability_score, 100) [OFFSET(95)] AS instability_score_p95,
 AVG(tachycardia_burden) AS avg_tachycardia_burden_proportion,
 AVG(hypotension_burden) AS avg_hypotension_burden_proportion,
 AVG(icu_los_days) AS avg_icu_los_days,
 AVG(CAST(hospital_expire_flag AS INT64)) AS mortality_rate
FROM  final_cohort_data GROUP BY cohort_group ORDER BY cohort_group DESC;
```

Figure 7: SQL sample: Vital Signs Monitoring.

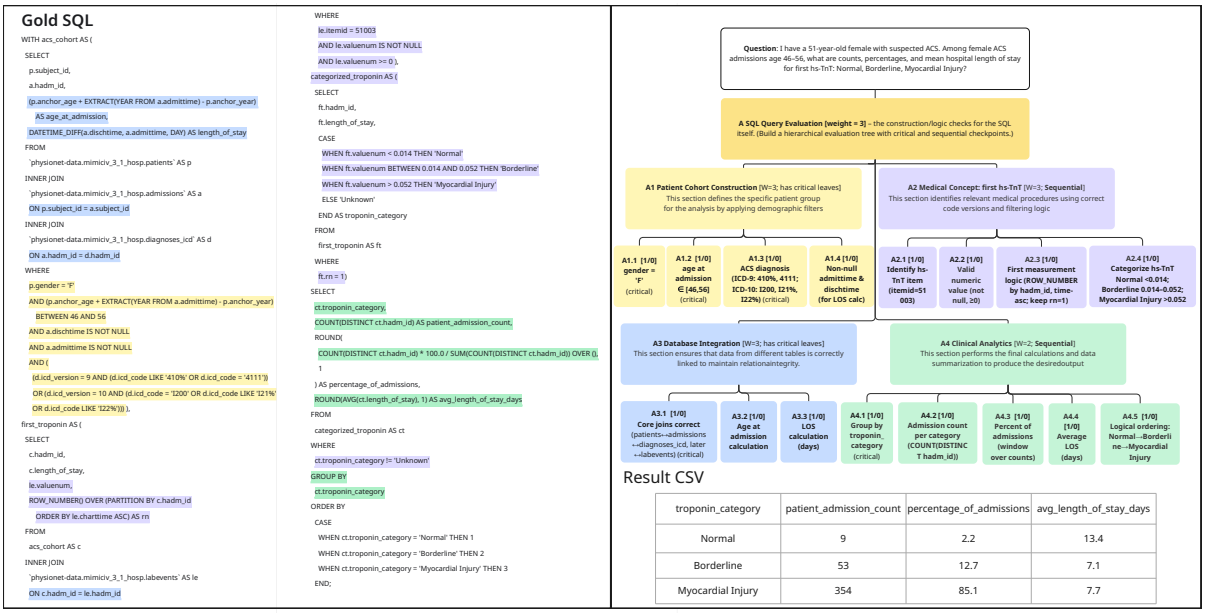

Figure 8: SQL evaluation rubric tree and Result for a CLINSQL sample: Laboratory Results Analysis.

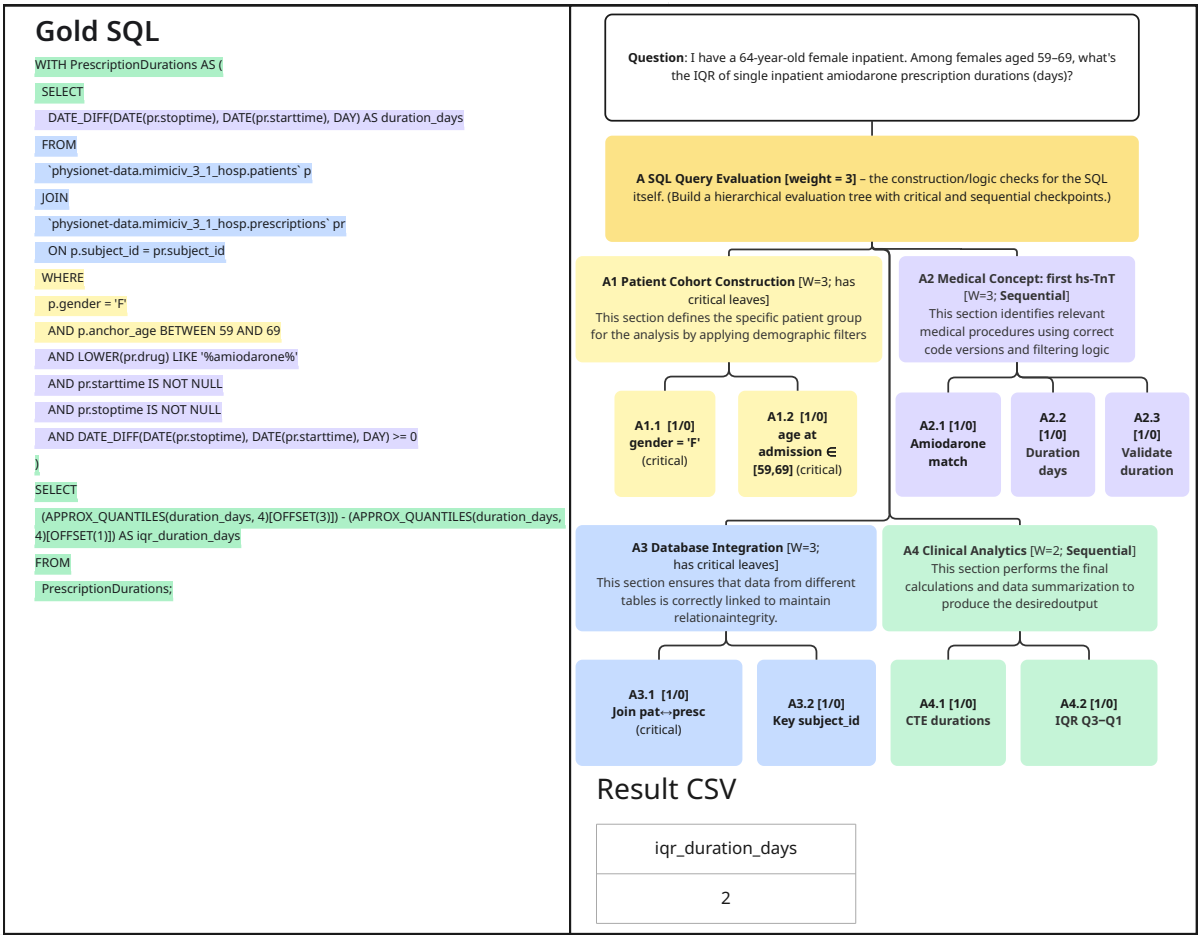

Figure 9: SQL evaluation rubric tree and Result for a CLINSQL sample: Medication Management.

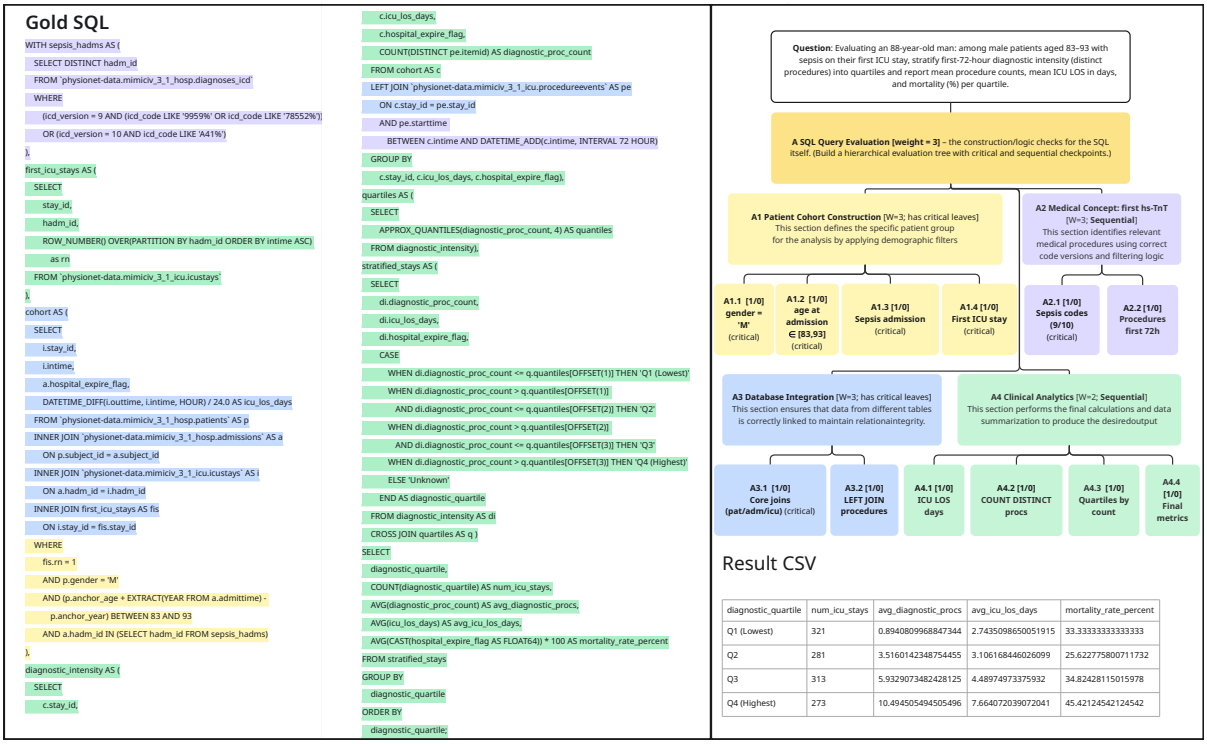

Figure 10: SQL evaluation rubric tree and Result for a CLINSQL sample: Diagnostic Procedures.

Gold SQL

```
WITH
  pe_admissions AS (
    SELECT
      hadm_id
    FROM
      `physionet-data.mimiciv_3_1_hosp.diagnoses_icd`
    WHERE
      (icd_version = 10 AND icd_code LIKE 'I26%')
      OR (icd_version = 9 AND icd_code LIKE '415.1%')
    GROUP BY
      hadm_id
  ),
  cohort_base AS (
    SELECT pat.subject_id, adm.hadm_id,
      (pat.anchor_age + EXTRACT(YEAR FROM adm.admittime) - pat.anchor_year) AS
age_at_admission,
      adm.admittime,
      adm.dischtime,
      COALESCE(adm.deathtime, pat.dod) AS deathtime
    FROM
      `physionet-data.mimiciv_3_1_hosp.patients` AS pat
    INNER JOIN
      `physionet-data.mimiciv_3_1_hosp.admissions` AS adm ON pat.subject_id = adm.subject_id
    INNER JOIN
      pe_admissions AS pe ON adm.hadm_id = pe.hadm_id
    WHERE
      pat.gender = 'F'
      AND (pat.anchor_age + EXTRACT(YEAR FROM adm.admittime) - pat.anchor_year) BETWEEN 70
AND 80
  ),
  diagnoses_flags AS (
    SELECT
      dx.hadm_id,
      MAX(IF((dx.icd_version = 10 AND dx.icd_code IN ('R65.21', 'A41.9')) OR (dx.icd_version = 9 AND
dx.icd_code IN ('995.92', '038.9')), 1, 0)) AS has_sepsis,
      MAX(IF((dx.icd_version = 10 AND dx.icd_code LIKE 'I21%') OR (dx.icd_version = 9 AND
dx.icd_code LIKE '410%'), 1, 0)) AS has_mi,
      MAX(IF((dx.icd_version = 10 AND dx.icd_code LIKE 'N18%') OR (dx.icd_version = 9 AND
dx.icd_code LIKE '585%'), 1, 0)) AS has_ckd,
      MAX(IF((dx.icd_version = 10 AND STARTS_WITH(dx.icd_code, 'C')) OR (dx.icd_version = 9 AND
dx.icd_code BETWEEN '140' AND '209'), 1, 0)) AS has_cancer,
      MAX(IF((dx.icd_version = 10 AND dx.icd_code LIKE 'N17%') OR (dx.icd_version = 9 AND
dx.icd_code LIKE '584%'), 1, 0)) AS has_aki,
      MAX(IF((dx.icd_version = 10 AND dx.icd_code = 'J80') OR (dx.icd_version = 9 AND dx.icd_code IN
('518.82', '518.5')), 1, 0)) AS has_ards
    FROM
      `physionet-data.mimiciv_3_1_hosp.diagnoses_icd` AS dx
    WHERE
      dx.hadm_id IN (SELECT hadm_id FROM cohort_base)
    GROUP BY
      dx.hadm_id
  ),
  cohort_features AS (
    SELECT
      cb.hadm_id,
      DATETIME_DIFF(cb.dischtime, cb.admittime, DAY) AS los_days,
      (cb.deathtime IS NOT NULL AND DATETIME_DIFF(cb.deathtime, cb.admittime, DAY) <= 90) AS
is_dead_at_90_days,
      COALESCE(df.has_aki, 0) AS has_aki,
      COALESCE(df.has_ards, 0) AS has_ards,
      (
        (cb.age_at_admission - 70) * 2
        + (COALESCE(df.has_sepsis, 0) * 25)
        + (COALESCE(df.has_cancer, 0) * 20)
        + (COALESCE(df.has_mi, 0) * 15)
        + (COALESCE(df.has_ckd, 0) * 10)
      ) AS risk_score
    FROM
      cohort_base AS cb
    LEFT JOIN
      diagnoses_flags AS df ON cb.hadm_id = df.hadm_id
  ),
  risk_stratification AS (
    SELECT
      *,
      NTILE(5) OVER (ORDER BY risk_score) AS risk_quintile
    FROM
      cohort_features
  ),
  general_pop_mortality AS (
    SELECT
      SAFE_DIVIDE(
        COUNTIF(cb.deathtime IS NOT NULL AND DATETIME_DIFF(cb.deathtime, cb.admittime, DAY)
<= 90),
        COUNT(cb.hadm_id)
      ) AS general_pop_90d_mortality_rate
    FROM (
      SELECT
        adm.hadm_id,  adm.admittime,
        COALESCE(adm.deathtime, pat.dod) AS deathtime
      FROM
        `physionet-data.mimiciv_3_1_hosp.patients` AS pat
      INNER JOIN
        `physionet-data.mimiciv_3_1_hosp.admissions` AS adm ON pat.subject_id = adm.subject_id
      WHERE
        pat.gender = 'F'
        AND (pat.anchor_age + EXTRACT(YEAR FROM adm.admittime) - pat.anchor_year) BETWEEN 70
AND 80
    ) AS cb)
SELECT
  rs.risk_quintile,
  COUNT(rs.hadm_id) AS total_patients,
  MIN(rs.risk_score) AS min_risk_score,
  MAX(rs.risk_score) AS max_risk_score,
  SAFE_DIVIDE(SUM(IF(rs.is_dead_at_90_days, 1, 0)), COUNT(rs.hadm_id)) AS
pe_cohort_90d_mortality_rate,
  gpm.general_pop_90d_mortality_rate,
  AVG(rs.has_aki) AS aki_rate,
  AVG(rs.has_ards) AS ards_rate,
  APPROX_QUANTILES(
    IF(NOT rs.is_dead_at_90_days, rs.los_days, NULL), 100 IGNORE NULLS
  )[OFFSET(50)] AS median_survivor_los_days
FROM
  risk_stratification AS rs
CROSS JOIN
  general_pop_mortality AS gpm
GROUP BY
  rs.risk_quintile,
  gpm.general_pop_90d_mortality_rate
ORDER BY
  rs.risk_quintile;
```

Figure 11: SQL sample: Disease Diagnosis and Outcomes.

C Error Analysis

Gold SQL

```
WITH.
  ..
SELECT
  COUNT(DISTINCT subject_id) AS patient_count,
  COUNT(hadm_id) AS admission_count,
  ROUND(AVG(troponin_t_value), 4) AS mean_troponin_t,
  ROUND(APPROX_QUANTILES(troponin_t_value, 100)[OFFSET(50)], 4) AS median_troponin_t,
  ROUND(APPROX_QUANTILES(troponin_t_value, 100)[OFFSET(25)], 4) AS p25_troponin_t,
  ROUND(APPROX_QUANTILES(troponin_t_value, 100)[OFFSET(75)], 4) AS p75_troponin_t,
  ROUND(
    (APPROX_QUANTILES(troponin_t_value, 100)[OFFSET(75)] -
APPROX_QUANTILES(troponin_t_value, 100)[OFFSET(25)]),
    4
  ) AS iqr_troponin_t,
  ROUND(MIN(troponin_t_value), 4) AS min_elevated_troponin_t,
  ROUND(MAX(troponin_t_value), 4) AS max_elevated_troponin_t
FROM
  first_troponin_t
WHERE
  rn = 1
  AND troponin_t_value > 0.01;
```

Failed

```
...
SELECT
  COUNT(DISTINCT e.subject_id) AS patient_count,
  COUNT(DISTINCT e.hadm_id) AS admission_count,
  ROUND(AVG(e.hs_tnt_ng_ml), 6) AS hs_tnt_mean_ng_per_ml,
  q.qarr[OFFSET(2)] AS hs_tnt_median_ng_per_ml,
  q.qarr[OFFSET(1)] AS hs_tnt_q1_ng_per_ml,
  q.qarr[OFFSET(3)] AS hs_tnt_q3_ng_per_ml,
  q.qarr[OFFSET(3)] - q.qarr[OFFSET(1)] AS hs_tnt_iqr_ng_per_ml
FROM eligible_first_troponin e
CROSS JOIN quartiles q;
```

Query:

For a 60-year-old woman: among female patients aged 55–65 admitted with AMI whose first hs-TnT >0.01 ng/mL, report patient and admission counts and hs-TnT mean, median, and IQR.

Analysis:

Error: "SELECT list expression references q.qarr which is neither grouped nor aggregated".
The failed query incorrectly references ungrouped columns in SELECT with CROSS JOIN, while the golden query uses proper aggregation functions.

Gold Result CSV

patient_count	admission_count	mean_troponin_t	median_troponin_t	...
687	808	1.0738	0.29	...

Predicted Result CSV

result
""

Figure 12: A sample of Error Analysis: Output Schema and Formatting

Gold SQL

```
WITH HeartFailureAdmissions AS (
SELECT DISTINCT
  diag.hadm_id
FROM
  `physionet-data.mimiciv_3_1_hosp.patients` p
JOIN
  `physionet-data.mimiciv_3_1_hosp.diagnoses_icd` diag ON p.subject_id = diag.subject_id
WHERE
  p.gender = 'M'
  AND (
    STARTS_WITH(diag.icd_code, '428')
    OR STARTS_WITH(diag.icd_code, 'I50')
  )
),
AdmissionSodium AS (
SELECT
  le.valuenum,
  ROW_NUMBER() OVER(PARTITION BY le.hadm_id ORDER BY le.charttime ASC) as rn
FROM
  `physionet-data.mimiciv_3_1_hosp.labevents` le
JOIN
  HeartFailureAdmissions hfa ON le.hadm_id = hfa.hadm_id
WHERE
  le.itemid = 50983
  AND le.valuenum IS NOT NULL
  AND le.valuenum BETWEEN 120 AND 160
)
SELECT
  MIN(valuenum) AS min_admission_serum_sodium
FROM
  AdmissionSodium
WHERE
  rn = 1;
```

Failed

```
...
sodium_lab_items AS (
 SELECT DISTINCT d.itemid
 FROM `physionet-data.mimiciv_3_1_hosp.d_labitems` d
 WHERE
  d.loinc_code = '2951-2'
  OR (LOWER(d.label) LIKE '%sodium%' AND LOWER(COALESCE(d.fluid, '')) LIKE '%serum%')
)
...
```

Query:

For a 65-year-old man with heart failure, what is the minimum admission serum sodium observed among male heart failure hospitalizations?

Analysis:

The failed query attempts to use a loinc_code field that doesn't exist in the MIMIC-IV schema, while the golden query uses the correct itemid approach for identifying specific lab tests.

Gold Result CSV

min_admission_serum_sodium
120.0

Predicted Result CSV

result
""

Figure 13: A sample of Error Analysis: Cohort Specification and Coding

Gold SQL

```
SELECT
  ROUND(APPROX_QUANTILES(DATE_DIFF(DATE(pr.stoptime), DATE(pr.starttime), DAY), 2)[OFFSET(1)], 2) AS median_treatment_duration_days
FROM `physionet-data.mimiciv_3_1_hosp.patients` p
JOIN `physionet-data.mimiciv_3_1_hosp.prescriptions` pr ON p.subject_id = pr.subject_id
WHERE
 p.gender = 'F'
 AND p.anchor_age BETWEEN 59 AND 69
 AND pr.starttime IS NOT NULL
 AND pr.stoptime IS NOT NULL
 AND DATE_DIFF(DATE(pr.stoptime), DATE(pr.starttime), DAY) > 0
 AND (
  LOWER(pr.drug) LIKE '%amlodipine%' OR
  LOWER(pr.drug) LIKE '%nifedipine%' OR
  LOWER(pr.drug) LIKE '%nicardipine%' OR
  LOWER(pr.drug) LIKE '%felodipine%'
 );
```

Failed

```
...
 WHERE pt.gender = 'F'
  AND pt.anchor_age BETWEEN 59 AND 69
  AND pr.hadm_id IS NOT NULL
  AND pr.starttime IS NOT NULL
  AND pr.stoptime IS NOT NULL
  AND TIMESTAMP(pr.starttime) >= adm.admittime
  AND TIMESTAMP(pr.stoptime) <= adm.dischtime
  AND TIMESTAMP(pr.stoptime) > TIMESTAMP(pr.starttime)
  AND REGEXP_CONTAINS(
   LOWER(COALESCE(pr.drug, '')),
...
```

Query:

In a 64-year-old female inpatient, what is the median duration of a single inpatient dihydropyridine CCB prescription among women 59–69?

Analysis:

Error: "No matching signature for operator >= for argument types: TIMESTAMP, DATETIME"
The failed query mixes TIMESTAMP and DATETIME types in comparisons, while it should use consistent DATE functions throughout.

Gold Result CSV

median_treatment_duration_days
120.0

Predicted Result CSV

result
""

Figure 14: A sample of Error Analysis: Aggregation and Clinical Statistics

D SQL Generation Prompts

We include the exact prompt templates used for model prompting and refinement in both Chain-of-Thought (CoT) and Direct Output regimes.

SQL Generation Prompt (CoT)

You are a clinical data analyst expert specializing in the MIMIC-IV database. Your goal is to produce a correct BigQuery SQL query for the question below.

Constraints:
- Target platform: Google BigQuery.
- Use the correct datasets:
`physionet-data.mimiciv_3_1_hosp`, `physionet-data.mimiciv_3_1_icu`.

MIMIC-IV Schema Reference (HOSP + ICU):
{schema text}

Clinical question:
"{Question}"

Your output should be organized in the following two parts:
Reasoning:
- Think step by step about relevant tables, joins, filters, groupings, and edge cases.
- Briefly justify important choices.
SQL (wrap the final query in a fenced code block using ```` ```sql ```` and ```` ``` ````):

Think step by step and then generate the complete SQL query.

Figure 15: Chain-of-Thought SQL generation prompt used in our experiments.

Refinement Prompt (CoT)

You are a clinical data analyst expert for the MIMIC-IV dataset. The following SQL failed to run on Google BigQuery. Refine it to resolve the error and better answer the question.

Constraints:
- Use valid BigQuery SQL.
- Use the correct datasets:
`physionet-data.mimiciv_3_1_hosp`, `physionet-data.mimiciv_3_1_icu`.
- Modify only what is necessary; prefer minimal, correct fixes.

MIMIC-IV Schema Reference (HOSP + ICU):
{schema text}

Clinical question:
{Question}

Previous SQL attempt (for reference):
{Previous SQL (provided as a fenced code block)}

BigQuery error message:
{Error message}

Your output should be organized in the following two parts:
Reasoning:
- Step by step, explain the cause of the error and the fix.
- Justify key changes briefly.
SQL (wrap the final corrected query in a fenced code block using ```` ```sql ```` and ```` ``` ````):

Think step by step and then generate the complete corrected SQL query.

Figure 16: Chain-of-Thought SQL refinement prompt used in our experiments.

SQL Generation Prompt (Direct Output)

You are a clinical data analyst expert specializing in the MIMIC-IV database. Your goal is to produce a correct BigQuery SQL query for the question below.

Constraints:
- Target platform: Google BigQuery.
- Use the correct datasets:
`physionet-data.mimiciv_3_1_hosp`, `physionet-data.mimiciv_3_1_icu`.

MIMIC-IV Schema Reference (HOSP + ICU):
{schema text}

Clinical question:
"{Question}"

Output format:
- Return only a single fenced SQL code block containing the final query (use ```` ```sql ```` and ```` ``` ````).
- Do not include explanations, or any text outside the fenced SQL block.

Figure 17: Direct Output SQL generation prompt used in our experiments.

Refinement Prompt (Direct Output)

You are a clinical data analyst expert for the MIMIC-IV dataset. The following SQL failed to run on Google BigQuery. Refine it to resolve the error and better answer the question.

Constraints:
- Use valid BigQuery SQL.
- Use the correct datasets:
`physionet-data.mimiciv_3_1_hosp`, `physionet-data.mimiciv_3_1_icu`.
- Modify only what is necessary; prefer minimal, correct fixes.

MIMIC-IV Schema Reference (HOSP + ICU):
{schema text}

Clinical question:
{Question}

Previous SQL attempt (for reference):
{Previous SQL (provided as a fenced code block)}

BigQuery error message:
{Error message}

Output format:
- Return only a single fenced SQL code block containing the corrected query (use ```` ```sql ```` and ```` ``` ````).
- Do not include explanations, or any text outside the fenced SQL block.

Apply the minimal fix internally and output only the final corrected SQL.

Figure 18: Direct Output SQL refinement prompt used in our experiments.

E Configuration of Evaluated Models

Organization	Model	Release	Version	# Inference Pipeline
			Proprietary Models	
OpenAI	GPT-5-mini	2025-08	`gpt-5-mini-2025-08-07`	API
	GPT-5-nano	2025-08	`gpt-5-nano-2025-08-07`	
	GPT-5	2025-08	`gpt-5-chat-2025-08-07`	
	GPT-4.1	2025-04	`gpt-4.1-2025-04-14`	
	o4-mini	2025-04	`o4-mini-2025-04-16`	
Google	Gemini-2.5-Pro	2025-06	`gemini-2.5-pro`	API
	Gemini-2.5-Flash	2025-06	`gemini-2.5-flash`	
xAI	Grok-4-Fast-Reason.	2025-09	`grok-4-fast-reasoning`	API
	Grok-4-Fast-Non-Reason.	2025-09	`grok-4-fast-non-reasoning`	
Mistral AI	Mistral-Medium	2025-05	`mistral-medium-2505`	API
			Open-Source Models	
DeepSeek	DeepSeek-R1	2025-01	`deepseek-ai/DeepSeek-R1-0528`	vLLM
	DeepSeek-V3.1	2025-08	`deepseek-ai/DeepSeek-V3.1`	
Qwen Team	Qwen3-Coder-480B-A35B-Ins.	2025-07	`Qwen/Qwen3-Coder-480B-A35B-Instruct`	vLLM
	Qwen3-235B-A22B-Ins.	2025-07	`Qwen/Qwen3-235B-A22B-Instruct-2507`	
	Qwen3-235B-A22B-Think.	2025-07	`Qwen/Qwen3-235B-A22B-Thinking-2507-FP8`	
	Qwen3-Next-80B-A3B-Ins.	2025-09	`Qwen/Qwen3-Next-80B-A3B-Instruct`	
	Qwen3-Next-80B-A3B-Think.	2025-09	`Qwen/Qwen3-Next-80B-A3B-Thinking`	
Meta AI	Llama-4-Maverick-17B-128E-Ins.	2025-04	`meta-llama/Llama-4-Maverick-17B-128E-Instruct`	vLLM
	Llama-4-Scout-17B-16E-Ins.	2025-04	`meta-llama/Llama-4-Scout-17B-16E-Instruct`	
Defog.ai	SQLCoder-7B-2	2024-02	`defog/sqlcoder-7b-2`	vLLM
Google	MedGemma-27B	2025-06	`google/medgemma-27b-text-it`	HF
Baichuan	Baichuan-M2-32B	2025-08	`baichuan-inc/Baichuan-M2-32B`	vLLM

Table 7: Configuration of models evaluated in CLINSQL. We report official release month and canonical API/HF identifiers when available.

F Schema-Hinted Inference Setup

This section describes the schema-hinted inference setup.

Scope. We run GPT-5-mini on the CLINSQL validation set, covering all six clinical domains and the easy, medium, and hard difficulty tiers. The baseline remains the standard CoT pipeline with up to two execution-driven refinements.

Prompt augmentation. For each query, we construct a hint block from gold artifacts. We parse the reference SQL and extract ICD codes. We also read the header row of the reference result table to obtain the expected output column names. These hints are appended to the standard CoT prompt, instructing the model to include ICD filters and align SELECT aliases to the expected columns. Full schema-hinted CoT prompt templates are shown in Figure 19 and Figure 20.

SQL Generation Prompt (Schema-Hinted CoT)

You are a clinical data analyst expert specializing in the MIMIC-IV database. Your goal is to produce a correct BigQuery SQL query for the question below.

Constraints:
- Target platform: Google BigQuery.
- Use the correct datasets:
`physionet-data.mimiciv_3_1_hosp`, `physionet-data.mimiciv_3_1_icu`.

MIMIC-IV Schema Reference (HOSP + ICU):
{schema text}

Schema-hinted context:
Relevant ICD code filters observed in validated SQL examples:
- {ICD code patterns}
Incorporate the necessary ICD filters or joins when identifying the clinical cohort.
Expected column names for the final CSV output:
- {Column names}
Align your SELECT aliases with these column names and preserve ordering when applicable.

Clinical question:
"{Question}"

Your output should be organized in the following two parts:
Reasoning:
- Think step by step about relevant tables, joins, filters, groupings, and edge cases.
- Briefly justify important choices.
SQL (wrap the final query in a fenced code block using ```` ```sql ```` and ```` ``` ````):

Think step by step and then generate the complete SQL query.

Figure 19: Schema-hinted Chain-of-Thought SQL generation prompt used in our experiments.

Refinement Prompt (Schema-Hinted CoT)

You are a clinical data analyst expert for the MIMIC-IV dataset. The following SQL failed to execute. Refine it to resolve the issues and better answer the question.

Constraints:
- Use valid BigQuery SQL.
- Use the correct datasets:
`physionet-data.mimiciv_3_1_hosp`, `physionet-data.mimiciv_3_1_icu`.
- Modify only what is necessary; preserve previously correct logic.

MIMIC-IV Schema Reference (HOSP + ICU):
{schema text}

Schema-hinted context:
Relevant ICD code filters observed in validated SQL examples:
- {ICD code patterns}
Incorporate the necessary ICD filters or joins when identifying the clinical cohort.
Expected column names for the final CSV output:
- {Column names}
Align your SELECT aliases with these column names and preserve ordering when applicable.

Clinical question:
{Question}

Previous SQL attempt (for reference):
{Previous SQL (provided as a fenced code block)}

Execution feedback:
{Execution feedback}

Your output should be organized in the following two parts:
Reasoning:
- Step by step, explain the cause of the error and the fix.
- Justify key changes briefly.
SQL (wrap the final corrected query in a fenced code block using ```` ```sql ```` and ```` ``` ````):

Think step by step and then generate the complete corrected SQL query.

Figure 20: Schema-hinted Chain-of-Thought SQL refinement prompt used in our experiments.

G Validation Score Comparisons

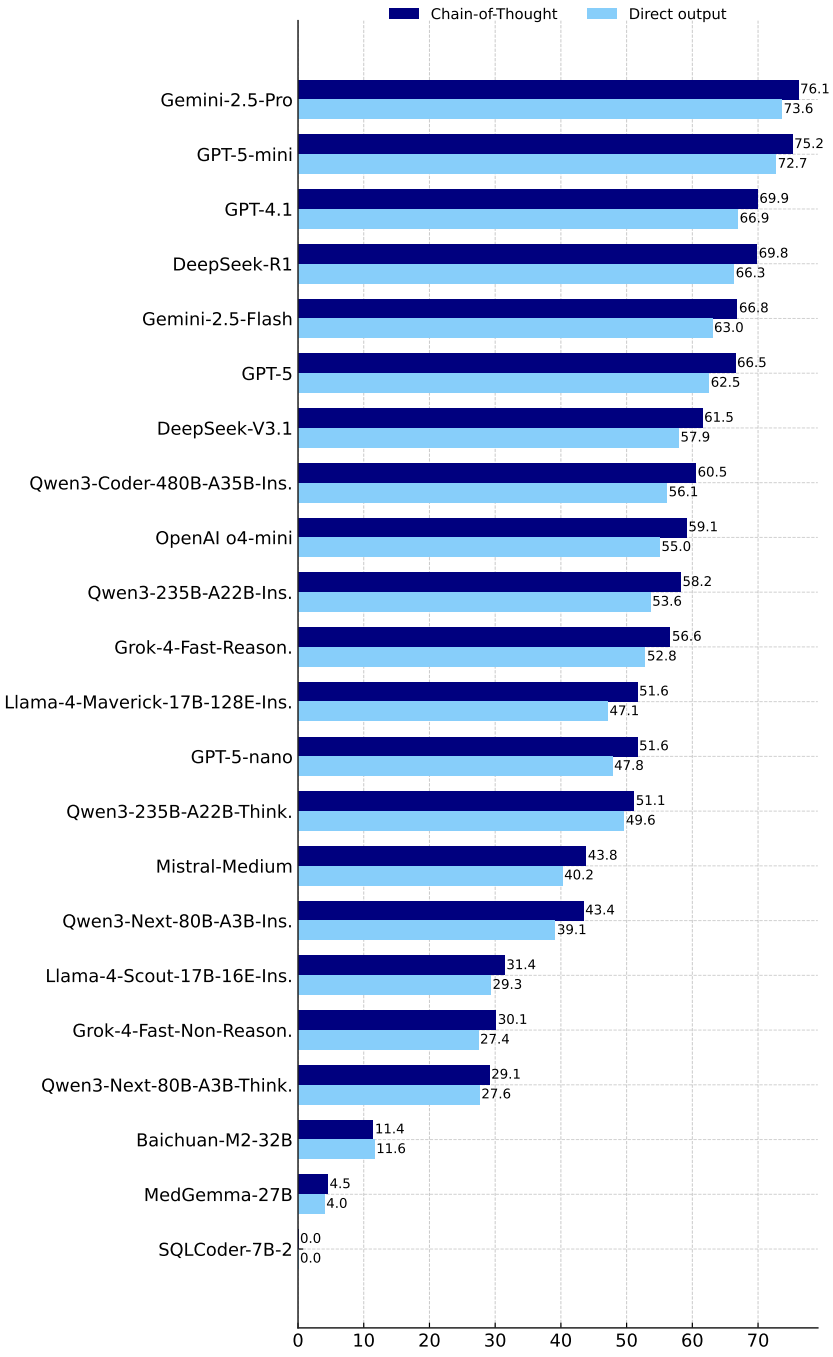

Figure 21: Full validation execution score comparison between Chain-of-Thought reasoning and Direct Output for all models.

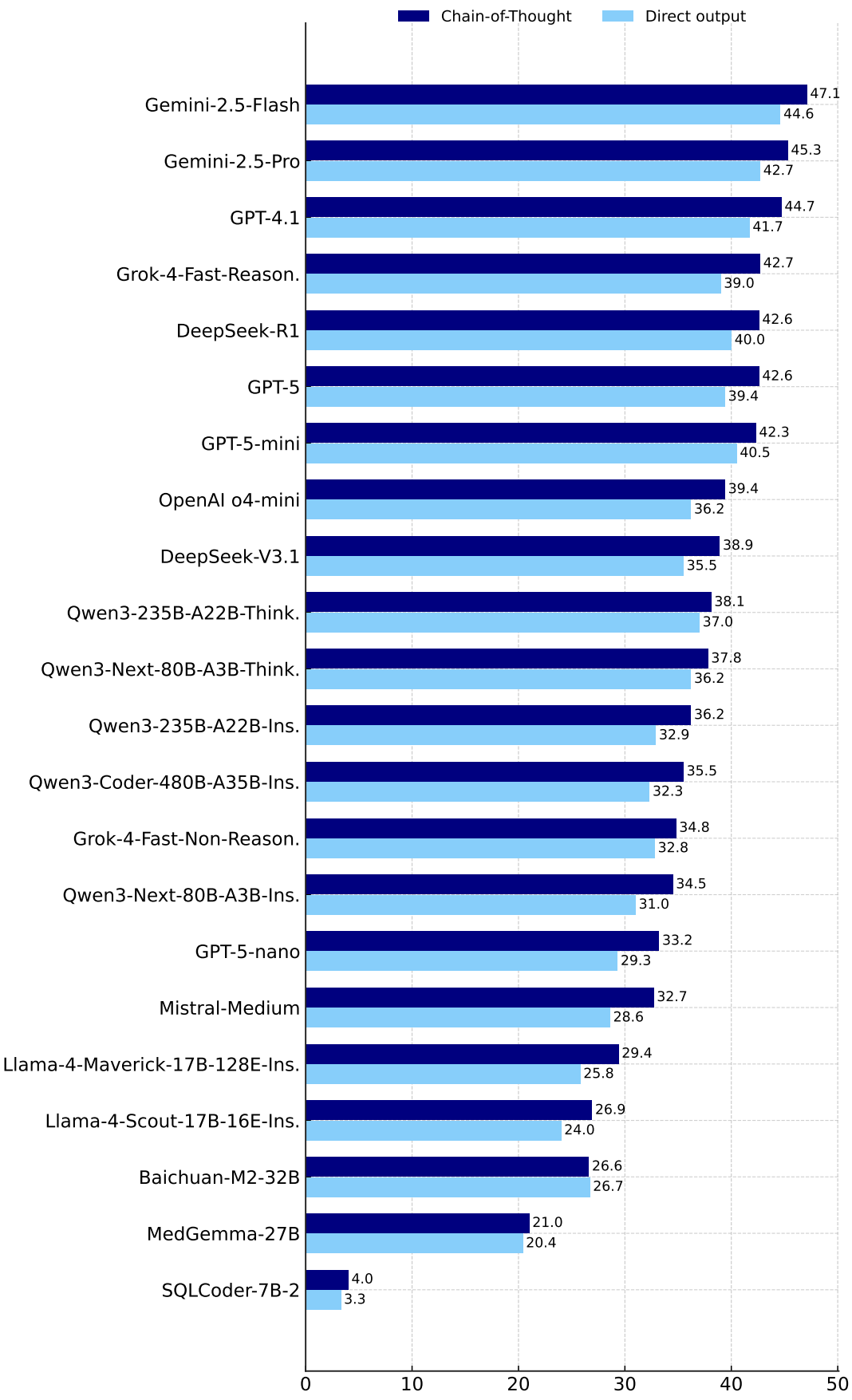

Figure 22: Full validation SQL score comparison between Chain-of-Thought reasoning and Direct Output for all models.

H Scenario-Level Results

Tables 8 and 9 extend the main results by reporting scenario-specific SQL and execution score on the CLINSQL validation and test splits.

Scenario abbreviations Demog.=Patient Demographics and Admissions; Vitals=Vital Signs Monitoring; Labs=Laboratory Results Analysis; Meds=Medication Management; Dx Proc.=Diagnostic Procedures; Dx & Outc.=Disease Diagnosis and Outcomes.

	Test Set						Avg. Validation		Avg. Test	
	Demog.		Vitals		Labs					
Model	**SQL**	**Exec**	**SQL**	**Exec**	**SQL**	**Exec**	**SQL**	**Exec**	**SQL**	**Exec**
Proprietary Models										
GPT-5-mini	50.76	80.70	35.06	72.65	40.70	65.09	42.30	75.16	42.72	74.67
GPT-5-nano	44.44	49.51	31.39	43.16	35.08	49.02	33.18	51.58	36.13	52.03
Gemini-2.5-Pro	53.41	79.22	38.06	73.97	46.98	63.66	45.31	76.14	47.28	73.73
GPT-5	54.01	76.63	35.54	50.53	39.48	57.89	42.62	66.52	45.93	68.42
GPT-4.1	56.15	77.94	34.70	58.07	43.02	48.39	44.69	69.92	46.23	67.79
Gemini-2.5-Flash	58.82	70.53	40.35	63.55	46.53	54.40	47.06	66.75	47.48	65.01
OpenAI o4-mini	52.15	67.35	35.71	46.57	34.99	48.45	39.41	59.07	41.23	59.22
Grok-4-Fast-Reason.	48.81	65.90	34.83	51.01	43.74	45.50	42.67	56.58	42.78	58.46
Grok-4-Fast-Non-Reason.	49.15	37.25	34.34	31.58	30.61	21.97	34.77	30.10	39.85	30.41
Mistral-Medium	46.07	51.97	29.54	41.22	32.96	32.22	32.71	43.81	35.33	45.10
Open-source Models										
DeepSeek-R1	49.15	74.86	36.24	62.57	53.47	66.91	42.63	69.79	44.91	69.15
DeepSeek-V3.1	54.21	65.66	34.36	54.83	40.77	53.66	38.90	61.46	43.19	60.71
Qwen3-Coder-480B-A35B-Ins.	54.33	71.97	32.53	38.48	32.92	49.78	35.54	60.51	39.05	58.18
Qwen3-235B-A22B-Ins.	45.67	69.72	31.90	41.63	38.59	55.94	36.24	58.15	36.36	58.63
Qwen3-Next-80B-A3B-Ins.	52.77	59.68	30.63	41.24	28.23	42.33	34.48	43.41	34.48	49.26
Qwen3-235B-A22B-Think.	45.25	59.19	33.11	45.09	42.74	28.94	38.08	51.11	38.20	48.54
Qwen3-Next-80B-A3B-Think.	44.56	31.36	38.95	15.68	35.33	19.75	37.77	29.06	38.13	27.01
Llama-4-Maverick-17B-128E-Ins.	43.56	55.67	25.14	42.47	22.40	37.66	29.36	51.63	29.11	47.49
Llama-4-Scout-17B-16E-Ins.	40.69	37.54	24.16	30.52	21.03	24.73	26.88	31.44	27.89	30.40
Baichuan-M2-32B	37.17	22.68	23.60	5.29	26.57	4.22	26.60	11.40	29.97	15.27
MedGemma-27B	32.74	9.27	18.07	0.22	16.51	2.31	21.03	4.46	20.92	4.00
SQLCoder-7B-2	9.85	0.00	2.96	0.00	5.31	0.00	3.99	0.00	5.29	0.00

Table 8: Scenario-level SQL and execution score (%) on CLINSQL validation and test sets. This table lists Demog., Vitals, and Labs scenarios.

	Test Set						Avg. Validation		Avg. Test	
	Meds		Dx Proc.		Dx & Outc.					
Model	**SQL**	**Exec**	**SQL**	**Exec**	**SQL**	**Exec**	**SQL**	**Exec**	**SQL**	**Exec**
Proprietary Models										
GPT-5-mini	51.23	75.06	33.72	75.64	44.90	79.55	42.30	75.16	42.72	74.67
GPT-5-nano	40.31	58.35	27.25	58.29	38.28	54.28	33.18	51.58	36.13	52.03
Gemini-2.5-Pro	52.39	77.38	41.90	72.36	50.97	76.43	45.31	76.14	47.28	73.73
GPT-5	51.99	71.07	45.15	75.38	50.05	80.22	42.62	66.52	45.93	68.42
GPT-4.1	53.12	71.03	37.81	75.44	52.74	77.52	44.69	69.92	46.23	67.79
Gemini-2.5-Flash	48.62	67.96	41.96	67.10	48.83	67.32	47.06	66.75	47.48	65.01
OpenAI o4-mini	47.45	70.01	35.00	66.04	42.53	58.19	39.41	59.07	41.23	59.22
Grok-4-Fast-Reason.	40.97	67.01	38.87	63.81	49.44	58.75	42.67	56.58	42.78	58.46
Grok-4-Fast-Non-Reason.	46.33	33.14	35.14	32.71	44.13	26.61	34.77	30.10	39.85	30.41
Mistral-Medium	37.97	58.60	26.62	45.59	38.95	41.97	32.71	43.81	35.33	45.10
Open-source Models										
DeepSeek-R1	43.39	72.87	37.52	70.16	49.09	67.97	42.63	69.79	44.91	69.15
DeepSeek-V3.1	44.19	62.13	41.75	71.46	44.32	57.61	38.90	61.46	43.19	60.71
Qwen3-Coder-480B-A35B-Ins.	39.24	66.38	34.91	63.86	41.01	59.94	35.54	60.51	39.05	58.18
Qwen3-235B-A22B-Ins.	37.89	60.35	31.68	69.55	32.46	55.79	36.24	58.15	36.36	58.63
Qwen3-Next-80B-A3B-Ins.	31.93	52.76	31.28	57.37	32.84	43.37	34.48	43.41	34.48	49.26
Qwen3-235B-A22B-Think.	42.01	62.32	29.41	47.17	36.43	50.04	38.08	51.11	38.20	48.54
Qwen3-Next-80B-A3B-Think.	40.21	45.10	31.08	31.32	38.67	19.95	37.77	29.06	38.13	27.01
Llama-4-Maverick-17B-128E-Ins.	28.86	47.08	24.13	51.65	31.15	51.33	29.36	51.63	29.11	47.49
Llama-4-Scout-17B-16E-Ins.	27.65	28.91	24.68	28.72	29.75	32.37	26.88	31.44	27.89	30.40
Baichuan-M2-32B	36.31	25.84	26.37	18.65	30.24	16.28	26.60	11.40	29.97	15.27
MedGemma-27B	16.87	2.48	22.52	9.05	19.70	1.35	21.03	4.46	20.92	4.00
SQLCoder-7B-2	5.24	0.00	4.04	0.00	4.54	0.00	3.99	0.00	5.29	0.00

Table 9: Scenario-level SQL and execution score (%) on CLINSQL validation and test sets. This table lists Meds, Dx Proc., and Dx & Outc. scenarios.

I Annotation Interface

We provide the graphical interface that annotators use while labeling CLINSQL samples, alongside the JSON file that is exported after an annotation is submitted. The pairing highlights how rubric items are surfaced during labeling and then captured in the structured log for future evaluation.

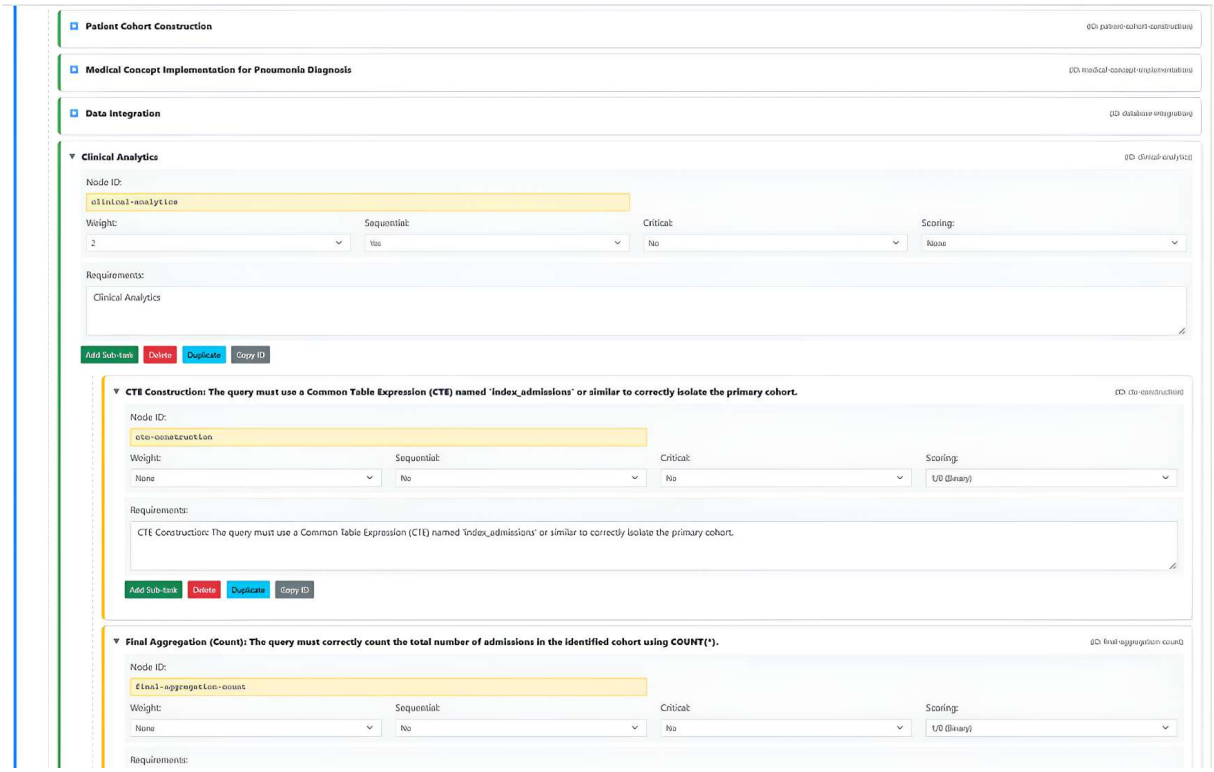

Figure 23: Graphical annotation interface used by annotators when labeling a single CLINSQL sample.

```
{
  "id": "results-validation-root",
  "requirements": "Results Validation",
  "sequential": false,
  "sub_tasks": [
    {
      "id": "output-format-validation",
      "requirements": "Output Format Validation",
      "weight": 1,
      "sequential": true,
      "sub_tasks": [
        (
          "id": "csv-file-exists",
          "requirements": "CSV File Exists",
          "sequential": false,
          "critical": false,
          "scoring": "1/0",
          "sub_tasks": []
        ),
        (
          "id": "column-name-exists",
          "requirements": "Column Name Exists",
          "sequential": true,
          "critical": false,
          "sub_tasks": [
            (
              "id": "column-for-total-admissions-exists",
              "requirements": "A column for the total number of index admissions exists (e.g., 'total_cohort_admissions', 'num_admissions', 'total_patients').",
              "sequential": false,
              "critical": false,
              "scoring": "1/0",
              "sub_tasks": []
            )
          ]
        ),
        (
          "id": "no-null-empty-values-in-admissions-column",
          "requirements": "The value in the column for the total number of admissions is not NULL or empty.",
          "sequential": false,
          "critical": false,
          "scoring": "1/0",
          "sub_tasks": []
        )
      ]
    },
    {
      "id": "output-feature-validation",
      "requirements": "Clinical Value Validation",
      "weight": 2,
      "sequential": true,
      "sub_tasks": [
        (
          "id": "total-admissions-validation",
          "requirements": "Validation of the total number of admissions",
```


Figure 24: JSON export generated from the completed annotation, preserving rubric selections and metadata.

J Execution Success Rates

We benchmark execution reliability across all CLINSQL scenarios using the test split. Table 10 reports the queries that executed without errors for every model and scenario, while Table 11 and Table 12 break down execution success by the initial query (A1) and up to two refinement attempts (A2 and A3).

Model	Scenarios (%)					
	Demog.	Vitals	Labs	Meds	Dx Proc.	Dx & Outc.
GPT-5-mini	100.0	98.7	92.3	93.2	97.1	90.5
GPT-5-nano	91.5	89.5	97.4	85.1	94.3	75.7
Gemini-2.5-Pro	97.2	97.4	91.0	97.3	95.7	90.5
GPT-5	100.0	96.1	94.9	100.0	97.1	100.0
GPT-4.1	98.6	94.7	96.2	93.2	98.6	93.2
Gemini-2.5-Flash	93.0	85.5	82.1	93.2	87.1	81.1
OpenAI o4-mini	100.0	94.7	93.6	97.3	97.1	86.5
Grok-4-Fast-Reason.	97.2	93.4	76.9	90.5	84.3	73.0
Grok-4-Fast-Non-Reason.	59.2	55.3	21.8	33.8	34.3	18.9
Mistral-Medium	83.1	89.5	70.5	90.5	82.9	74.3
DeepSeek-R1	98.6	94.7	98.7	95.9	94.3	89.2
DeepSeek-V3.1	95.8	93.4	88.5	87.8	98.6	82.4
Qwen3-Coder-480B-A35B-Ins.	98.6	92.1	93.6	95.9	100.0	93.2
Qwen3-235B-A22B-Ins.	98.6	85.5	93.6	86.5	91.4	64.9
Qwen3-Next-80B-A3B-Ins.	80.3	71.1	65.4	74.3	70.0	47.3
Qwen3-235B-A22B-Think.	100.0	92.1	94.9	95.9	85.7	79.7
Qwen3-Next-80B-A3B-Think.	56.3	57.9	50.0	77.0	70.0	47.3
Llama-4-Maverick-17B-128E-Ins.	88.7	77.6	75.6	77.0	80.0	79.7
Llama-4-Scout-17B-16E-Ins.	49.3	55.3	59.0	47.3	51.4	44.6
Baichuan-M2-32B	62.0	52.6	48.7	55.4	51.4	47.3
MedGemma-27B	32.4	17.1	29.5	20.3	28.6	12.2
SQLCoder-7B-2	0.0	1.3	1.3	0.0	1.4	2.7

Table 10: Execution success rate (%) per model and scenario on the test split. Abbreviations defined in section H.

Model	Scenarios (%)								
	Demog.			Vitals			Labs		
	A1	A2	A3	A1	A2	A3	A1	A2	A3
GPT-5-mini	84.5	14.1	1.4	63.2	27.6	7.9	41.0	41.0	10.3
GPT-5-nano	60.6	23.9	7.0	56.6	22.4	10.5	53.8	32.1	11.5
Gemini-2.5-Pro	88.7	5.6	2.8	81.6	13.2	2.6	71.8	16.7	2.6
GPT-5	85.9	8.5	5.6	73.7	15.8	6.6	73.1	12.8	9.0
GPT-4.1	78.9	18.3	1.4	72.4	19.7	2.6	74.4	15.4	6.4
Gemini-2.5-Flash	62.0	19.7	11.3	47.4	26.3	11.8	33.3	30.8	17.9
OpenAI o4-mini	73.2	25.4	1.4	57.9	26.3	10.5	59.0	26.9	7.7
Grok-4-Fast-Reason.	43.7	36.6	16.9	38.2	38.2	17.1	9.0	33.3	34.6
Grok-4-Fast-Non-Reason.	19.7	23.9	15.5	30.3	11.8	13.2	3.8	11.5	6.4
Mistral-Medium	62.0	12.7	8.5	51.3	25.0	13.2	43.6	17.9	9.0
DeepSeek-R1	78.9	11.3	8.5	50.0	36.8	7.9	69.2	24.4	5.1
DeepSeek-V3.1	67.6	22.5	5.6	71.1	14.5	7.9	70.5	12.8	5.1
Qwen3-Coder-480B-A35B-Ins.	77.5	16.9	4.2	69.7	19.7	2.6	71.8	16.7	5.1
Qwen3-235B-A22B-Ins.	69.0	23.9	5.6	47.4	31.6	6.6	52.6	29.5	11.5
Qwen3-Next-80B-A3B-Ins.	32.4	26.8	21.1	28.9	30.3	11.8	15.4	29.5	20.5
Qwen3-235B-A22B-Think.	42.3	52.1	5.6	47.4	32.9	11.8	42.3	47.4	5.1
Qwen3-Next-80B-A3B-Think.	42.3	11.3	2.8	35.5	14.5	7.9	30.8	16.7	2.6
Llama-4-Maverick-17B-128E-Ins.	57.7	28.2	2.8	47.4	18.4	11.8	48.7	23.1	3.8
Llama-4-Scout-17B-16E-Ins.	21.1	19.7	8.5	22.4	22.4	10.5	16.7	24.4	17.9
Baichuan-M2-32B	25.4	21.1	15.5	14.5	27.6	10.5	19.2	19.2	10.3
MedGemma-27B	15.5	5.6	11.3	10.5	6.6	0.0	19.2	9.0	1.3
SQLCoder-7B-2	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0	1.3

Table 11: Per-attempt execution success rate (%). This table lists Demog., Vitals, Labs. Attempts A1, A2 and A3 correspond to the initial query and up to two refinements.

Model	Scenarios (%)								
	Meds			Dx Proc.			Dx & Outc.		
	A1	A2	A3	A1	A2	A3	A1	A2	A3
GPT-5-mini	41.9	37.8	13.5	51.4	37.1	8.6	44.6	32.4	13.5
GPT-5-nano	35.1	40.5	9.5	47.1	40.0	7.1	29.7	32.4	13.5
Gemini-2.5-Pro	79.7	14.9	2.7	75.7	18.6	1.4	74.3	9.5	6.8
GPT-5	71.6	25.7	2.7	84.3	10.0	2.9	78.4	13.5	8.1
GPT-4.1	73.0	20.3	0.0	75.7	21.4	1.4	78.4	13.5	1.4
Gemini-2.5-Flash	51.4	31.1	10.8	45.7	30.0	11.4	35.1	29.7	16.2
OpenAI o4-mini	59.5	33.8	4.1	57.1	37.1	2.9	56.8	25.7	4.1
Grok-4-Fast-Reason.	24.3	54.1	12.2	32.9	37.1	14.3	16.2	28.4	28.4
Grok-4-Fast-Non-Reason.	14.9	9.5	9.5	20.0	4.3	10.0	4.1	5.4	9.5
Mistral-Medium	48.6	31.1	10.8	48.6	28.6	5.7	47.3	17.6	9.5
DeepSeek-R1	59.5	31.1	5.4	70.0	18.6	5.7	51.4	27.0	10.8
DeepSeek-V3.1	56.8	27.0	4.1	75.7	17.1	5.7	55.4	8.1	18.9
Qwen3-Coder-480B-A35B-Ins.	67.6	17.6	10.8	61.4	32.9	5.7	60.8	23.0	9.5
Qwen3-235B-A22B-Ins.	35.1	33.8	17.6	60.0	28.6	2.9	32.4	24.3	8.1
Qwen3-Next-80B-A3B-Ins.	27.0	21.6	25.7	21.4	30.0	18.6	21.6	13.5	12.2
Qwen3-235B-A22B-Think.	27.0	59.5	9.5	20.0	52.9	12.9	28.4	43.2	8.1
Qwen3-Next-80B-A3B-Think.	31.1	28.4	17.6	41.4	21.4	7.1	35.1	9.5	2.7
Llama-4-Maverick-17B-128E-Ins.	45.9	20.3	10.8	61.4	17.1	1.4	51.4	20.3	8.1
Llama-4-Scout-17B-16E-Ins.	8.1	18.9	20.3	11.4	18.6	21.4	8.1	23.0	13.5
Baichuan-M2-32B	21.6	25.7	8.1	21.4	24.3	5.7	18.9	16.2	12.2
MedGemma-27B	12.2	4.1	4.1	21.4	5.7	1.4	2.7	2.7	6.8
SQLCoder-7B-2	0.0	0.0	0.0	0.0	0.0	1.4	0.0	0.0	2.7

Table 12: Per-attempt execution success rate (%). This table lists Meds, Dx Proc., Dx & Outc.

K Human–GPT Agreement Study

Because our rubric-based evaluation relies on GPT-5 as the judge, we run a human–GPT agreement study to validate the reliability of its rubric decisions. We randomly sample 100 CLINSQL validation/test examples and re-score them with two medically trained annotators from the same pool that constructs the dataset and rubrics. For each example, both annotators independently evaluate rubric leaf nodes for the SQL and results trees using the same annotation guidelines, without access to GPT-5 scores or each other's labels.

We compare GPT-5's leaf-level decisions and aggregated pass/fail outcomes against each annotator, and also compare the two annotators with each other. Table 13 reports agreement rates at the leaf level and at the final pass/fail level for both SQL and results. Disagreements are largely concentrated in borderline, partial-credit cases, and do not materially change relative model rankings, supporting the use of GPT-5 as a reliable and scalable rubric judge for our tree-structured evaluation.

Pair	**Leaf-level agreement (SQL, %)**	**Leaf-level agreement (results, %)**	**Pass/fail agreement (SQL, %)**	**Pass/fail agreement (results, %)**
GPT-5 vs. Annotator 1	83.4	87.1	90.2	92.3
GPT-5 vs. Annotator 2	82.1	85.9	88.7	91.0
Annotator 1 vs. Annotator 2	86.8	89.5	92.4	94.1

Table 13: Human–GPT agreement on 100 randomly sampled CLINSQL examples. Leaf-level agreement compares rubric leaf decisions, while pass/fail agreement compares aggregated outcomes for SQL and results.

L Inter-Annotator Agreement and Reconciliation Protocols

The trustworthiness of an evaluation benchmark is essential for expert-domain assessment (Ke et al., 2025a,b; Ouyang et al., 2024). To this end, CLINSQL uses validator-based quality control to stabilize scenario design, gold SQL, and rubric leaves. Each scenario is created by a primary annotator and independently reviewed by a validator. Validators re-run the SQL in BigQuery, inspect result tables, and check rubric trees for coverage and correctness. For each item they record one of three outcomes: accept as-is, accept with minor edits, or major revision or reject. Across the benchmark, validators accept 87% of items as-is, request 9% minor edits (e.g., tightening a time window, adjusting an inclusion criterion, or clarifying a rubric leaf), and request 4% major revision or rejection.

To quantify inter-annotator agreement (IAA), we additionally sample 50 validation/test scenarios and ask a second annotator, distinct from both the original annotator and the validator, to re-annotate them independently at two levels. For gold SQL, the second annotator writes a fresh query based only on the natural-language description and schema. We canonicalize both SQL queries and execute them on the redacted BigQuery database. In 46/50 cases (92%), the two queries produce identical result tables (up to row/column ordering). In 3/50 cases (6%), the results differ only by small numerical variations and are treated as near-miss agreements. In the remaining 1/50 case (2%), the second annotator interprets the clinical question differently, yielding a clinically distinct cohort. Exact SQL-level agreement is 92%; counting near-miss cases as acceptable alternatives yields 98%.

For rubric leaves, an independent annotator reconstructs the SQL and results rubrics using shared templates and guidelines. We treat each rubric as a set of atomic checks (leaf presence and criticality) and compare the two annotators' trees. Raw agreement on leaf presence and criticality is 91% with Cohen's $\kappa = 0.82$, and the critical-first aggregation yields a pass/fail agreement of 47/50 scenarios (94%). Table 14 summarizes these reconciliation and IAA statistics.

Statistic	Value
Validator-based quality control (all CLINSQL items)	
Accept as-is	87%
Accept with minor edits	9%
Major revision / reject	4%
Double-annotation study (n=50 validation/test scenarios)	
SQL exact match (identical results)	46/50 (92%)
SQL near-miss (minor numeric differences)	3/50 (6%)
SQL distinct (clinically different)	1/50 (2%)
Rubric leaf agreement (presence + criticality)	91% (Cohen's $\kappa = 0.82$)
Rubric pass/fail agreement	47/50 (94%)

Table 14: Inter-annotator agreement (IAA) and reconciliation statistics for CLINSQL. Validator outcomes are reported on the full benchmark; double-annotation statistics are computed on a 50-scenario validation/test sample.

M Reconciling Execution Passes with SQL Analysis Failures

SQL analysis and execution scoring capture complementary notions of correctness: SQL analysis checks whether the query encodes the intended cohort logic, whereas execution scoring evaluates whether the observed results are clinically plausible and consistent with the gold answer. At the model level, these signals are strongly aligned. Across all models in the main tables, the Pearson correlation between SQL Score and Execution Score is $r = 0.8597$, and the Spearman rank correlation is $\rho = 0.8554$, indicating near-identical model rankings.

To diagnose divergence cases, we run a targeted outlier study. We identify model–dataset points with a large absolute gap between SQL Score and Execution Score (at least 20 percentage points) and randomly sample 40 such outliers for manual inspection of both SQL and result tables. Most outliers (about 95%) exhibit high execution scores but low SQL scores, reflecting partial or imprecise cohort logic (e.g., missing secondary exclusion criteria, incomplete temporal logic, or incorrect aggregation granularity). In these cases, execution remains high because the resulting aggregates stay close to reference values despite logical deviations. The remaining outliers (about 5%) show high SQL scores but lower execution scores, typically when correct logic yields clinically implausible values in narrow subgroups. These findings indicate that discrepancies arise from meaningful error types rather than rubric misalignment.

Table 15 summarizes the correlation statistics and outlier breakdown.

Statistic	**Value**
Pearson correlation (SQL Score vs. Execution Score)	$r = 0.8597$
Spearman rank correlation (SQL Score vs. Execution Score)	$\rho = 0.8554$
Outlier threshold ($\lvert$SQL – Execution$\rvert \geq 20$ points)	40 cases sampled
High Execution / low SQL among outliers	$\approx 95\%$
High SQL / low Execution among outliers	$\approx 5\%$

Table 15: Reconciliation of SQL analysis and execution scoring. Correlations are computed across all models in the main tables; outlier statistics are based on 40 sampled large-gap cases.

N Annotation Guideline

N.1 Part I: Annotation Guidelines

N.1.1 Overview

Each example in our benchmark consists of:

- The type of realistic clinical scenario, representing one of the six scenario types defined in the paper
- A natural language clinical question that requires a database query to be solved
- A gold standard SQL query that accurately translates the clinical question into executable database operations
- A results table in CSV format, generated from the execution of the gold-standard SQL query
- An evaluation guideline for evaluating the accuracy of the SQL queries and executed results generated by other models

You will be assigned the following three sequential roles:

1. **Query Annotator:** Develop the clinical scenario, provide the patient context, and formulate the natural language question.
2. **SQL Annotator:** Analyze database requirements, construct the gold-standard SQL implementation, execute the SQL queries, analyze the results table, and annotate the specific database tables that were used and key features in the results table (columns, values).
3. **Evaluation Guideline Annotator:** For each example, create a tailored guideline for evaluating the SQL query and validating its results.

While the annotation interface will guide you, it is essential to follow the instructions carefully.

N.2 Step 1: Query Annotation

As the **Query Annotator**, your task is to create realistic clinical scenarios and formulate natural language questions that require database queries to solve. You will develop questions that reflect authentic clinical decision-making processes and information needs.

N.2.1 Clinical Scenario Development

1. **Select and Understand Your Assigned Patient Information and Scenario Type**
    - You will be assigned one of six clinical scenario types and a specific patient from the MIMIC-IV database, selected based on that scenario type for filtering: Patient Demographics & Admissions, Vital Signs Monitoring, Laboratory Results Analysis, Medication Management, Diagnostic Procedures, or Disease Diagnosis & Outcomes.
    - Before you begin, familiarize yourself with the MIMIC tables and understand the information for the specific patient assigned, and review the scenario definition.

N.2.2 Natural Language Question Formulation

1. **Craft the Clinical Question**
    - Write a natural language question that a **physician** would realistically ask, given the information of the provided patient.
    - Ensure the question requires database querying and cannot be answered through simple observation.
    - Use appropriate medical terminology while maintaining clarity.

N.3 Step 2: SQL Annotation

Your task is to analyze the clinical question, identify database requirements, construct the gold-standard SQL query, and document the implementation details.

N.3.1 Database Analysis and Schema Mapping

1. **Clinical Question Analysis**
    - Identify the specific clinical data elements required to answer the question.
2. **MIMIC-IV Database Mapping**
    - Identify all MIMIC-IV tables required to answer the question.
    - Map clinical concepts in the question to specific database tables and columns.
    - Determine relevant clinical thresholds, normal ranges, and medical domain knowledge.
    - Map clinical conditions, procedures, and interventions mentioned in the question to their corresponding ICD codes.

N.3.2 Gold-Standard SQL Annotation

1. **SQL Query Development**
    - Write a complete, executable SQL query that accurately answers the clinical question.
    - Ensure the query handles edge cases and data quality issues common in clinical databases.
2. **Query Execution and Result Generation**
    - Execute the SQL query against the MIMIC-IV database.
    - Generate the complete results table in CSV format.
    - Verify that results are clinically meaningful and interpretable.

N.4 Evaluation Guideline Annotation

As the **Evaluation Guideline Annotator**, your responsibility is to create comprehensive criteria for evaluating SQL queries and executed results generated by other models attempting to answer the clinical question.

N.4.1 Understanding Evaluation Rubric Structure

Before building evaluation rubrics, you must understand the foundational concepts that govern how evaluation scores are calculated in our benchmark system.

Critical vs. Non-Critical Nodes Our evaluation system employs two types of assessment nodes:

- **Critical Nodes** [Critical]: Essential criteria whose failure immediately causes the parent node to fail, regardless of other sibling node performance. Critical nodes represent fundamental requirements that must be satisfied for meaningful evaluation.
- **Non-Critical Nodes**: Allow partial scoring at the parent level. When mixed with critical nodes, non-critical nodes contribute to averaging only after all critical nodes pass.
- **Score = 1:** The requirement is fully satisfied. The SQL query or result demonstrates correct and clinically appropriate implementation of this component.
- **Score = 0:** The requirement is not satisfied. The component is missing, incorrectly implemented, or produces clinically invalid output.

For example:

- If Gender Selection [1] [Critical] and Age Range Selection [1] [Critical] $\rightarrow$ Patient Cohort Construction [1]

- If Gender Selection [1] [Critical] and Age Range Selection [0] [Critical] → Patient Cohort Construction [0]
- If Gender Selection [1] [Critical], Age Range Selection [1] [Critical], and Time Filter [0] (non-critical) → Patient Cohort Construction = (0)/1 = 0 (average of non-critical nodes)

Sequential Dependencies [sequential] Some evaluation nodes are marked as **sequential**, indicating logical dependencies among child nodes where failure at an earlier step renders subsequent evaluations meaningless. For example, if a SQL query fails to correctly filter the patient cohort, evaluating the aggregation logic becomes pointless.

For example:

- If Table Join Logic [1] [sequential] and Key Matching [1] [sequential] → Data Integration = (1+1)/2 = 1 (all sequential steps succeed)
- If Table Join Logic [0] [sequential] and Key Matching [not evaluated] [sequential] → Data Integration = (0)/1 = 0 (sequential failure stops evaluation)
- If Table Join Logic [1] [sequential], Key Matching [1] [sequential], and Final Validation [0] [sequential] → Data Integration = (1+1+0)/3 = 0.67 (sequential failure after partial evaluation)

Weight Assignment [Weight X] Each major evaluation category is assigned a weight reflecting its relative importance in the overall assessment. Weights enable proportional scoring where more critical aspects (e.g., patient cohort construction) receive higher influence than secondary considerations.

N.4.2 Quantified Weight Scale

Our evaluation framework employs a 3-point weight scale based on clinical importance:

Weight 1: Basic Supportive Criteria

- Represents supplementary evaluation components that provide additional context.
- Examples: Output formatting, minor data type handling, non-essential temporal constraints.

```
-- Output formatting and rounding
SELECT ROUND(AVG(procedure_count), 2) as avg_imaging_procedures

-- Column aliasing for readability
COUNT(DISTINCT pr.icd_code) as procedure_count
```

Typically assigned to elements that enhance quality but are not fundamental to clinical correctness.

Weight 2: Standard Clinical Requirements

- Represents standard clinical database operations and moderate complexity reasoning.
- Examples: Medical concept implementation, aggregation functions, procedure identification.

```
-- Medical concept implementation - ICD code pattern matching
(pr.icd_version = 10 AND (
  pr.icd_code LIKE 'B%' OR     -- Imaging procedures
  pr.icd_code LIKE '3E0%' OR   -- CT procedures
  pr.icd_code LIKE 'BW%' OR    -- X-ray procedures
  pr.icd_code LIKE 'B3%'       -- Ultrasound procedures
))

-- Aggregation functions for clinical analytics
COUNT(DISTINCT pr.icd_code) as procedure_count
```

```
AVG(procedure_count)

-- ICD version handling
(pr.icd_version = 9 AND (
  pr.icd_code LIKE '87%' OR    -- Diagnostic radiology
  pr.icd_code LIKE '88%'       -- Other diagnostic procedures
))
```

Assigned to components that demonstrate competent clinical data analysis capabilities.

Weight 3: Critical Clinical Elements

- Represents essential requirements whose failure undermines clinical validity and elements requiring substantial clinical domain knowledge and SQL proficiency.
- Examples: Core patient demographic filtering, critical medical code selection, fundamental table relationships, patient cohort construction, database integration with complex joins, clinical analytics.

```
-- Critical patient cohort construction
WHERE p.gender = 'M'
  AND p.anchor_age BETWEEN 60 AND 70

-- Fundamental table relationships
FROM `physionet-data.mimiciv_3_1_hosp.patients` p
JOIN `physionet-data.mimiciv_3_1_hosp.procedures_icd` pr
  ON p.subject_id = pr.subject_id

-- Essential grouping for per-patient analysis
GROUP BY p.subject_id

-- Critical medical filtering logic
WHERE p.gender = 'M'
  AND p.anchor_age BETWEEN 60 AND 70
  AND (
    -- Comprehensive ICD version and code handling
    (pr.icd_version = 10 AND (...)) OR
    (pr.icd_version = 9 AND (...))
  )
```

Reserved for components that are absolutely essential for producing clinically meaningful results and require deep understanding of both clinical domain and advanced SQL capabilities.

Scoring aggregation follows the critical-first protocol described in Algorithm 1.

N.4.3 Build SQL Query Evaluation Rubric

Create a hierarchical evaluation tree tailored to your specific clinical question and SQL implementation. The structure should reflect the logical flow of SQL query construction while identifying critical checkpoints. See Figure 3 for an example sql rubric tree.

N.4.4 Build Results Validation Rubric

Create validation criteria based on the actual generated CSV file from your gold-standard SQL execution, combined with your clinical knowledge. See Figure 4 for an example results rubric tree.

N.5 Part II: Validation Guidelines

As a **Validator**, your role is to ensure every clinical example meets our benchmark standards. To do this, you will perform a comprehensive review of all its components: the natural language question, the SQL query, the executed results, and the evaluation guideline.

N.5.1 Clinical Question Assessment

- Real-world Relevance: Question represents authentic clinical decision-making scenarios
- Medical Language: Accurate clinical terminology and healthcare concepts
- Scenario Match: Aligns with designated clinical category
- Linguistic Quality: Clear, grammatically sound, and unambiguous phrasing
- Query Requirement: Necessitates database analysis, not simple observation

N.5.2 SQL Implementation Review

- Database Standards: Uses correct MIMIC-IV paths (`physionet-data.mimiciv_3_1_hosp`)
- Schema Validation: Accurate table references, columns, and join relationships
- Medical Logic: Valid age computation, ICD handling, and temporal analysis
- Technical Function: Error-free execution with proper NULL management
- Query Coverage: Comprehensively addresses clinical question requirements

N.5.3 Output Verification

- Structure: Well-formed CSV with meaningful column labels
- Medical Plausibility: Values fall within clinically acceptable boundaries
- Data Integrity: Complete dataset without missing essential information
- Logic Alignment: Output corresponds to SQL query operations

N.5.4 Evaluation Framework Review

- Component Coverage: SQL evaluation addresses all query elements
- Priority Identification: [Critical] labels properly applied to essential parts
- Order Dependencies: [Sequential] tags used where sequence matters
- Output Standards: Adequate value ranges and format specifications
- Assessment Clarity: Unambiguous binary scoring system

N.5.5 Complexity Level Classification

- Difficulty Assessment: Evaluate and categorize the annotated query–SQL pair according to the appropriate complexity level based on SQL complexity and clinical reasoning requirements.
- Classification Accuracy: Ensure each example is correctly assigned to Easy, Medium, or Hard difficulty levels to maintain uniform standards throughout the benchmark.

N.5.6 Action Required

- If the example fails **any** of the above checks, revise it if corrections are minor (e.g., grammar fixes, small SQL adjustments, or evaluation refinements).
- If issues are significant (e.g., clinically inappropriate question, fundamentally incorrect SQL, or incomplete evaluation framework), you may **reject** the example or heavily revise.
- Provide brief justification when making revisions or rejections.

N.5.7 Mark as Validated

Once all checks have passed, mark the example as **Validated**. This confirms it is ready for inclusion in the final dataset.

O Judge Prompt

We use an LLM-as-a-judge to score rubric leaf nodes with binary decisions and short explanations. The SQL- and results-level prompt templates are shown in Figure 25 and Figure 26.

Judge Prompt: SQL Evaluation

You are evaluating SQL queries for clinical data analysis based on specific requirements.

Evaluation Criteria:
{node.requirements}

Clinical Question:
{query}

SQL to Evaluate (fenced):
{test_sql}

Gold Standard SQL:
{gold_sql}

Instructions:
1. Evaluate if the SQL meets the specific requirement: "{node.requirements}".
2. Focus on whether the implementation satisfies the requirement, not on syntactic perfection.
3. Use the gold standard SQL as reference for best practices and expected approach.
4. Score: 1 if requirement is fully met, 0 if not met.
5. Provide a brief explanation of your assessment.

Response Format:
Score: [0 or 1]
Explanation: [Brief explanation of why the score was given]

Figure 25: LLM judge prompt template for SQL-level rubric evaluation.

Judge Prompt: Results Evaluation

You are evaluating clinical query results based on specific requirements.

Evaluation Criteria:
{node.requirements}

Clinical Question:
{query}

Results to Evaluate:
{test_results}

Gold Standard Results:
{gold_results}

Instructions:
1. Evaluate if the results meet the specific requirement: "{node.requirements}".
2. For "CSV File Exists" requirements: if results data is shown above and not empty, it means a CSV file exists.
3. Use the gold standard results as reference for expected format and values.
4. Consider clinical plausibility, data format, and completeness.
5. Score: 1 if requirement is fully met, 0 if not met.
6. Provide a brief explanation of your assessment.

Response Format:
Score: [0 or 1]
Explanation: [Brief explanation of why the score was given]

Figure 26: LLM judge prompt template for results-level rubric evaluation.

P MIMIC-IV Schema

MIMIC-IV — HOSP module

admissions
Columns: subject_id, hadm_id, admittime, dischtime, deathtime, admission_type, admit_provider_id, admission_location, discharge_location, insurance, language, marital_status, race, edregtime, edouttime, hospital_expire_flag

patients
Columns: subject_id, gender, anchor_age, anchor_year, anchor_year_group, dod

transfers
Columns: subject_id, hadm_id, transfer_id, eventtype, careunit, intime, outtime

labevents
Columns: labevent_id, subject_id, hadm_id, specimen_id, itemid, charttime, storetime, value, valuenum, valueuom, ref_range_lower, ref_range_upper, flag, priority, comments

d_labitems
Columns: itemid, label, fluid, category, loinc_code

microbiologyevents
Columns: microevent_id, subject_id, hadm_id, micro_specimen_id, order_provider_id, chartdate, charttime, spec_itemid, spec_type_desc, test_seq, storedate, storetime, test_itemid, test_name, org_itemid, org_name, isolate_num, quantity, ab_itemid, ab_name, dilution_text, dilution_comparison, dilution_value, interpretation, comments

diagnoses_icd
Columns: subject_id, hadm_id, seq_num, icd_code, icd_version

d_icd_diagnoses
Columns: icd_code, icd_version, long_title

procedures_icd
Columns: subject_id, hadm_id, seq_num, chartdate, icd_code, icd_version

d_icd_procedures
Columns: icd_code, icd_version, long_title

emar
Columns: subject_id, hadm_id, emar_id, emar_seq, poe_id, pharmacy_id, enter_provider_id, charttime, medication, event_txt, scheduletime, storetime

emar_detail
Columns: subject_id, emar_id, emar_seq, parent_field_ordinal, administration_type, pharmacy_id, barcode_type, reason_for_no_barcode, complete_dose_not_given, dose_due, dose_due_unit, dose_given, dose_given_unit, will_remainder_of_dose_be_given, product_amount_given, product_unit, product_code, product_description, prior_infusion_rate, infusion_rate, infusion_rate_adjustment, infusion_rate_adjustment_amount, infusion_rate_unit, route, infusion_complete, completion_interval, new_iv_bag_hung, continued_infusion_in_other_location, restart_interval, side, site, non_formulary_visual_verification

prescriptions
Columns: subject_id, hadm_id, pharmacy_id, poe_id, poe_seq, order_provider_id, starttime, stoptime, drug_type, drug, formulary_drug_cd, gsn, ndc, prod_strength, form_rx, dose_val_rx, dose_unit_rx, form_val_disp, form_unit_disp, doses_per_24_hrs, route

pharmacy
Columns: subject_id, hadm_id, pharmacy_id, poe_id, starttime, stoptime, medication, proc_type, status, entertime, verifiedtime, route, frequency, disp_sched, infusion_type, sliding_scale, lockout_interval, basal_rate, one_hr_max, doses_per_24_hrs, duration, duration_interval, expiration_value, expiration_unit, expirationdate, dispensation, fill_quantity

poe
Columns: poe_id, poe_seq, subject_id, hadm_id, ordertime, order_type, order_subtype, transaction_type, discontinue_of_poe_id, discontinued_by_poe_id, order_provider_id, order_status

poe_detail
Columns: poe_id, poe_seq, subject_id, field_name, field_value

hcpcsevents

Columns: subject_id, hadm_id, chartdate, hcpcs_cd, seq_num, short_description

d_hcpcs
Columns: code, category, long_description, short_description

drgcodes
Columns: subject_id, hadm_id, drg_type, drg_code, description, drg_severity, drg_mortality

services
Columns: subject_id, hadm_id, transfertime, prev_service, curr_service

provider
Columns: provider_id

omr
Columns: subject_id, chartdate, seq_num, result_name, result_value

MIMIC-IV — ICU module

icustays
Columns: subject_id, hadm_id, stay_id, first_careunit, last_careunit, intime, outtime, los

chartevents
Columns: subject_id, hadm_id, stay_id, caregiver_id, charttime, storetime, itemid, value, valuenum, valueuom, warning

datetimesevents
Columns: subject_id, hadm_id, stay_id, caregiver_id, charttime, storetime, itemid, value, valueuom, warning

inputevents
Columns: subject_id, hadm_id, stay_id, caregiver_id, starttime, endtime, storetime, itemid, amount, amountuom, rate, rateuom, orderid, linkorderid, ordercategoryname, secondaryordercategoryname, ordercomponenttypedescription, ordercategorydescription, patientweight, totalamount, totalamountuom, isopenbag, statusdescription, originalamount, originalrate

ingredientevents
Columns: subject_id, hadm_id, stay_id, caregiver_id, starttime, endtime, storetime, itemid, amount, amountuom, rate, rateuom, orderid, linkorderid, statusdescription, originalamount, originalrate

outputevents
Columns: subject_id, hadm_id, stay_id, caregiver_id, charttime, storetime, itemid, value, valueuom

procedureevents
Columns: subject_id, hadm_id, stay_id, caregiver_id, starttime, endtime, storetime, itemid, value, valueuom, location, locationcategory, orderid, linkorderid, ordercategoryname, ordercategorydescription, patientweight, isopenbag, continueinnextdept, statusdescription, originalamount, originalrate

d_items
Columns: itemid, label, abbreviation, linksto, category, unitname, param_type, lownormalvalue, highnormalvalue

caregiver
Columns: caregiver_id

Concise notes (commonly confusing columns)

- hadm_id vs. stay_id: hadm_id is the hospital admission identifier; stay_id tracks an ICU stay within an admission.
- charttime vs. storetime: charttime captures when the event occurred; storetime records when it was entered or verified.
- itemid: numeric key for labs, measurements, or medications (lookup in d_labitems or d_items).
- value / valuenum / valueuom: textual value, numeric value, and unit respectively; use valuenum for calculations.
- seq_num: ordering field for diagnoses/procedures, where lower values often imply higher priority.
- poe_id / poe_seq: provider order identifier plus sequence; detailed attributes live in poe_detail.
- orderid / linkorderid: link infusion segments and associated orders over time in ICU inputs.
- interpretation: microbiology susceptibility call (e.g., S/I/R).