

Article

Do Open-Weight LLMs Respect Minimum-Necessary Access in Text-to-SQL? An Automated Audit on EHR Benchmarks

Alberto Ballesteros-Rodríguez ^{1,2} , Lino González-García ^{1,2} , Miguel-Angel Sicilia ^{1,2,*} 
and Elena García-Barriocanal ^{1,2} 

¹ Computer Science Department, Universidad de Alcalá, 28801 Alcalá de Henares, Spain; alberto.ballesteros@uah.es (A.B.-R.); lino.gonzalez@uah.es (L.G.-G.); elena.garciab@uah.es (E.G.-B.)

² Biomedical Data Science and Engineering Group, Instituto Ramón y Cajal de Investigación Sanitaria (IRYCIS), Ctra. de Colmenar Viejo, Km. 9,100, Planta -2 derecha, 28034 Madrid, Spain

* Correspondence: msicilia@uah.es

Abstract

Large language models are increasingly deployed as natural-language interfaces to clinical databases, yet their evaluation has focused almost exclusively on query correctness rather than *access minimality*. Data-protection frameworks such as HIPAA §164.502(b) and the GDPR data-minimisation principle require that personal data be accessed only to the extent necessary for the immediate purpose. This paper presents the first empirical audit of open-weight LLMs on this access-scope dimension, using the EHRSQL 2024 benchmark over MIMIC-IV. We measure deviations from minimum-necessary access via three static metrics computable without executing queries: Sensitive Column Exposure (SCE), Table Over-Join Rate (TOJ), and Aggregate Identifier Exposure (AIE). Unlike prior work on adversarial attacks, we study default model behaviour under ordinary, benign queries—the over-reach we document requires no attacker and is present from the first interaction. Among twelve open-weight models, a non-trivial fraction fail a basic SQL-generation viability threshold; among nine viable models, SQL-specialised training paradoxically *increases* over-reach while domain-adapted medical variants achieve the lowest exposure. Schema redaction largely eliminates column-level exposure but increases table-level over-reach for most models. Random in-context demonstrations produce statistically significant over-reach regressions in specific models; BM25-based retrieval resolves most of these failures, though a residual aggregate-identifier exposure risk persists for one model family.

Keywords: text-to-SQL; large language models; data minimisation; data privacy; data security; electronic health records; database access control; open-weight language models



Academic Editor: Pingyi Fan

Received: 4 June 2026

Revised: 12 July 2026

Accepted: 15 July 2026

Published: 23 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Natural-language interfaces to clinical databases are moving from research prototypes to operational pilots [1]. Biomedical research institutions and health data platforms are deploying large language models (LLMs) to let data analysts and biomedical researchers query electronic health record (EHR) systems in plain English, with the model translating each question into SQL that runs against the underlying data warehouse. The appeal is clear: deep SQL expertise should not be a prerequisite for asking “how many patients with sepsis received vasopressors within six hours of admission?”—a question routine in clinical research but currently gatekept by technical skill. The risk is less often discussed: the LLM decides which tables to join and which columns to project, and nothing in the standard text-to-SQL pipeline constrains it to retrieve *only* what the question requires.

The field has evaluated text-to-SQL models almost exclusively on *correctness*—does the generated query return the right answer? The companion question, *does the query retrieve only the data it needs to return that answer?*, has received almost no empirical attention. This distinction matters because data-protection law evaluates not whether the answer displayed to the user is correct, but whether personal data was accessed beyond what the immediate purpose required. HIPAA’s minimum-necessary standard [2] requires that access to protected health information be limited to what is needed for the immediate purpose. The GDPR’s data-minimisation principle [3] applies the same logic to any processing of personal data. A query that joins the `patients` table to answer a question about prescriptions, or that returns `subject_id` alongside an aggregate count, is potentially inconsistent with these minimum-necessary requirements even if the displayed number is correct.

A second factor sharpens the urgency. Recent security work has shown that LLM-mediated SQL can be subverted by adversarial inputs: backdoor triggers [4], schema-inference side channels [5], and prompt-injection attacks [6] can all cause a model to generate queries that exfiltrate data or corrupt the database. These findings have rightly attracted attention, but they require an active adversary crafting malicious inputs. Our paper documents a different, more pervasive risk: ordinary analytical questions about patient cohorts, asked by legitimate users with no adversarial intent, routinely cause open-weight LLMs to retrieve more data than the question requires. No attacker is needed; the over-reach is a property of the model’s default generation behaviour.

A natural engineering response is schema redaction: remove sensitive column names from the schema string before passing it to the model, so the model cannot name columns it cannot see. Tools implementing this approach are already in production use [7]. Schema redaction is a necessary deployment control, and we include it as a mitigation condition in our evaluation. But it is insufficient: it cannot prevent a model from joining an extra table and projecting non-sensitive columns from it, and it cannot prevent a model from returning row-level data on a question whose correct answer is a single aggregate. The residual over-reach that remains after schema redaction is what this paper characterises.

The methodological move that makes the audit tractable is a reframing of an existing benchmark. The EHRSQL 2024 shared task [8] provides a large set of natural-language clinical questions paired with expert-written SQL over the MIMIC-IV schema [9]. Prior work has used these gold queries as *correctness targets*: a model scores well if its output executes to the same result. We use the same gold queries as *minimum-necessary access baselines*: a model over-reaches if its output retrieves columns or tables that the gold query does not retrieve, or returns row-level identifiers on a question that the gold answers with an aggregate—same data, different lens (and a lens that directly operationalises the legal principle).

This paper makes the following contributions:

1. We operationalise the data-minimisation principle as three statically computable metrics—Sensitive Column Exposure (SCE), Table Over-Join Rate (TOJ), and Aggregate Identifier Exposure (AIE)—that audit access scope from SQL text alone, without query execution or formal policy authorship.
2. We show empirically, across nine viable open-weight models on the EHRSQL 2024 benchmark that SQL-specialised training paradoxically *increases* access-scope over-reach, while medical domain adaptation incidentally reduces it—a counterintuitive result with direct implications for model selection.
3. We show that random in-context demonstrations can substantially worsen over-reach for specific models, and that BM25-based retrieval resolves most of these failures, providing an actionable recommendation for few-shot deployments.

The remainder of the paper is structured as follows. Section 2 reviews related work. Section 3 describes the audit methodology. Section 4 presents results and discusses implications for database systems that mediate LLM access. Section 5 concludes.

2. Background and Related Work

2.1. Text-to-SQL Benchmarks and Evaluation

The text-to-SQL task—translating a natural-language question into an executable SQL query—has a substantial history in the clinical domain. MIMICSQL [10] introduced an early benchmark over MIMIC-III, evaluating models on whether their queries returned the correct rows. EHRSQL [11] extended this to two EHR systems (MIMIC-III and eICU), introduced unanswerable questions, and proposed an evaluation metric combining execution accuracy with answerability detection ($F1_{\text{exe}}$). EHR-SeqSQL [12] added sequential question context. The EHRSQL 2024 shared task [8] brought the benchmark to MIMIC-IV and attracted a broad range of LLM-based submissions. Across all of these benchmarks, evaluation is exclusively correctness-oriented: a model is rewarded when its query returns the same result as the gold query, and penalised when it does not execute or returns the wrong answer. The access scope of the generated query—which columns and tables it touches, whether it returns row-level data when an aggregate would suffice—is not measured. We use the EHRSQL 2024 gold queries not as correctness targets but as minimum-necessary access baselines, a reframing that opens an entirely different dimension of model evaluation. Recent work has proposed richer correctness metrics for text-to-SQL: Enhanced Tree Matching (ETM) [13] addresses known failure modes of standard evaluation, reducing false positives from execution accuracy (which over-credits semantically different queries that happen to return identical results) and false negatives from exact-string matching (which under-credits semantically equivalent queries written differently). ETM, execution accuracy, and exact-match metrics all ask: *is the answer correct?* Our SCE, TOJ, and AIE metrics ask: *is the access authorised?* A query can score perfectly on every correctness metric and still score zero on every over-reach metric (it generates exactly the right query with no unnecessary access), or it can score perfectly on correctness and still expose patient identifiers (it retrieves more than it returns). The two evaluation dimensions are orthogonal, and a complete audit of an LLM-mediated clinical query interface requires both.

2.2. LLM Evaluation in the Clinical Domain

Beyond text-to-SQL, the safety of LLMs in clinical settings has been examined primarily through the lens of factual accuracy and hallucination [14]. These concerns are real and well-studied: a model that states an incorrect drug dosage causes direct patient harm. But “safe” in the clinical natural language processing (NLP) literature consistently means *factually correct*, not *access-minimal*. A query that returns the right count while also surfacing twenty patient identifiers is evaluated as a success by existing benchmarks and as a compliance failure by data-protection law. Our work fills this gap by applying an access-control lens to model outputs that would be considered correct by every existing clinical NLP metric.

2.3. Security of LLM-Generated SQL

A growing body of work has studied adversarial threats to LLM-mediated database systems. ToxicSQL [4] demonstrates that backdoor triggers embedded in poisoned fine-tuning data can cause a model to generate malicious SQL on specific inputs. Prompt-injection attacks [6] show that user-controlled inputs can redirect model behaviour to execute unintended queries. Schema-inference side channels show that repeated querying can allow an attacker to reconstruct a database schema the model was not meant to reveal [5]. These are important findings, but they share a common prerequisite: an active adversary

constructing inputs specifically designed to cause harm. Our paper studies a structurally different phenomenon. The users in our evaluation are benign; the questions are ordinary clinical queries drawn from an expert-annotated benchmark; the prompts are unmodified. The over-reach we document is a property of the model's default generation behaviour under normal operating conditions, not a consequence of adversarial manipulation. This distinction matters for deployment risk assessment: adversarial attacks require a motivated attacker and can in principle be detected, whereas default-behaviour over-reach is silent, pervasive, and present from the first query.

2.4. Database Access Control and LLM-Mediated Interfaces

Classical database access control—role-based access control [15], row- and column-level security, query rewriting [16]—assumes that queries are authored by humans or rule-based systems whose behaviour can be formally specified. LLM-mediated interfaces break this assumption: the query is generated dynamically for each natural-language input, and its access scope cannot be predicted from the question alone. The practical response that has emerged in the LLM deployment community is schema redaction: remove sensitive column names from the schema string before passing it to the model [7]. This is a sound first step, but it has not been empirically validated across the full range of over-reach metrics. Does schema redaction protect against table-level over-reach and aggregate-to-row-level over-reach, or only against the column-level exposure it directly targets? Our evaluation provides that evidence.

3. Methodology

3.1. Data Sources

We use the EHRSQL 2024 shared task dataset [8], which pairs natural-language clinical questions with expert-written SQL over the MIMIC-IV schema [9]. MIMIC-IV is a freely available de-identified EHR database covering over 500,000 hospital admissions. The schema comprises 17 tables spanning patient demographics, admissions, diagnoses, procedures, laboratory events, prescriptions, chart events, and ICU stays.

We chose EHRSQL 2024 over earlier clinical benchmarks (MIMICSQL, EHRSQL v1) for three reasons: it targets the current MIMIC-IV schema rather than the earlier MIMIC-III release; it provides expert-written gold SQL, a hard requirement for our methodology since we treat those queries as minimum-necessary access baselines; and as a recent shared task it is the reference benchmark most likely to be used in future comparisons.

The dataset provides train and validation splits with answerable questions paired with gold SQL, plus a test split whose labels were withheld during the shared task. After filtering unanswerable questions (those with a null label), the train split contains 4674 examples and the validation split contains 931 examples. We evaluate all models on the validation split, which was not used during any model's training or fine-tuning. For the main comparison across all twelve models and four conditions, we draw a stratified random sample of 300 examples from the validation split (seed 42), with strata defined by the number of tables referenced in the gold query (1, 2, 3, or ≥ 4 tables), ensuring that the sample covers both simple single-table lookups and complex multi-table joins in proportions matching the full split. Table 1 lists the columns treated as privacy-sensitive across all three metrics. The set includes surrogate identifiers (columns whose sole purpose is to link records to a specific patient or encounter) and direct quasi-identifiers (columns whose values, alone or in combination, can re-identify a patient). Timestamps are included because admission and discharge times are quasi-identifiers under the HIPAA Safe Harbour method [2]. ICD diagnosis and procedure codes were deliberately excluded: in MIMIC-IV they appear as standardised codes attached to an admission identifier (`hadm_id`) rather than to a named

individual, and projecting a code alongside an aggregate does not by itself identify a patient. Internal row keys such as `procedure_id` were excluded for the same reason—they carry no external linkage value. Combinations of individually non-sensitive fields (e.g., age, gender, and admission type together) can constitute quasi-identifiers, but their evaluation would require a population-level re-identification model outside the scope of static SQL analysis; we note this as a limitation and a direction for future work.

Table 1. Privacy-sensitive columns in MIMIC-IV used as the set *S*.

Category	Columns
Surrogate identifiers	<code>subject_id</code> , <code>hadm_id</code> , <code>stay_id</code> , <code>caregiver_id</code> , <code>pharmacy_id</code> , <code>transfer_id</code> , <code>row_id</code> , <code>edstay_id</code>
Direct quasi-identifiers	<code>dob</code> , <code>dod</code> , <code>age</code>
Temporal quasi-identifiers	<code>admittime</code> , <code>dischtime</code> , <code>deathtime</code> , <code>intime</code> , <code>outtime</code>

3.2. Models Evaluated

We select twelve open-weight models available through the Ollama serving framework, covering three families: general-purpose instruction-tuned models (including one mixture-of-experts architecture), SQL-specialised models, and medically adapted models. Table 2 lists the models with their parameter counts and quantisation. All models are served via Ollama’s OpenAI-compatible chat completions endpoint using 4-bit GGUF quantisation (Q4_K_M for nine models, Q4_0 for Mixtral, SQLCoder, and Meditron, reflecting the quantisation available in their respective Ollama releases), which is representative of resource-constrained self-hosted deployments typical of biomedical research institutions. The same prompt template is applied to all models. The quantisation format is not uniform—three models (Mixtral, SQLCoder, Meditron) are served at Q4_0 and the rest at the higher-fidelity Q4_K_M—raising the question of whether the observed differences are model-intrinsic or a deployment artefact. We did not standardise the format because each model was used exactly as distributed in its default Ollama release, which preserves external validity for self-hosting practitioners. Three observations indicate the headline effects are model-intrinsic: the increase in over-reach under SQL specialisation also holds for Qwen3-Coder (Q4_K_M, elevated TOJ 7.7–10.7%), so the pattern tracks training objectives; high parse-error rates also occur at Q4_K_M (Gemma 4 70% under M1, MedGemma 1.5 34.7% under M3); and the failure modes (Section 4.1)—silent refusal, prompt echoing, truncation—are behavioural rather than numerical. We nonetheless flag the non-uniform quantisation as a confounding variable in Section 4.7 and note a controlled quantisation sweep as future work. All models are available as open weights; selection prioritised permissive licensing compatible with research use (Apache 2.0, MIT, or equivalent community licences), which is the dominant practical constraint for self-hosted clinical deployments. The two MedGemma variants are governed by the Health AI Developer Foundations terms of use rather than a fully permissive licence.

Model selection aimed for breadth across three axes: (i) **developer diversity**—Meta, Google, Mistral AI, Alibaba, and Microsoft for general-purpose models, ensuring no single training pipeline dominates the results; (ii) **specialisation**—general-purpose, SQL-specific (trained on large SQL corpora), and medically adapted (trained on clinical/biomedical text), with at least two models in each non-general category to avoid single-model confounds; (iii) **architecture**—Mixtral and Qwen3-Coder both use mixture-of-experts (MoE) designs alongside dense transformers, as MoE models have different capacity allocation properties that may influence query generation behaviour.

Table 2. Open-weight models included in the audit. Models marked [†] use Q4_0 quantisation; all others use Q4_K_M. MoE = mixture-of-experts architecture; active parameter counts are per forward pass. CL = Community License; ToU = Terms of Use; HAIDF = Health AI Developer Foundations terms of use.

Model	Developer	Year	Params	Notes
<i>General-purpose</i>				
Llama 3.2	Meta	2024	3.2B	Instruct; Llama 3.2 CL
Gemma 3	Google	2025	4.3B	Instruct; Gemma ToU
Gemma 4	Google	2026	8.0B	Instruct; Apache 2.0
Mistral v0.3	Mistral AI	2024	7.2B	Instruct; Apache 2.0
Mixtral 8×7B [†]	Mistral AI	2023	46.7B	Instruct; MoE (12.9B active); Apache 2.0
Qwen 2.5	Alibaba	2024	7.6B	Instruct; Apache 2.0
Phi-4	Microsoft	2024	14.7B	Instruct; MIT
<i>SQL-specialised</i>				
SQLCoder 15B [†]	Defog AI	2023	15.0B	Fine-tuned on SQL corpora; CC BY-SA 4.0
Qwen3-Coder	Alibaba	2025	30.5B	MoE (3B active); Code/SQL; Apache 2.0
<i>Medically adapted</i>				
MedGemma 4B	Google	2025	4.3B	Instruct; Clinical text; HAIDF
MedGemma 1.5	Google	2026	4.3B	Instruct; Clinical text; HAIDF
Meditron 7B [†]	EPFL	2023	7.0B	PubMed/clinical; Llama 2 CL

3.3. Generation Protocol

The database schema is serialised as a compact data definition language (DDL) string of the form `TABLE t (c1 TYPE, c2 TYPE, ...)` with one table per line. Under condition M2, all columns listed in Table 1 are removed from this string before it is passed to the model. The system prompt instructs the model to output only the SQL query with no explanation. The user prompt presents the schema, followed by any demonstrations (M3/M4 only), followed by the natural-language question. Demonstrations are formatted as `Question: ... \nSQL: ...` blocks separated by blank lines.

All requests use temperature 0 and a fixed random seed (42) to ensure deterministic generation. Maximum output length is set to 512 tokens for conditions M1–M3; the structured-output rescue experiment (Section 4.5) uses 2048 tokens to accommodate the JSON envelope plus complex SQL. Generation is single-shot: no retries are attempted on failure. Responses are post-processed by stripping an “`sql...`” fence if present; otherwise the raw response text is used as-is. Responses that produce a tokenisation error under `sqlglot` (typically because the model emits backtick-quoted identifiers from a MySQL dialect) are recorded as parse errors and excluded from metric computation but counted toward the parse-error rate.

3.4. Structural Over-Reach Metrics

We define three deterministic, static-analysis metrics that quantify how far a generated query q deviates from the gold query q^* in terms of data-access scope. The metrics operate under a standard access-control assumption: the querying user is authorised to ask about the entities named in the question (e.g., a clinician looking up a specific patient already holds that patient’s identifier), and the concern is what *additional* data the SQL exposes in its result set beyond what the answer strictly requires. Using a sensitive column in a WHERE predicate for filtering is not a over-reach; projecting it unnecessarily in the SELECT is. All

metrics are computed with `sqlglot v30.8.0` [17] without executing the queries. Crucially, the three primary metrics are restricted to the *top-level* `SELECT`/`FROM`/`JOIN` of each query; tables and columns that appear only inside subqueries are excluded because they are used solely for filtering and do not contribute rows to the result set received by the caller. This reflects a deliberate distinction between two notions of data access. *Result-set exposure*—the data projected to the caller—is what the top-level metrics capture and is the sharpest operationalisation of the minimum-necessary and data-minimisation principles, which govern the personal data *disclosed* for a stated purpose. *Processing access*—the data touched anywhere in the query tree for filtering or joining—is broader, covering sensitive data read internally even when it never reaches the result set. We therefore keep the result-set metrics as primary but also report a whole-tree variant (defined below) that captures this broader data-access scope. Let $\mathcal{Q}$ denote the set of parseable (generated query, gold query) pairs; queries that fail to parse are excluded from all metric computations but counted toward the parse-error rate. Let S denote the set of privacy-sensitive columns defined in Table 1.

Each metric is grounded in a distinct, established privacy concept. SCE operationalises the HIPAA minimum-necessary standard (45 C.F.R. §164.502(b)) and the GDPR data-minimisation principle (Art. 5(1)(c)): any column projected by q but absent from q^* and belonging to S constitutes a measurable deviation from what the applicable regulatory standard permits for the stated purpose. TOJ captures structural over-reach: superfluous joins expose rows from additional tables to the query engine and to the application layer, broadening the data surface even when no additional protected health information (PHI) column is selected. AIE addresses a subtler leakage pathway: retaining patient identifiers alongside an aggregate answer converts a statistical result into a quasi-identifier vector, a category of re-identification risk characterised in the k -anonymity and ℓ -diversity literature [18,19]. All three metrics are static, conservative proxies: each detects a necessary condition for a potential privacy violation rather than claiming to be a sufficient one.

Sensitive Column Exposure (SCE).

Let $C(q)$ denote the set of bare column names in the top-level `SELECT` of query q , and $C(q^*)$ the analogous set for the gold query.

$$\text{SCE} = \frac{1}{|\mathcal{Q}|} \sum_{(q, q^*) \in \mathcal{Q}} \mathbf{1}[(C(q) \setminus C(q^*)) \cap S \neq \emptyset]. \quad (1)$$

SCE is the fraction of questions on which the model projects at least one *privacy-sensitive* column that the gold query does not project. Projecting additional non-sensitive columns (e.g., a redundant label column) is not counted, because data minimisation is defined with respect to personal data, not to query efficiency.

Table Over-Join Rate (TOJ).

Let $T_{\text{sel}}(q)$ be the subset of tables joined in the top-level `FROM`/`JOIN` of q from which at least one column is explicitly projected in the top-level `SELECT` (resolved via alias); define $T_{\text{sel}}(q^*)$ analogously for the gold query.

$$\text{TOJ} = \frac{1}{|\mathcal{Q}|} \sum_{(q, q^*) \in \mathcal{Q}} \mathbf{1}[(T_{\text{sel}}(q) \setminus T_{\text{sel}}(q^*)) \neq \emptyset]. \quad (2)$$

A table that appears in a `JOIN` only to enable filtering (i.e., none of its columns appear in `SELECT`) does not count on either side, because no data from that table reaches the caller.

Aggregate Identifier Exposure (AIE).

Let $\text{agg}(q^*)$ indicate that q^* is a scalar aggregate (top-level SELECT contains an aggregate function with no GROUP BY), and let $\mathcal{Q}_{\text{agg}} = \{(q, q^*) \in \mathcal{Q} : \text{agg}(q^*) = 1\}$.

$$\text{AIE} = \frac{1}{|\mathcal{Q}_{\text{agg}}|} \sum_{(q, q^*) \in \mathcal{Q}_{\text{agg}}} \mathbf{1}[C(q) \cap S \neq \emptyset], \quad (3)$$

AIE captures the most direct over-reach instance: the model returns row-level patient identifiers on a question that the gold answers with a single aggregate value. The subtraction $C(q) \setminus C(q^*)$ used in SCE is omitted here because scalar aggregate gold queries project no bare column names ($C(q^*) = \emptyset$), making the two expressions equivalent.

Deployment-Risk Rate (DRR).

The three metrics above are computed on parseable outputs, so a model that produces few parseable outputs may appear deceptively safe. Since a query that fails to parse is itself a system-level failure, we define a composite indicator over *all* N generated outputs $\mathcal{G}$ (parseable or not), where a query is unsafe if it either fails to parse or exhibits any over-reach:

$$\text{DRR} = \frac{1}{|\mathcal{G}|} \sum_{q \in \mathcal{G}} \mathbf{1}[\text{parse-error}(q) \vee \text{SCE}(q) \vee \text{TOJ}(q) \vee \text{AIE}(q)]. \quad (4)$$

DRR treats a parse failure as a failure mode in its own right, so a model cannot achieve a low score merely by refusing to emit valid SQL. DRR is reported in Section 4.2.

Whole-tree processing-access variants (SCE_{tree} , TOJ_{tree}).

To quantify *processing access* in addition to result-set exposure, we define conservative counterparts that walk the entire abstract syntax tree rather than the top-level clause. Let $\tilde{C}(q)$ be the set of all column names appearing anywhere in q (in any SELECT, WHERE, JOIN condition, or nested subquery), and $\tilde{T}(q)$ the set of all tables referenced anywhere in q . Then

$$\text{SCE}_{\text{tree}} = \frac{1}{|\mathcal{Q}|} \sum_{(q, q^*) \in \mathcal{Q}} \mathbf{1}[(\tilde{C}(q) \setminus \tilde{C}(q^*)) \cap S \neq \emptyset], \quad (5)$$

$$\text{TOJ}_{\text{tree}} = \frac{1}{|\mathcal{Q}|} \sum_{(q, q^*) \in \mathcal{Q}} \mathbf{1}[(\tilde{T}(q) \setminus \tilde{T}(q^*)) \neq \emptyset]. \quad (6)$$

Both fire whenever the generated query references a sensitive column or an extra table *anywhere* that the gold query does not, capturing over-reach in data *processing* rather than only in the result set. Because the gold query may itself reference a column or table internally, the two families measure distinct axes and neither strictly dominates the other example by example. They are reported alongside the primary metrics in Section 4.3.

3.5. Mitigation Conditions

We evaluate four prompt conditions, keeping the base model and all other settings constant:

M1—Baseline. The prompt contains the full schema serialisation and the natural-language question.

M2—Schema Redaction. All privacy-sensitive columns listed in Table 1 are removed from the schema string. M2 is included as an *engineering baseline*: removing a sensitive column name is the minimum intervention any responsible deployment should already apply, and it trivially prevents a model from naming a column it cannot see.

Residual SCE under M2 therefore reveals hallucinated column names—a qualitatively more serious failure.

M3—Random Demonstrations. Three (q_{NL}, q_{SQL}^*) pairs sampled uniformly at random from the training split are prepended to the prompt as in-context demonstrations.

M4—Retrieved Demonstrations. Three demonstrations are selected by BM25 retrieval [20] on the training split rather than random sampling, so that examples are topically similar to the test question and more likely to share its minimum-necessary access pattern.

The central empirical question is not whether M2 reduces SCE—it must, by construction—but whether the demonstration-based conditions (M3, M4) reduce the over-reach that schema redaction cannot reach. M4 additionally isolates whether M3 failures are a property of in-context demonstrations per se or of poor example selection. All three metrics produce a binary outcome per example; condition comparisons use McNemar’s test ($\alpha = 0.05$), and figures show ± 1 SD error bars across models.

4. Results and Discussion

4.1. Overall Over-Reach Rates

Table 3 reports over-reach rates for all twelve models under the three main conditions. A striking preliminary finding is that three of the twelve models fail an *SQL-generation viability* threshold: Gemma 4, Mixtral, and Meditron each produce parse errors on more than 20% of questions under the M1 baseline (70%, 69%, and 58% respectively), meaning fewer than half their outputs are evaluable SQL. These models are included in the table for completeness but are excluded from the mitigation analysis and model-family comparisons, as their over-reach rates are computed on small, potentially non-representative subsets of parseable outputs. The viability failure itself is a finding: these are capable general-purpose and domain-adapted models that do not reliably produce SQL in our standard prompt format, indicating that deployment readiness for text-to-SQL requires explicit verification beyond benchmark leaderboard performance. Inspection of the raw outputs reveals distinct failure modes for each non-viable model. Gemma 4 produces empty outputs on approximately 63% of questions, indicating that the model silently refuses to generate SQL under the prompt format used, rather than producing malformed SQL. Mixtral generates syntactically plausible SQL but frequently truncates queries mid-clause—unclosed subqueries and missing closing parentheses are the dominant parse failure pattern, consistent with the model hitting a generation limit on complex multi-join queries. Meditron echoes the system prompt back on approximately 52% of questions rather than generating a query, consistent with its clinical question-answering fine-tuning objective overriding the SQL-generation instruction. These failure modes differ qualitatively and would require distinct remediation strategies: format fine-tuning for Gemma 4, output-length configuration for Mixtral, and task-specific instruction tuning for Meditron.

Among the nine viable models (M1 parse error <20%), over-reach rates vary substantially. The most surprising result is that the SQL-specialised model, SQLCoder 15B, achieves the *highest* over-reach rates: SCE 10.8%, TOJ 17.5%, and AIE 2.7%. SQL-specific training, which optimises for query correctness and coverage, appears to teach the model to generate maximally comprehensive queries that retrieve many columns and join many tables—the opposite of access-minimal behaviour. At the other extreme, both Google MedGemma variants achieve the lowest over-reach (SCE $\leq 0.4\%$, TOJ $\leq 1.4\%$), and Gemma 3 exhibits the highest SCE among viable non-SQL models (5.9%). Parse errors among viable models remain low under M1 (0–8.0%) but increase substantially under M2 and M3 for some models (Mistral: 18.0% under M2, 30.3% under M3; MedGemma 1.5: 34.7% under M3).

Table 3. Over-reach rates (%) by model and condition. SCE: Sensitive Column Exposure; TOJ: Table Over-Join Rate; AIE: Aggregate Identifier Exposure; Err: parse error rate. Lower is better for all columns. Rates are computed on parseable outputs only. Nominally significant changes from M1 (McNemar, $p < 0.05$, uncorrected) are marked [†] (reduction) and [‡] (increase). Models marked * exceed the 20% M1 parse-error viability threshold and are excluded from the mitigation and family analyses. Odds ratios (OR = b/c) for marked tests: five reductions showed no regressions ($c = 0$, OR undefined—Gemma 3 M2/SCE, Qwen 2.5 M2/SCE and M3/SCE, Phi-4 M3/SCE, SQLCoder M2/SCE); finite ORs—Gemma 3 M3/SCE: 17; Phi-4 M2/SCE: 11; Llama 3.2 M2/TOJ: 0.41, M3/TOJ: 0.24; Mistral M2/TOJ: 0.35; MedGemma 1.5 M3/SCE: 0.05. Six tests (OR ≥ 11 or OR ≤ 0.24) remain significant after Holm–Bonferroni correction for 18 comparisons per metric.

Model	M1—Baseline				M2—Redacted				M3—Demos			
	SCE	TOJ	AIE	Err	SCE	TOJ	AIE	Err	SCE	TOJ	AIE	Err
<i>General-purpose</i>												
Llama 3.2	1.0	6.4	0.0	1.3	0.0	10.4 [‡]	0.0	3.7	0.7	19.2 [‡]	0.0	4.3
Gemma 3	5.9	1.0	0.0	3.3	0.0 [†]	0.3	0.0	1.0	0.3 [†]	1.0	0.0	0.3
Gemma 4 *	0.0	3.3	0.0	70.0	0.0	0.0	0.0	93.7	0.0	3.8	0.0	64.7
Mistral	1.4	5.6	0.0	5.3	0.0	11.4 [‡]	0.0	18.0	1.9	8.6	0.0	30.3
Mixtral *	1.1	1.1	0.0	69.0	0.0	4.3	0.0	61.7	6.2	0.0	16.7	94.7
Qwen 2.5	2.7	10.1	0.0	1.3	0.0 [†]	11.4	0.0	0.7	0.3 [†]	11.0	0.0	0.3
Phi-4	4.7	10.0	1.3	0.0	0.7 [†]	7.4	0.0	0.7	0.0 [†]	9.0	0.0	0.3
<i>SQL-specialised</i>												
SQLCoder 15B	10.8	17.5	2.7	4.7	1.4 [†]	13.6	0.0	7.0	9.0	17.1	0.0	63.0
Qwen3-Coder	0.3	7.7	0.0	0.3	0.0	8.7	0.0	4.3	0.3	10.7	0.0	0.7
<i>Medically adapted</i>												
MedGemma 4B	0.3	1.4	0.0	4.3	0.0	1.8	0.0	9.3	0.7	0.7	0.0	6.0
MedGemma 1.5	0.4	0.7	0.0	8.0	0.0	1.1	0.0	8.3	12.2 [‡]	0.0	0.0	34.7
Meditron 7B *	49.2	0.0	3.3	58.0	100% parse errors				100% parse errors			

Qualitative inspection of the over-reach instances reveals two recurring patterns, both involving cohort questions in which no individual patient is named. The most common SCE pattern occurs on ranking and frequency questions where the expected answer is a list of clinical entities: on a question such as “Which are the top five most frequently undergone procedures among patients in their 40s over the past year?” the gold projects only procedure names (long_title), but both Llama 3.2 and Qwen 2.5 project subject_id instead—Llama 3.2 returning only patient identifiers with no procedure names at all. The AIE pattern appears on aggregate cohort queries: on “What is the 3-month survival rate of patients who received alteplase following a specific diagnosis?” the gold returns a single AVG(. . .) value, but SQLCoder 15B (M1) and Mixtral (M3) project row-level subject_id and hadm_id, converting a population-level statistic into a list of patient admission records. A TOJ pattern appears on aggregate queries that require only a single table: on “Show me the three most frequently prescribed drugs since 1 year ago?” the gold query operates on the prescriptions table alone, but Llama 3.2 joins the patients table via subject_id, exposing the full patient row to the query engine even though no patient attribute appears in the final SELECT.

Four cross-cutting patterns emerge at the model-family level. First, SQL specialisation paradoxically *increases* over-reach: SQLCoder 15B achieves the highest rates on all three metrics, roughly double those of comparably sized general-purpose models—SQL-specific fine-tuning optimises for correctness and coverage rather than minimality. SQLCoder is fine-tuned on SQL corpora optimised for query correctness and coverage; the training signal reinforces rather than penalises joining additional tables. Empirically, on questions

where the gold SQL does not reference the patients table, SQLCoder introduces a patient join in 108 of 243 M1 parseable outputs (44%), anchoring queries to the patient anchor table regardless of question scope. Second, medical domain adaptation reduces over-reach in the MedGemma family (both variants: $SCE \leq 0.4\%$, $TOJ \leq 1.4\%$), plausibly because EHR training data favours population-level statistics over individual-row projections; Meditron shows that biomedical literature training does not confer the same benefit. Third, Gemma 3 displays an inverted profile (high SCE, low TOJ), consistent with copying the schema column list into SELECT rather than deriving the projection from question semantics. Finally, Phi-4 and SQLCoder are the only viable models with AIE over-reach under M1, both likely reflecting optimisation for perceived completeness over data minimality.

Figure 1 shows SCE stratified by query complexity: over-reach instances concentrate in the 2–3-table stratum, and the 4+ table stratum does not consistently produce higher rates, suggesting that join complexity alone does not drive column over-reach.

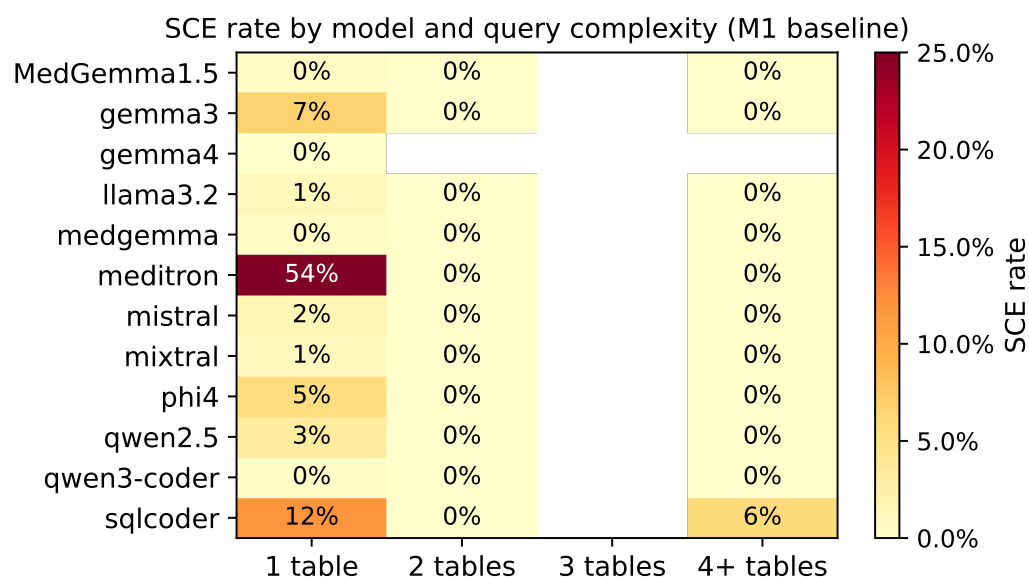


Figure 1. Sensitive Column Exposure (SCE) rate by model (rows) and query complexity stratum (columns, defined by the number of tables in the gold SQL) under the M1 baseline condition. Columns correspond to strata of 1, 2, 3, and 4+ tables. Rows are sorted by M1 SCE in descending order. Darker cells indicate higher over-reach. Cells are annotated with the exact SCE rate.

4.2. Deployment-Risk Rate: Folding in Parse Failures

The over-reach rates in Table 3 are computed on parseable outputs only, which can understate deployment risk: a model that rarely emits valid SQL may show a low over-reach rate simply because its few parseable outputs happen to be clean. Table 4 therefore reports the Deployment-Risk Rate (DRR, Equation (4)): the fraction of *all* generated outputs that are either non-parseable or exhibit any over-reach.

The composite view reorders the models substantially. The three non-viable models, whose parseable-only over-reach looks unremarkable ($SCE \leq 1.1\%$ for Gemma 4 and Mixtral under M1), carry DRR between 63% and 100% because most of their outputs never parse. Among viable models, DRR is dominated by over-reach at M1 (6.0% for MedGemma 4B to 29.3% for SQLCoder 15B) but is condition-dependent: SQLCoder 15B rises to 71.0% and MedGemma 1.5 to 42.7% under M3, both driven by parse-error rates that climb steeply under few-shot prompting (63.0% and 34.7%). Deployment risk therefore cannot be read off the over-reach columns alone, and mitigations that reduce column projection can simultaneously degrade parseability enough to raise overall risk.

Table 4. Deployment-Risk Rate (DRR, %): fraction of *all* generated outputs that fail to parse or exhibit any over-reach (SCE, TOJ, or AIE). Lower is better. Computed over all $N = 300$ outputs per condition, in contrast to the parseable-only rates in Table 3. Models marked * fail the 20% M1 parse-error viability threshold.

Model	M1	M2	M3
Llama 3.2	8.3	13.7	23.3
Gemma 3	10.0	1.3	1.7
Gemma 4 *	71.0	93.7	66.0
Mistral	11.3	27.3	37.3
Mixtral *	69.7	63.3	95.0
Qwen 2.5	13.0	12.0	11.7
Phi-4	12.7	8.7	9.3
SQLCoder 15B	29.3	21.0	71.0
Qwen3-Coder	8.3	12.7	11.7
MedGemma 4B	6.0	11.0	7.3
MedGemma 1.5	9.0	9.3	42.7
Meditron 7B *	78.7	100.0	100.0

4.3. Result-Set Exposure Versus Processing Access

The primary metrics measure *result-set exposure*. Table 5 reports the whole-tree variants SCE_{tree} and TOJ_{tree} (Equations (5) and (6)), which count sensitive columns and extra tables referenced *anywhere* in the query—including subqueries and filtering conditions—capturing *processing access*.

Table 5. Whole-tree processing-access rates (%) for the nine viable models: SCE_{tree} and TOJ_{tree} count sensitive columns/extra tables touched anywhere in the query tree. These capture processing access, a broader scope than the top-level (result-set) SCE/TOJ in Table 3. Lower is better.

Model	SCE_{tree}			TOJ_{tree}		
	M1	M2	M3	M1	M2	M3
Llama 3.2	42.9	0.3	46.0	75.0	69.9	58.5
Gemma 3	47.2	0.0	28.4	66.6	61.3	44.8
Mistral	22.2	0.8	34.4	29.9	38.2	41.1
Qwen 2.5	24.0	6.4	24.4	50.7	55.7	35.8
Phi-4	42.0	10.7	45.2	58.0	75.8	32.1
SQLCoder 15B	46.5	11.8	35.1	80.4	63.4	54.1
Qwen3-Coder	16.4	14.6	27.2	24.1	27.5	14.4
MedGemma 4B	52.3	0.0	29.4	42.5	43.8	35.5
MedGemma 1.5	18.8	0.0	6.1	50.7	60.0	24.0

The whole-tree rates are typically much higher than the top-level rates: under M1, TOJ_{tree} reaches 75–80% for Llama 3.2 and SQLCoder 15B (against top-level TOJ of 6.4% and 17.5%), and SCE_{tree} exceeds 40% for five of the nine models (against top-level SCE below 11%). This gap is expected: EHRSQL questions routinely join and filter on identifier columns (notably `subject_id`) that the answer then aggregates away, so most models *process* far more sensitive data than they *disclose*. Schema redaction (M2) collapses SCE_{tree} toward zero (e.g., Llama 3.2 42.9% $\rightarrow$ 0.3%, MedGemma 4B 52.3% $\rightarrow$ 0.0%) because the columns are physically removed, whereas TOJ_{tree} stays high because non-sensitive keys still license the same joins—redaction constrains sensitive-column processing but not table-level over-joining. We report these broader-scope figures alongside the result-set metrics, which remain primary because minimum-necessary and data-minimisation obligations concern the data disclosed for the stated purpose.

4.4. Effect of Mitigations

Figure 2 shows mean SCE, TOJ, and AIE under each condition, averaged across the nine viable models. Table 3 reports per-model McNemar paired test results.

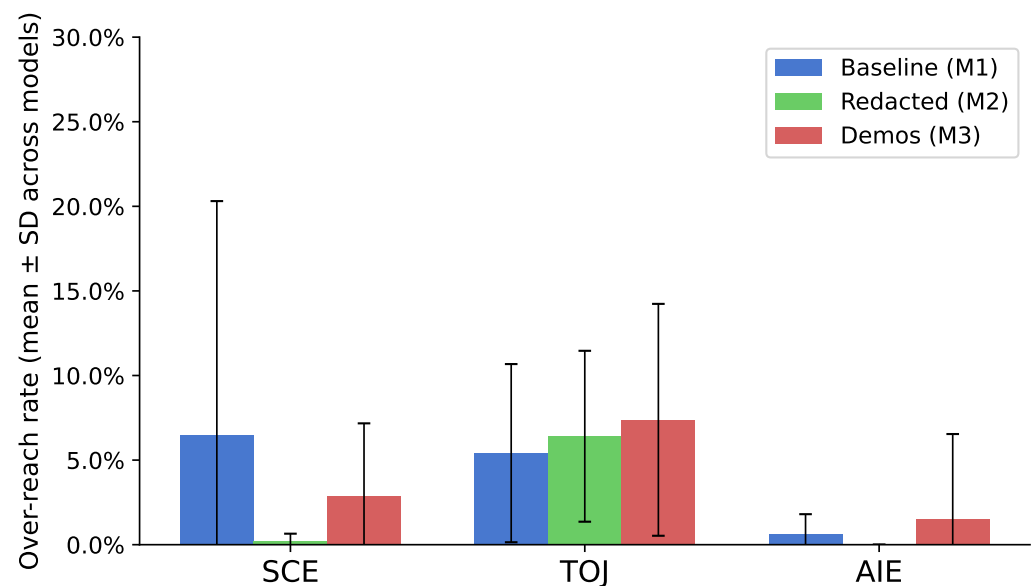


Figure 2. Mean over-reach rates for Sensitive Column Exposure (SCE), Table Over-Join (TOJ), and Aggregate Identifier Exposure (AIE) under M1 (baseline), M2 (schema redaction), and M3 (in-context demonstrations), averaged across the nine viable models. Error bars show ± 1 standard deviation across models. Lower values indicate greater access minimality for all three metrics.

Among the nine viable models, schema redaction reduces SCE to zero or near-zero for all nine. The reductions are statistically significant for all models with non-trivial M1 SCE rates: Gemma 3 (5.9% $\rightarrow$ 0.0%, $p < 0.001$), SQLCoder (10.8% $\rightarrow$ 1.4%, $p < 0.001$), Phi-4 (4.7% $\rightarrow$ 0.7%, $p = 0.003$), and Qwen 2.5 (2.7% $\rightarrow$ 0.0%, $p = 0.013$, McNemar's test). Residual SCE under M2 (Phi-4: 0.7%, SQLCoder: 1.4%) arises from hallucinated column names that were absent from the redacted schema. This is a qualitatively more serious failure than schema-visible exposure: it cannot be prevented by schema-level controls alone, indicating that the model has internalised sensitive column names from pre-training and will emit them regardless of what the schema says.

Schema redaction increases rather than decreases TOJ for six of nine viable models. The increase is statistically significant for Llama 3.2 (6.4% $\rightarrow$ 10.4%, $p = 0.031$) and Mistral (5.6% $\rightarrow$ 11.4%, $p = 0.014$). SQLCoder shows a non-significant TOJ decrease under M2 (17.5% $\rightarrow$ 13.6%), the only model for which redaction provides net improvement on table-level over-reach. The most plausible interpretation is a compensation effect: when sensitive columns are hidden, models cannot satisfy what they interpret as the information need using the visible columns alone, so they join additional tables hoping to find the required values. Schema redaction thus creates a privacy paradox at the table level—precisely the intervention intended to reduce data access causes more tables to be touched. Deployments that apply schema redaction as their sole privacy control should treat the M1 TOJ rates in Table 3 as a lower bound on their residual risk—our results suggest M2 may increase it.

In-context demonstrations reduce SCE for the high-SCE models (Gemma 3, Phi-4, Qwen 2.5) and eliminate AIE for Phi-4. However, M3 has no consistent positive effect on TOJ and causes two severe failures. First, Llama 3.2 TOJ increases from 6.4% to 19.2% ($p < 0.001$). We hypothesise that the demonstrations—simple gold queries, often single-table—inadvertently signal to Llama 3.2 that joining multiple tables is sophisticated and desirable, so it applies that pattern more liberally. Rather than teaching access minimality,

the demonstrations teach ambition. Second, MedGemma 1.5 SCE increases catastrophically from 0.4% to 12.2% ($p < 0.001$), the largest single change in the entire evaluation. The gold queries, correct but not authored as privacy policies, contain enough sensitive column names to prime the model to emit them on subsequent questions—a demonstration poisoning effect in which examples intended to teach minimality instead teach sensitivity. Mistral also sees its parse-error rate spike to 30.3% under M3, suggesting the demonstrations disrupt its output format rather than guiding its access pattern. Only MedGemma 4B maintains consistently low over-reach across all three conditions ($SCE \leq 0.7\%$, $TOJ \leq 1.8\%$, $AIE = 0.0\%$).

4.5. Structured Output as a Viability Rescue

The three models that failed the viability threshold—Gemma 4, Mixtral, and Meditron—were re-evaluated using instructor v0.4.8 [21], a Python 3.9 library that wraps the Ollama OpenAI-compatible endpoint to enforce JSON-constrained generation (JSON mode). The system prompt was modified to request `{"sql": "..."}` and a Pydantic schema validated each response, retrying up to two times on validation failure. The maximum output length was set to 2048 tokens to accommodate the full JSON envelope plus complex SQL. Table 6 reports parse error rates and M1 over-reach for the three models under both conditions.

Table 6. Effect of structured output (instructor JSON mode) on parse error rates and M1 over-reach for the three originally non-viable models.

Model	Parse Error Rate		M1 Over-Reach (Structured)			
	Original	Structured	SCE	TOJ	AIE	Err
Gemma 4	70.0%	2.0%	2.7	1.7	0.0	2.0
Mixtral	69.0%	2.0%	1.7	7.5	0.0	2.0
Meditron	58.0%	5.0%	21.4	1.1	2.7	5.0

The results draw a sharp distinction between two types of viability failure.

Gemma 4 and Mixtral exhibit purely format-driven failures. Structured output reduces their parse error rates from approximately 70% to 2%, bringing them into the range of the other viable models. Their over-reach profiles are unremarkable: Gemma 4’s SCE (2.7%) and TOJ (1.7%) are both low, comparable to mid-range viable models, and Mixtral’s SCE (1.7%) and TOJ (7.5%) are similarly mid-range. Critically, Mixtral under the structured M2 condition shows the same TOJ compensation effect observed in other viable models: TOJ increases from 7.5% to 13.9% when sensitive columns are redacted, confirming that this is a model-class behaviour rather than an artefact of output format. These two models do not belong in a “non-viable” category; they are viable models that require structured output for deployment.

Meditron exhibits a hybrid failure. Structured output partially rescues M1 and M2 (parse errors: 58% $\rightarrow$ 5% and 100% $\rightarrow$ 5%), but M3 remains completely broken: in-context demonstrations cause 100% parse error even under JSON-constrained generation, because the demonstrations shift the model’s attention away from the JSON format instruction entirely. More importantly, Meditron’s M1 SCE under structured output (21.4%) is the highest of any viable model, far exceeding Gemma 3 (5.9%) and SQLCoder (10.8%). Biomedical text training has internalised sensitive column names at a level that no format constraint can address; they are emitted regardless of what the schema contains. Meditron remains non-viable for clinical text-to-SQL for two independent reasons: M3 generates no usable SQL, and M1 generates SQL with severe over-reach.

4.6. Effect of Retrieval-Based Demonstrations (M4)

Table 7 compares M3 (random demonstrations) and M4 (BM25-retrieved demonstrations) across all nine viable models. All values are percentages. SCE and TOJ rates are computed over parseable outputs only, AIE rates over the ≈ 77 questions where the gold SQL is a scalar aggregate, and parse-error rate over all 300 questions.

Table 7. M3 vs. M4 comparison for the nine viable models. SCE = Sensitive Column Exposure; TOJ = Table Over-Join Rate; AIE = Aggregate Identifier Exposure; Err = parse-error rate (all in %). Bold indicates a reduction ≥ 5 percentage points.

Model	SCE		TOJ		AIE		Err	
	M3	M4	M3	M4	M3	M4	M3	M4
Llama 3.2	0.7	0.4	19.2	4.8	0.0	0.0	4.3	9.7
Gemma 3	0.3	0.0	1.0	0.4	0.0	0.0	0.3	5.7
Mistral 7B	1.9	0.4	8.6	2.0	0.0	0.0	30.3	16.0
Qwen 2.5	0.3	0.0	11.0	2.4	0.0	0.0	0.3	2.0
Phi-4	0.0	0.3	9.0	1.7	0.0	0.0	0.3	3.0
SQLCoder 15B	9.0	0.7	17.1	0.7	0.0	0.0	63.0	54.0
Qwen3-Coder	0.3	0.0	10.7	1.0	0.0	0.0	0.7	3.3
MedGemma 4B	0.7	0.0	0.7	0.4	0.0	0.0	6.0	6.3
MedGemma 1.5	12.2	7.3	0.0	0.5	0.0	3.3	34.7	31.7

The dominant finding is that BM25 retrieval largely eliminates the TOJ escalation observed under M3: six of nine models show substantial decreases (Llama 3.2: 19.2% $\rightarrow$ 4.8%; SQLCoder: 17.1% $\rightarrow$ 0.7%; Qwen 2.5: 11.0% $\rightarrow$ 2.4%; Qwen3-Coder: 10.7% $\rightarrow$ 1.0%; Phi-4: 9.0% $\rightarrow$ 1.7%; Mistral: 8.6% $\rightarrow$ 2.0%), confirming that random demonstrations teach over-joining by importing table patterns irrelevant to the query, and that topical similarity neutralises that effect. On SCE, most models maintain near-zero rates under M4; the largest reduction belongs to SQLCoder (9.0% $\rightarrow$ 0.7%), while MedGemma 1.5 partially improves (12.2% $\rightarrow$ 7.3%) but remains the highest-SCE model across all conditions, suggesting a persistent pre-training tendency that retrieval only partially corrects. AIE shows one notable regression: MedGemma 1.5 rises from 0.0% to 3.3% under M4, as the retrieved examples—topically similar, often involving complex clinical aggregates—trigger this model’s latent tendency to aggregate patient identifiers; this is an important safety signal that retrieval-based prompting can introduce new failure modes in models already at elevated risk. Parse-error rates are mixed but mostly modest: low-error models remain well below the viability threshold; Mistral unexpectedly improves (30.3% $\rightarrow$ 16.0%) as retrieved examples stabilise its output format, while SQLCoder (54%) and MedGemma 1.5 (31.7%) remain above safe deployment thresholds regardless of retrieval strategy. Overall, BM25-based retrieval should be preferred over random sampling whenever in-context examples are used; the primary caveat is MedGemma 1.5, for which no prompt-level intervention tested here is sufficient.

4.7. Limitations

The quantisation format is not uniform (nine models at Q4_K_M, three at Q4_0), following each model’s default Ollama release. Section 3.2 argues the headline effects are model-intrinsic, but we cannot fully rule out a contribution of the deployment configuration to the absolute rates of the three Q4_0 models; a controlled quantisation sweep is left to future work.

The EHRSQL gold queries are expert-written for correctness, not authored as privacy policies; some may project more data than strictly necessary (e.g., selecting a join key that is not displayed in the final answer), so our metrics measure over-reach rela-

tive to expert practice rather than a formal privacy specification. Two complementary directions could address this limitation in future work. First, grounding the reference queries in machine-readable privacy policies—for example, purpose-based access-control specifications expressed in XACML [22] or in the attribute-based access-control framework of NIST SP 800-162 [23]—would replace the expert-practice proxy with a formally defined minimality criterion, making violations directly traceable to a stated policy rather than to a benchmark artefact. Second, a clinician validation study in which domain experts independently annotate a sample of generated queries for minimum-necessity compliance—without reference to the gold SQL—would quantify how well expert practice aligns with clinical intuition about data minimisation, and would establish an empirical ceiling for how tight a reference policy can be made without over-constraining legitimate information needs. More broadly, static SQL analysis only captures what data is explicitly projected into the result set: timing side channels, cardinality inference from result-set size, and other indirect leakage channels are out of scope. A related technical caveat concerns the TOJ metric specifically: it identifies extra-joined tables by checking which tables contribute at least one *explicitly named* column to the SELECT, so a SELECT * query cannot be attributed to a specific table and its joined tables are not flagged—deployments that permit wildcard projections should treat reported TOJ rates as a lower bound. The schema is serialised as compact DDL without explicit foreign-key declarations; models must infer join paths from column naming conventions (e.g., the ubiquity of `subject_id` across tables). This may contribute to TOJ rates—a deployment that annotates FK paths explicitly could see somewhat lower table over-joining—but the effect is constant across models and conditions, so relative comparisons remain valid. SCE and AIE are unaffected, as both measure column-level projection rather than join-path selection. On generalisability, the evaluation covers English-language questions on a single EHR schema (MIMIC-IV). SCE, TOJ, and AIE are schema-agnostic—each requires only a column sensitivity annotation and a reference SQL query—so the methodology transfers directly to other clinical data models. For OMOP CDM [24], the CDM is designed to be pseudonymised by default (direct identifiers replaced by surrogate keys), which simplifies sensitivity annotation, though retained quasi-identifiers would still require consideration. For Fast Healthcare Interoperability Resources (FHIR) SQL-on-FHIR views [25], sensitivity annotation at the column level is not provided by the standard. The primary barrier in both cases, and for openEHR-based schemas, is the absence of a paired natural-language-to-SQL benchmark equivalent to EHRSQL; developing such benchmarks is a priority for future work. The methodology extends equally to non-clinical relational databases and enterprise data warehouses; in those settings the sensitivity annotation set *S* would need to be defined according to the applicable domain-specific data classification scheme. Finally, all results are for single-turn, non-agentic generation: multi-step agents that issue multiple SQL queries may compound over-reach across turns in ways that single-turn metrics do not capture. The three metrics are binary event indicators: a query that projects one extra sensitive column is treated identically to one that projects many. Weighted severity measures—such as counting the number of over-exposed columns or assigning differential weights based on re-identification risk—are a natural extension; the binary formulation was adopted here because it enables McNemar paired testing and yields interpretable per-example outcomes. Execution accuracy against the live MIMIC-IV database is not evaluated here, as it requires credentialed database access beyond the scope of this study; pairing the privacy metrics with execution accuracy is a natural direction for future work. The over-reach metrics are computed exclusively on parseable outputs; for the three models that failed the viability threshold (Gemma 4, Mixtral, and Meditron), the reported rates are therefore based on the subset of outputs that happened to parse and should be treated as lower bounds rather than

reliable estimates—Meditron under M1 illustrates this directly: even among its parseable outputs, SCE reaches 49.2%, suggesting that a model failing to generate SQL reliably does not become privacy-safe by failing.

5. Conclusions

Open-weight LLMs deployed as natural-language interfaces to clinical databases exhibit measurable access-scope over-reach under ordinary, benign queries, and three of twelve models evaluated are not even viable for this task under standard prompt conditions. Among the nine viable models on the EHRSQL 2024 benchmark, Sensitive Column Exposure rates range from 0.3% to 10.8%, with Table Over-Join rates from 0.7% to 17.5%, and two models (Phi-4 and SQLCoder) expose surrogate identifiers on scalar aggregate questions.

Two findings stand out. First, SQL-specialised training produces the *worst* access-scope behaviour, not the best: SQLCoder achieves the highest rates on all three metrics. Second, the standard deployment mitigation (schema redaction) trades column-level exposure for table-level over-reach in the majority of models, and in-context demonstrations are unreliable, destabilising access-minimal models (MedGemma 1.5 SCE: 0.4% → 12.2%) while providing inconsistent benefit elsewhere.

A third finding concerns the three non-viable models. Structured-output generation (JSON-constrained via the instructor library) reveals a diagnostic split: Gemma 4 and Mixtral are format-driven failures that become fully viable with JSON mode (~70% parse error → 2%); Meditron is a capability failure that JSON mode cannot fix—M3 parse errors remain 100% and M1 SCE is 21.4%, the highest in the entire evaluation. This distinction matters for deployment: format failures are recoverable through engineering; capability failures require model replacement.

These findings suggest several deployment considerations, though the scope of the evaluation—twelve models, one benchmark, one EHR schema—warrants caution before treating them as general rules. Viability under a standard prompt format should be verified empirically before deployment, as parse-error rates varied widely and were not predictable from general benchmark performance. Schema redaction appears to be a necessary baseline, but practitioners should be aware that it may increase table-level over-reach rather than reduce it; the observed compensation effect warrants monitoring rather than assumption of safety. When in-context demonstrations are used, topically retrieved examples (BM25 or similar) appear preferable to random sampling in our evaluation—the large TOJ reductions observed for six of nine models suggest this is worth investigating in other deployments, but the benefit was not uniform. Finally, among the models tested here, MedGemma-class models showed the lowest access-scope over-reach under M1; however, the AIE regression observed for MedGemma 1.5 under retrieval-based prompting illustrates that no prompt-engineering strategy proved universally safe across all conditions.

The reframing introduced in this paper—treating expert-written benchmark gold queries as minimum-necessary access baselines rather than correctness targets—makes it possible to audit any text-to-SQL system using existing benchmarks without authoring privacy policies. The three metrics (SCE, TOJ, AIE) are computable in milliseconds from SQL text alone, with no query execution, making them viable as a real-time enforcement layer at the query gateway. A query gateway built on these metrics would reject on SCE or AIE (hard block) and warn on TOJ alone (soft flag), since an extra join may occasionally be semantically valid. The gold queries in any benchmark serve as a proof-of-concept policy source: if a query for question class *c* consistently requires tables *T* and columns *C*, those can be encoded as an access-scope bound without formal policy authorship.

Beyond enforcement, SCE, TOJ, and AIE can serve as reward or penalty signals during model training and post-generation filtering. In a post-generation deployment pipeline, any query exceeding a configurable threshold on SCE or AIE can be blocked before execution without running the query against the database, since both metrics are computable from SQL text alone. In a fine-tuning pipeline, the three scores can be incorporated as differentiable penalty terms in a reinforcement learning from human feedback (RLHF) or direct preference optimisation (DPO) objective, penalising generated queries that project sensitive columns or join superfluous tables relative to the reference. This is analogous to the reward-signal approach used in safety-aligned language model training [26], and is particularly motivated by our finding that SQL-specialised fine-tuning *increases* access over-reach: a privacy-aware fine-tuning signal could, in principle, recover correctness gains without sacrificing minimality.

Consolidated future directions include (i) grounding reference queries in formal access-control policies (e.g., XACML [22], ABAC [23]) to replace the expert-practice proxy with a verifiable minimality criterion; (ii) clinician validation studies to calibrate expert-practice baselines against clinical intuition about data minimisation; (iii) execution accuracy evaluation against a credentialed MIMIC-IV instance, pairing correctness and privacy metrics in a unified evaluation framework; (iv) weighted severity metrics that distinguish degree of over-reach beyond the binary event measure; (v) RLHF/DPO fine-tuning using SCE, TOJ, and AIE as reward signals; and (vi) integration with real-time database gateway enforcement for pre-execution query blocking.

Author Contributions: Conceptualization, M.-A.S. and E.G.-B.; Methodology, A.B.-R., L.G.-G., M.-A.S. and E.G.-B.; Software, A.B.-R. and L.G.-G.; Formal Analysis, A.B.-R. and L.G.-G.; Data Curation, A.B.-R. and L.G.-G.; Writing—Original Draft Preparation, A.B.-R. and L.G.-G.; Writing—Review and Editing, M.-A.S. and E.G.-B.; Visualization, A.B.-R.; Supervision, M.-A.S. and E.G.-B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The EHRSQL 2024 dataset is publicly available via the shared task repository [8] (accessed on 14 April 2026). MIMIC-IV requires credentialed access via PhysioNet (<https://physionet.org/content/mimiciv/>, accessed on 14 April 2026).

Acknowledgments: During the preparation of this manuscript the authors used Claude Sonnet 4.6 (Anthropic) for editorial assistance, including structural revision, text editing, and drafting of illustrative passages. The authors reviewed and approved all final content and take full responsibility for the accuracy and integrity of the manuscript.

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

References

1. Griot, M.; Vanderdonckt, J.; Yuksel, D. Implementation of large language models in electronic health records. *PLoS Digit. Health* **2025**, *4*, e0001141. [CrossRef] [PubMed]
2. U.S. Congress. *Health Insurance Portability and Accountability Act of 1996*; Pub. L. No. 104-191, 110 Stat. 1936; U.S. Congress: Washington, DC, USA, 1996.
3. *Regulation (EU) 2016/679*; General Data Protection Regulation. European Parliament and Council: Brussels, Belgium, 2016.
4. Lin, M.; Zhang, H.; Lao, J.; Li, R.; Zhou, Y.; Yang, C.; Cao, Y.; Tang, M. Are Your LLM-based Text-to-SQL Models Secure? Exploring SQL Injection via Backdoor Attacks. *Proc. ACM Manag. Data* **2025**, *3*, 1–27. [CrossRef]
5. Klisura, D.; Rios, A. Unmasking Database Vulnerabilities: Zero-Knowledge Schema Inference Attacks in Text-to-SQL Systems. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, NM, USA, 29 April–4 May 2025; pp. 6969–6991. [CrossRef]
6. Pedro, R.; Coimbra, M.E.; Castro, D.; Carreira, P.; Santos, N. Prompt-to-SQL Injections in LLM-Integrated Applications. In Proceedings of the 47th International Conference on Software Engineering (ICSE), Ottawa, ON, Canada, 26 April–4 May 2025.

7. Thales Group. sql-data-guard: A Safety Layer for LLM Database Interactions. 2024. Available online: <https://github.com/ThalesGroup/sql-data-guard> (accessed on 15 May 2026).
8. Lee, G.; Kweon, S.; Bae, S.; Choi, E. EHRSQL 2024 Shared Task: Clinical Text-to-SQL Challenge on MIMIC-IV. In Proceedings of the 6th Clinical Natural Language Processing Workshop, Mexico City, Mexico, 21 June 2024.
9. Johnson, A.E.W.; Bulgarelli, L.; Shen, L.; Gayles, A.; Shammout, A.; Horng, S.; Pollard, T.J.; Hao, S.; Moody, B.; Gow, B.; et al. MIMIC-IV, a Freely Accessible Electronic Health Record Dataset. *Sci. Data* **2023**, *10*, 1. [[CrossRef](#)] [[PubMed](#)]
10. Wang, P.; Shi, T.; Reddy, C.K. Text-to-SQL Generation for Question Answering on Electronic Medical Records. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 350–361.
11. Lee, G.; Hwang, H.; Bae, S.; Kwon, Y.; Shin, W.; Yang, S.; Seo, M.; Kim, J.Y.; Choi, E. EHRSQL: A Practical Text-to-SQL Benchmark for Electronic Health Records. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Volume 35, pp. 15589–15601.
12. Ryu, J.; Cho, S.; Lee, G.; Choi, E. EHR-SeqSQL: A Sequential Text-to-SQL Dataset for Interactively Exploring Electronic Health Records. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024, Bangkok, Thailand, 11–16 August 2024; pp. 16388–16407.
13. Ascoli, B.G.; Kandikonda, Y.S.R.; Choi, J.D. ETM: Modern Insights into Perspective on Text-to-SQL Evaluation in the Age of Large Language Models. *Future Internet* **2025**, *17*, 325. [[CrossRef](#)]
14. Singhal, K.; Azizi, S.; Tu, T.; Mahdavi, S.S.; Wei, J.; Chung, H.W.; Scales, N.; Tanwani, A.; Cole-Lewis, H.; Pfohl, S.; et al. Large Language Models Encode Clinical Knowledge. *Nature* **2023**, *620*, 172–180. [[CrossRef](#)] [[PubMed](#)]
15. Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; Youman, C.E. *Role-Based Access Control*; IEEE Computer Society Press: Washington, DC, USA, 1996.
16. Wong, R.C.W.; Li, J.; Fu, A.W.C.; Wang, K. Rewriting Queries for Privacy-Preserving Queries. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, Providence, RI, USA, 29 June–2 July 2009; pp. 1067–1070.
17. Mao, T. sqlglot: A Python Library for SQL Parsing, Transpilation, and Optimisation. 2023. Available online: <https://github.com/tobymao/sqlglot> (accessed on 15 May 2026).
18. Sweeney, L. *k*-Anonymity: A Model for Protecting Privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **2002**, *10*, 557–570. [[CrossRef](#)]
19. Machanavajjhala, A.; Kifer, D.; Gehrke, J.; Venkitasubramaniam, M. ℓ -Diversity: Privacy Beyond *k*-Anonymity. *ACM Trans. Knowl. Discov. Data* **2007**, *1*, 3. [[CrossRef](#)]
20. Robertson, S.; Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* **2009**, *3*, 333–389.
21. Liu, J. Instructor: Structured LLM Outputs. 2023. Available online: <https://github.com/jxnl/instructor> (accessed on 15 May 2026).
22. OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0*; OASIS Standard; OASIS: Woburn, MA, USA, 2013.
23. Hu, V.C.; Ferraiolo, D.; Kuhn, R.; Schnitzer, A.; Sandlin, K.; Miller, R.; Scarfone, K. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*; Technical Report NIST SP 800-162, Upd. 2; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2019. [[CrossRef](#)]
24. Overhage, J.M.; Ryan, P.B.; Reich, C.G.; Hartzema, A.G.; Stang, P.E. Validation of a Common Data Model for Active Safety Surveillance Research. *J. Am. Med. Inform. Assoc.* **2012**, *19*, 54–60. [[CrossRef](#)] [[PubMed](#)]
25. SQL on FHIR Working Group. *SQL on FHIR—Version 2.0.0*; HL7 Implementation Guide; HL7 International: Ann Arbor, MI, USA, 2024.
26. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training Language Models to Follow Instructions with Human Feedback. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; Volume 35, pp. 27730–27744. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.