DRIFTBENCH: AN EXECUTION-BASED BENCHMARK FOR LLM SQL QUERY REPAIR UNDER SCHEMA EVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Production analytics break during schema evolution: a benign table or column rename can invalidate thousands of downstream queries. Large language models (LLMs) are an attractive repair tool. However, the field lacks *execution-based* benchmarks that (i) guarantee the drift actually affects the query, (ii) preserve semantics through migrated data, and (iii) score correctness beyond syntactic validity. We present **DriftBench**, a procedural generator and evaluation harness for SQL repair under schema evolution. It builds paired *original* and *drifted* databases in isolated namespaces, migrates data through canonical migration plans, and derives oracle repairs by AST rewriting, so a repair counts as correct only if it reproduces the pre-drift result set. Across 18,624 system evaluations spanning seven LLMs, a deterministic schema-diff oracle, and a fuzzy heuristic, the best model reaches **92.0%** semantic equivalence on the 2,000-instance rename benchmark against 13.7% for the heuristic. Under the default prompt, execution success overstates repair quality for every LLM on every rename suite; on the 292-instance rename suite, the gap is 5.9–19.9 points for LLMs and 53.1 for the heuristic; given both schema versions, a deterministic diff solves renames perfectly, yet on structural drift it collapses to 0–6.7% while LLMs reach 100%; and on column semantic transforms, six of nine systems execute 98–100% of the time while only two exceed 20% equivalence. To test transfer, we mine the histories of 40 open-source applications, identify 855 schema-verified renames across 34 of them, and build a 300-instance real-migration slice. There, seven current models execute 94–100% of the time but are correct on only 75–90%, and 93% of their failures are queries that run and return wrong rows; the real migration's rename list lifts them to 93–100%. Model rankings on the synthetic held-out-vocabulary and cross-domain suites predict real-slice rankings (Spearman ρ=0.70 and 0.80), and the semantic-transform gap persists in current models (46–87% equivalence at 100% execution).

1 INTRODUCTION

Schema evolution is unavoidable in production data systems (Curino et al., 2008; Roddick, 1995): administrators rename columns, split tables, or change the units in which a value is stored (Moon et al., 2008), and these edits propagate outward. Dashboards stop working, pipelines break, and reports can silently produce incorrect numbers (Rahm & Do, 2000; Qian et al., 2012). Repairing affected queries has traditionally been manual (Curino et al., 2009) and error-prone, because naive string replacement can alter meaning when aliases, nested scopes, or string literals are involved (Pavlo et al., 2017; Abiteboul et al., 1995).

Large language models (LLMs) are a natural candidate for this task (Chen et al., 2021; Li et al., 2022): they parse error messages, reason over schema structure, and emit valid SQL (Scholak et al., 2021; Wang et al., 2020). Evaluating them is the harder problem. SQL benchmarks target *generation* from natural language (Spider (Yu et al., 2018), WikiSQL (Zhong et al., 2017), EvoSchema (Zhang et al., 2025)) rather than *repair*, so they cannot express the constraint that matters in production: the repaired query must return the *same result* that the original returned on the *original* schema. Nor is "the query runs" an adequate substitute: a repair can execute successfully while returning incorrect rows (Chaudhuri & Vardi, 1993; Chu et al., 2018).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

DriftBench rests on three design principles. **Drift is guaranteed to matter**: queries are conditioned on the sampled drift and validated by execution; the original must run on the original schema and *fail* on the drifted one (Curino et al., 2009). **Correctness is measured by execution**: one plan migrates both schema and data, and repairs are scored by multiset comparison against the pre-drift result (Chaudhuri & Vardi, 1993). **Ground truth is reliable**: oracle repairs are derived by rewriting the query's abstract syntax tree (AST) with `sqlglot` (Tobiko Data, 2023) rather than regular expressions (Begoli et al., 2018).

What the benchmark reveals. (i) *Execution success systematically overstates repair quality*: under the default prompt, the exec–equiv gap appears for every LLM on every rename suite, 7.1–11.8 percentage points (pp) on the main benchmark and up to 35.7 pp on real migrations (53.1 pp for the fuzzy heuristic on S2), and widens under semantic drift, where six of nine systems execute 98–100% of the time yet only two exceed 20% equivalence. (ii) *The information regime, not the model, sets the difficulty*: given both schema versions, a schema-diff plus AST rewrite solves renames exactly, so **DriftBench** withholds the original schema; on the 292-instance suite S2, the best LLM then reaches 90.4%. (iii) *Learned and deterministic repair swap places with drift type*: on table splits, which require a `JOIN` that no mapping can express, deterministic rewriting scores 0% while three LLMs score 100%. (iv) *The conclusions transfer to real migrations*: on a slice built from 855 schema-verified renames found in 34 of the 40 open-source projects we mined, current models execute almost always but are correct on only 75–90% of instances, synthetic-suite rankings predict real-slice rankings, and the project's own rename list closes most of the gap (§7.6).

Contributions. (1) **A procedural generator** of paired original/drifted schemas, drift-conditioned SQL workloads, and migrated data linked by canonical migration plans, parameterized by drift category, query complexity, difficulty, rename vocabulary, and schema domain (§4). (2) **An execution-based harness** with namespace isolation, real error capture, and multiset comparison under an explicit *information regime* (§3–5). (3) **A study of 18,624 system evaluations** across seven LLMs, a schema-diff oracle, and a fuzzy heuristic, spanning three drift families, two domains, and two rename vocabularies, with bootstrap CIs and exact paired tests. (4) **A real-migration transfer slice** of 300 instances built from schema-verified renames in open-source histories, evaluated with 24 models (seven current models, 12 further models from within-family scaling series, and five re-run from the core study) in 33,213 further model evaluations that also cover execution feedback (§7.6).

2 Related Work

Schema evolution and query maintenance. Schema versioning has a long history (Roddick, 1995; 1992). PRISM (Curino et al., 2008; 2009) introduced schema modification operators with semantics-preserving migration of schema and data, and was extended to transaction-time databases by Moon et al. (2008); in practice, migration relies on scripts and manual coordination (Rahm & Do, 2000), which remain difficult under aliasing, nested scopes, or joins (Qian et al., 2012). That work migrates *databases*; **DriftBench** evaluates repairing the *queries* a completed migration broke, when the migration record is unavailable to the repair system.

Text-to-SQL and schema robustness. Spider (Yu et al., 2018) and WikiSQL (Zhong et al., 2017) drove advances in semantic parsing (Wang et al., 2020; Lin et al., 2020; Li et al., 2023), schema linking (Lei et al., 2020), constrained decoding (Scholak et al., 2021), and conversational (Yu et al., 2019b;a) and compositional (Suhr et al., 2020; Shaw et al., 2021) generalization. Most closely related, EvoSchema (Zhang et al., 2025) perturbs BIRD schemas with ten column- and table-level operations and finds table-level perturbations more damaging, a pattern echoed by our per-category repair results. The tasks differ (Table 8, Appendix A): EvoSchema keeps a question fixed and regenerates SQL against a gold query, whereas **DriftBench** has no question; its input is a failing query plus the database's error, and correctness is equality with what that query returned before the schema changed.

Program and SQL repair with LLMs. LLMs are strong general code repairers (Chen et al., 2021; Li et al., 2022; Le et al., 2022). SQL work targets debugging and error explanation (Cen et al., 2025; Guo et al., 2018), query recommendation (Zheng et al., 2023), and dialect translation (Lin et al., 2025); SQLFixAgent (Cen et al., 2025) and similar pipelines repair a text-to-SQL model's answer to a question against a gold query, which does not transfer to our setting. We therefore compare against

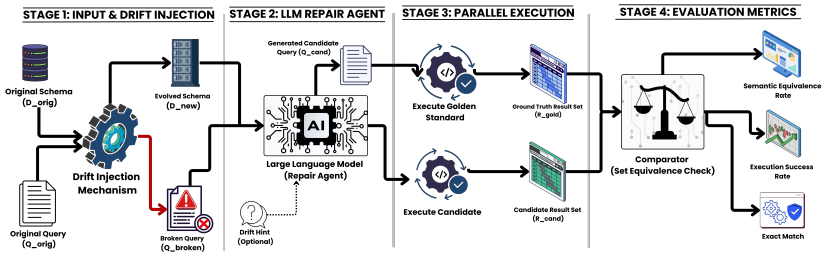


Figure 1: The **DriftBench** pipeline. Stage 1 injects controlled drift, yielding a drifted schema, a migrated database, and a broken query. Stage 2 prompts a repair system, which sees only the failing query, the captured error, and the drifted schema (R2); the drift hint appears only in the R3 ablation. Stage 3 executes the original query on the original namespace and the candidate on the drifted one. Stage 4 scores execution success and result-set equivalence.

the two applicable families: a deterministic schema-diff plus AST rewrite, the strongest method when both schema versions are known (§7.2), and fuzzy identifier matching.

Parsing, equivalence, and benchmarking. We use `sqlglot` (Tobiko Data, 2023) for AST manipulation (Calcite (Begoli et al., 2018) offers a full optimizer); rewriting has been studied for optimization (Chaudhuri & Gravano, 1996), view maintenance (Gupta et al., 1993), and provenance (Green et al., 2007). SQL equivalence is undecidable in general (Chu et al., 2018), so we compare result multisets on concrete instances. Related benchmarks cover TPC workloads (Meikel Poess and Raghu Nambiar , 2010), stress tests (Pavlo et al., 2017), metamorphic testing (Rigger & Su, 2020; Ba & Rigger, 2023), learned optimization (Marcus et al., 2021), and LLMs for database tasks (Nan et al., 2023).

3 Task Definition and Information Regimes

Let S be a schema, D a database over S, and Q a query. A drift set Δ transforms S into S' and, through a migration plan, D into a semantically equivalent D'. A repair system observes O and must output $\hat{Q}$ such that $[\![\hat{Q}]\!]_{D'} = [\![Q]\!]_D$, where $[\![\cdot]\!]$ denotes the result multiset. The difficulty of the task depends on the information contained in O. We distinguish **R1 (full history)**, $O = (Q, S, S', \Delta)$, where both schema versions are available and the rename mapping is therefore recoverable; **R2 (current schema only)**, $O = (Q, S', e)$, where e is the error raised when Q runs against S', so the mapping must be inferred; and **R3 (partial metadata)**, $O = (Q, S', e, h)$, where h is a natural-language description of the changes, e.g. from a migration tool. R2 is **DriftBench**'s default; R1 and R3 are evaluated as an upper bound (§7.2) and an ablation (§7.4), respectively.

R2 is the default for two reasons. First, R1 is scientifically closed: a schema-diff plus AST rewrite attains 100% equivalence on renames (§7.2), so a benchmark defined in R1 would measure parser coverage rather than repair ability. Second, R2 matches the setting in which breakage is typically discovered: analyst queries, notebooks, and dashboards are maintained separately from the migration that broke them, and an on-call engineer has a failing query and an error message. Reporting all three regimes separates failures caused by missing information from failures caused by missing capability.

4 Benchmark Design

Figure 1 shows the pipeline; this section covers how instances are built and §5 how they are scored.

Schema and data. The primary generator instantiates an e-commerce schema with four tables, `customers`, `products`, `orders`, and `order_items`, whose foreign keys support joins, correlated subqueries, and window functions with meaningful partitions. The schema is a *module*: a hospital domain (`patients`, `medications`, `visits`, `prescriptions`) with the same foreign-key pattern and its own rename vocabulary runs through the byte-identical pipeline (§7.5). Values are sampled deterministically from the instance seed; all records are synthetic.

Drift families. **DriftBench** implements three families spanning a spectrum from mapping-recoverable to structurally irreducible. *Renames* (the main benchmark) dominate day-to-day maintenance (Curino et al., 2008); each instance samples 1–4 operations bucketed into `single_column`, `multi_column` (2–3), `table_rename` (affecting `FROM`/`JOIN`), and `mixed` (table plus column). *Table split* partitions a table column-wise into two tables sharing a key, e.g. `customers` → `customer_profiles` + `customer_contacts`; a query touching both halves must synthesize a new `JOIN`, which no rename mapping can express. *Column semantic transform* renames a column *and* rescales its values, for example, `products.unit_price` (`DECIMAL`) → `price_cents` (`INTEGER`), $\times 100$, so a purely syntactic rewrite executes while silently corrupting every downstream aggregate.

Rename vocabulary. Targets come from explicit dictionaries. The default covers 11 column and 4 table identifiers with 31 and 11 targets, e.g. `customer_id` → {`cust_id`, `uid`, `client_id`}, `customers` → {`clients`, `customer_master`}, with mean old-to-new Levenshtein similarity 0.55. Because conventional targets could inflate results, a *held-out* dictionary of deliberately unconventional targets (similarity 0.35; e.g. `customer_id` → `party_key`, `orders` → `booking_fact`) is evaluated on matched suites (§7.5).

Canonical migration plans. Consistency is non-trivial when operations interact: renaming a table before a column inside it breaks any mapping keyed by *current* identifiers. **DriftBench** keys its plan by *original* identifiers, `orig_table.orig_column` → `new_column` and `orig_table` → `new_table`, making it order-independent. The same plan migrates DDL and rows, so the drifted database contains semantically identical data under new naming. This property makes result-set comparison a valid correctness test.

Drift-conditioned generation. The generator first draws Δ, then constructs a query that *must* reference at least one drifted identifier. Six template families of increasing complexity are used: simple selects, aggregations, 2–3 table joins, correlated and uncorrelated subqueries, CTEs (Finkelstein, 1982), and window functions (Zemke, 2012), balanced across instances ($\approx$333 per family for 2,000). Conditioning is necessary but not sufficient, so it is backed by execution validation (§5.1).

Oracle repairs by AST rewriting. String replacement is unreliable: an identifier may appear as an alias, inside a literal, or under different qualifiers in different clauses. **DriftBench** applies Δ to the query's AST with `sqlglot` (Tobiko Data, 2023); when parsing fails ($\approx$2% of queries), a conservative regex fallback skips literals and respects quoted identifiers, and those cases are logged. Structural drift uses transformation-specific rewriters (join synthesis for splits and an inverse-scaling expression for transforms), validated identically: the oracle must reproduce the pre-drift result set, or the instance is rejected.

5 Evaluation Protocol

5.1 Isolated Namespaces and Instance Validation

A subtle failure mode is an original query "accidentally" succeeding against drifted objects when old and new names coexist. **DriftBench** prevents this by creating two isolated namespaces per instance: `orig_{id}` and `drift_{id}`; these are separate schemas within one database on PostgreSQL and separate in-memory databases on SQLite.

Each candidate must then pass three execution checks: (i) the original query runs on the original namespace; (ii) it *fails* on the drifted namespace, confirming that the drift actually affects the query; and (iii) the oracle repair runs on the drifted namespace and returns a result multiset equal to (i). Candidates that fail any check are discarded and regenerated, making drift impact a guarantee rather than an intention. On SQLite, double-quoted unknown identifiers are interpreted as string literals, so column-only drift sometimes fails to break a single-table query; the validator rejects these candidates, which is why balanced SQLite suites fill about 58 of 100 requested slots per seed while PostgreSQL fills 100.

5.2 Prompting Conditions

For each validated instance, we capture the real database error by executing the original query against the drifted namespace. The **main prompt** (R2) contains the failing query, the resulting error with SQLSTATE where available, and the drifted schema; the model must infer what changed, matching the situation a DBA faces when a dashboard breaks. The **hint prompt** (R3) adds an explicit description of the changes, e.g. "`customers.customer_id` was renamed to `clients.uid`". No prompt contains the original schema, the migration plan, or the oracle repair.

5.3 Metrics

With Q the original query, $Q^\star$ the oracle repair, and $\hat{Q}$ the candidate, we report **execution success (Exec)**, $\mathbf{1}[\hat{Q}$ runs without error on the drifted namespace$]$; **semantic equivalence (Equiv)**, multiset equality between $[\![Q]\!]$ on the original namespace and $[\![\hat{Q}]\!]$ on the drifted one (Chaudhuri & Vardi, 1993), with rows normalized (case-insensitive, whitespace-trimmed, float tolerance 10^{-6}) and positional equality also required for ordered queries; **oracle score**, the multiset Jaccard similarity between $[\![Q^\star]\!]$ and $[\![\hat{Q}]\!]$; and **edit distance**, normalized Levenshtein distance (Lcvenshtcin, 1966) between $\hat{Q}$ and $Q^\star$. Equiv is the primary metric, and Exec is reported alongside it because the gap between the two is itself a finding.

6 Experimental Setup

We ask six questions: how accurately LLMs repair rename-drifted queries and how far execution diverges from semantic correctness (**RQ1**, §7.1); how frontier models compare with open models and with the R1 deterministic upper bound (**RQ2**, §7.2); whether the framework and the exec–equiv gap extend to drift that no rename mapping can express (**RQ3**, §7.3); how much explicit schema-change metadata helps and for which models (**RQ4**, §7.4); whether the conclusions hold under unconventional rename vocabularies and a different schema domain (**RQ5**, §7.5); and whether they transfer to real migrations and current models (**RQ6**, §7.6).

6.1 Evaluation Suites

We report eight suites (Table 9); S1–S7 total 18,624 system evaluations: **S1** (main, renames, N=2,000, seeds 42–46, PostgreSQL), **S2** (frontier models and R1 oracle, renames, N=292), **S3** (table split, N=100), **S4** (column transform, N=104), **S5** (paired prompt ablation, N=290), **S6** (default vs. held-out dictionary, N=116 per condition), **S7** (hospital domain, N=116), and **S8** (real-migration slice, N=300 from 34 of 40 mined projects, §7.6). S2–S8 use SQLite, and every results table restates its suite, N, seeds, and harness. **S1 is the headline benchmark**: "best result on the full benchmark" always means S1 on PostgreSQL (Stonebraker et al., 2018). **Absolute rates are consistently lower on SQLite** for the same models because its weaker type affinity and quoted-identifier semantics admit different failure patterns, and the validated instances differ. Absolute numbers must therefore be compared *within* a suite, which is how every claim below is framed.

6.2 Systems and Statistics

We evaluate four open LLMs served through a local API endpoint (**Gemma3-27B**, **Phi-4-14B**, **Mistral-7B**, **DeepSeek-R1-7B**) and three frontier API models (**GPT-4o-mini**, **Gemini-2.5-Flash-Lite**, **Claude-3-Haiku**), all with deterministic decoding (temperature 0, top-p 1.0, 2,048-token budget) (Touvron et al., 2023). Two non-learned systems anchor the range: a **fuzzy-string heuristic** that maps missing identifiers to the closest schema element by edit distance (Navarro, 2001), and a **deterministic schema-diff oracle** that sees both schema versions and applies the AST rewrite under the mapping they imply. For renames, that mapping is exactly recoverable, so the oracle upper-bounds any schema-diff procedure; it exists only in R1 and serves as a ceiling, not a competitor. For structural drift, we add a *smart deterministic* variant that guesses from name similarity where a split column landed. Each evaluation runs in a fresh namespace; SQL extraction handles Markdown fences, prose, and multi-statement outputs; every evaluation is recorded with the captured database error and SHA-1 hashes of its prompt and response (Gundersen & Kjensmo, 2018). Rates carry 95%

Table 1: Suite S1 (2,000 instances, seeds 42–46, PostgreSQL), with 95% bootstrap CIs. Gap is Exec−Equiv.

Model	Exec (%)	Equiv (%)	Gap	Orac.
Phi-4-14B	99.1 [98.6, 99.5]	**92.0** [90.8, 93.2]	7.1	.925
Gemma3-27B	98.7 [98.1, 99.1]	90.5 [89.1, 91.6]	8.2	.911
Mistral-7B	81.5 [79.7, 83.2]	69.7 [67.7, 71.7]	11.8	.707
DeepSeek-7B	60.3 [58.2, 62.5]	52.2 [50.0, 54.3]	8.1	.526
Fuzzy base.	22.9 [21.0, 24.7]	13.7 [12.3, 15.3]	9.2	.141

Table 2: Suite S1: semantic equivalence (%) by drift category. Per-template results are in Table 10.

Model	Single	Multi	Table	Mixed
Phi-4-14B	90.4	92.8	**98.6**	86.2
Gemma3-27B	90.4	92.2	96.2	83.0
Mistral-7B	66.6	69.2	85.6	57.4
DeepSeek-7B	56.4	60.4	57.4	34.6
Fuzzy base.	16.2	16.8	21.8	0.0

bootstrap confidence intervals (1,000 resamples; 2,000 for paired differences). In §7.6, all learned systems are served through one OpenRouter endpoint (Appendix B).

7 RESULTS

7.1 MAIN BENCHMARK (RQ1)

The best result on the full benchmark is 92.0% semantic equivalence (Phi-4-14B), followed by Gemma3-27B at 90.5%; both execute above 98% (Table 1). Mistral-7B reaches 69.7% and DeepSeek-R1-7B 52.2%, with the latter limited chiefly by execution failures (60.3%); the fuzzy heuristic executes 22.9% of the time and is correct 13.7% of the time. Every system executes more often than it is correct, by 7.1–11.8 pp. Execution alone would credit Phi-4 with 99.1% and obscure a failure class in which a repair returns a plausible but incorrect result set.

Where repair fails. **Mixed drift is consistently hardest** (Table 2): DeepSeek falls from 56–60% on single-operation categories to 34.6%, and the heuristic falls to 0.0%. When `customers` becomes `clients` and `customer_id` becomes `uid`, the join `customers c JOIN orders o ON c.customer_id = o.customer_id` requires three coordinated edits, while only one side of the predicate is renamed; updating only a subset yields an error or, worse, a query that executes on the wrong join key. **Table renames are easiest** for every system except DeepSeek (Phi-4 98.6%): a table name occurs once and is named outright by the error. Difficulty matters far less than category or structure; the top models vary by at most 2.4 pp across one to four drift operations. Essentially all execution failures carry SQLSTATE 42P01 or 42703, i.e. unresolved drift rather than syntax damage (Appendix A).

7.2 FRONTIER MODELS AND THE DETERMINISTIC UPPER BOUND (RQ2)

Table 3 places nine systems on three drift regimes. Given both schema versions (R1), the schema-diff oracle solves rename drift completely: *on renames, with full migration history, LLMs are not needed*, which is why the default benchmark withholds that history. Under R2 the same instances are markedly harder. Gemini-2.5-Flash-Lite leads at 90.4%, 9.6 pp below the oracle, ahead of GPT-4o-mini (86.6%), Claude-3-Haiku (85.3%), and the open models (81.8%, 81.2%). That residual measures the information R2 removes; supplying the mapping closes much of it (§7.4, §7.6). The exec–equiv gap persists across model families (5.9–19.9 pp) and reaches 53.1 pp for the fuzzy heuristic, which executes 59.9% of the time and is right 6.8% of the time: a system that renames identifiers to whatever looks closest is optimized for the execution metric and indifferent to the semantic one.

7.3 BEYOND RENAMES: STRUCTURAL DRIFT (RQ3)

We generate suites for the two structural operations of §4 through the same pipeline, with no methodological change (right of Table 3).

Table split: learned repair wins outright. The R1 oracle scores 0.0%: no rename mapping can turn a single-table scan into a two-table join. The split-aware deterministic variant executes 80.0% of the time and is *never* correct, while GPT-4o-mini, Gemini-2.5-Flash-Lite, and Phi-4-14B reach

Table 3: The same nine systems across three drift regimes on the SQLite harness: S2 (renames, N=292), S3 (table split, N=100), S4 (column transform, N=104), all seeds 42–43. The schema-diff oracle is the only system in regime R1; all others see the drifted schema only. Determ. rewrite is the mapping-based rewriter (the split-aware variant on S3).

	S2 Renames		S3 Split		S4 Transf.	
System	Exec	Equiv	Exec	Equiv	Exec	Equiv
Oracle (R1)	100.0	**100.0**	0.0	0.0	0.0	0.0
Gemini-2.5-FL	99.3	**90.4**	100.0	**100.0**	100.0	15.4
GPT-4o-mini	96.2	86.6	100.0	**100.0**	100.0	59.6
Claude-3-Haiku	97.3	85.3	83.0	33.0	100.0	**86.5**
Gemma3-27B	87.7	81.8	98.0	91.0	68.3	10.6
Phi-4-14B	89.4	81.2	100.0	**100.0**	99.0	12.5
Mistral-7B	82.9	63.0	38.0	36.0	98.1	7.7
DeepSeek-7B	56.2	46.9	43.0	21.0	23.1	1.9
Determ. rewrite	–	–	80.0	0.0	100.0	6.7
Fuzzy baseline	59.9	6.8	80.0	0.0	0.0	0.0

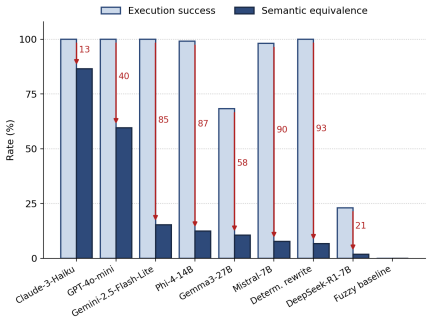


Figure 2: Suite S4 (104 column-transform instances). Almost every system executes far more often than it is correct; arrows mark the exec–equiv gap in points. The mapping-based deterministic rewriter executes 100% of the time and is correct 6.7% of the time.

Table 4: Suite S5: paired prompt ablation, 290 instances (seeds 42–46, SQLite) each evaluated under the R2 and R3 prompts. CIs are bootstrap CIs on the paired difference; p from an exact McNemar test.

Model	R2	R3	Δ (pp)	95% CI	p
Mistral-7B	69.3	86.9	**+17.6**	[+11.4,+23.4]	$3.3{\cdot}10^{-8}$
Gemma3-27B	79.3	89.7	**+10.3**	[+5.2,+15.5]	$2.3{\cdot}10^{-4}$
Phi-4-14B	89.3	87.2	−2.1	[−6.6,+2.1]	0.44
DeepSeek-7B	33.4	27.6	−5.9	[−13.1,+1.4]	0.15

Table 5: Robustness (Equiv %, SQLite). Suite S6: matched 116-instance suites under the default vs. held-out dictionary (seeds 47–48). Suite S7: hospital domain (116 instances, seeds 49–50), compared against the S6 default column.

System	Default dict.	Held-out dict. (Δ)	Hospital domain (Δ)
Phi-4-14B	85.3	86.2 (+0.9)	80.2 (−5.1)
Gemma3-27B	81.0	68.1 (−12.9)	71.6 (−9.4)
Mistral-7B	68.1	59.5 (−8.6)	57.8 (−10.3)
DeepSeek-7B	39.7	24.1 (−15.5)	32.8 (−6.9)
Fuzzy base.	10.3	**0.0** (−10.3)	2.6 (−7.7)

100.0% and Gemma3-27B 91.0% by reading the drifted DDL and writing the join. The ordering between deterministic and learned repair inverts completely relative to renames.

Column transform: the gap becomes the whole story. A rename *plus* a unit change makes a syntactically perfect repair wrong by a factor of 100. Six of nine systems execute 98–100% of the time, yet only two exceed 20% equivalence (Figure 2): Claude-3-Haiku (86.5%), which notices the `price_cents` naming and inserts a `/100.0` correction, and GPT-4o-mini (59.6%). The mapping-based rewriter executes 100.0% of the time but is correct only 6.7% of the time; each execution returns the right rows under the right column names with monetary totals inflated a hundredfold.

7.4 Does Schema-Change Metadata Help? (RQ4)

Table 4 evaluates 290 instances under both prompts. The effect of R3 is large and *model-dependent*: Mistral-7B gains +17.6 pp ($p = 3.3 \times 10^{-8}$) and Gemma3-27B +10.3 pp ($p = 2.3 \times 10^{-4}$), while Phi-4-14B and DeepSeek-R1-7B show no measurable benefit (−2.1 and −5.9 pp, n.s.). Mid-tier models rewrite reliably but mis-infer *what* to rewrite, which the hint resolves; Phi-4 already infers the mapping in 89% of cases, while DeepSeek's failures are dominated by malformed output.

7.5 Robustness: Rename Vocabulary and Schema Domain (RQ5)

Two properties of the generator could inflate results: conventional rename targets and evaluation in a single domain (Table 5).

Table 6: Suite S8 (300 real-migration instances, 34 projects). R2 Exec/Equiv (%); R3 adds the real rename list. Blocks: current models, re-served paper models, heuristics.

System	Exec	Equiv	Gap	R3
Schema-diff oracle (R1)	100.0	100.0	0.0	–
GLM-5.3-Flash	100.0	**89.7**	10.3	100.0
DeepSeek-V4.1-Flash	99.3	88.3	11.0	100.0
GPT-6-Luna	99.7	88.0	11.7	99.7
Claude-Haiku-4.5	99.7	83.3	16.3	96.3
Gemini-3.1-FL	99.7	83.3	16.3	98.7
gpt-oss-120b	100.0	83.3	16.7	98.0
Ministral-8B	94.0	75.0	19.0	93.0
Claude-3-Haiku	92.3	76.3	16.0	94.7
Gemini-2.5-FL	95.3	76.0	19.3	95.7
GPT-4o-mini	98.0	74.3	23.7	97.7
Gemma3-27B	91.0	73.3	17.7	97.0
Phi-4-14B	87.0	51.3	35.7	79.0
Schema-diff heuristic	61.7	45.7	16.0	–
Fuzzy baseline	15.7	13.7	2.0	–

Table 7: Top: Spearman correlation between the 12 models' Equiv on S8 and on each synthetic suite (permutation p). Bottom: mean Equiv (%) of six models, first→final turn, over up to three feedback turns: the database error; the error plus a preview of the returned rows and a request to check their meaning and units; the same preview with a neutral request.

Synthetic suite	ρ	p
S6 default dict.	0.39	0.217
S6 held-out dict.	0.70	0.015
S7 hospital	0.80	0.002
S3 table split	0.62	0.034
S4 col. transform	0.65	0.025

Suite	Error	Preview	Neutral
S8 real	80.2→80.8	81.1→79.7	80.5→80.6
S4 transform	52.7→52.7	50.3→87.7	52.7→72.1
S6 held-out	86.6→87.9	87.6→88.2	–
S3 split	99.0→99.0	99.0→99.2	–

Unconventional renames. Under the held-out dictionary, the fuzzy heuristic falls from 10.3% to **0.0%** (95% CI on Δ $[-16.4, -5.2]$), confirming that these targets remove the surface cues on which a matching heuristic depends. The strongest model is unaffected (Phi-4 $+0.9$, CI $[-8.6, +10.3]$), while weaker models degrade (Gemma3 -12.9 and DeepSeek -15.5, both significant; Mistral -8.6, n.s.); the ranking is preserved and the LLM–heuristic separation widens.

A second domain. Swapping only the schema module for the hospital domain, with generation, validation, and evaluation code byte-identical, costs every system 5–10 pp and preserves the ranking exactly. The shift also applies to the non-learned baseline, indicating that it reflects the domain's more abbreviated conventions (`mrn`, `dob`) rather than the models.

7.6 Transfer to Real Migrations and Current Models (RQ6)

A real-migration slice. Generation certifies every instance but cannot establish that the conclusions hold for real drift. We mine the full histories of 40 open-source Ruby-on-Rails applications that commit a schema snapshot. Of 1,309 rename operations in the forward direction of 18,039 migrations, we retain the 855 (508 commits, 34 projects) confirmed by the snapshots: before the migrating commit, the old identifier exists and the new one does not, and the reverse holds at that commit or within the next 15 schema commits. Thus, a parsing error can drop a rename but never admit one. Each verified commit yields the pre-migration schema restricted to the renamed tables, foreign-key neighbors, and distractor tables (median 8 tables, 88 columns); only the verified renames are applied, rows are synthesized from column types, and templated queries over the renamed identifiers must pass the validation of §5.1 and return rows. We sample 300 instances covering all 34 projects (at most ten per project; commits 2008–2026). Real renames are lexically *closer* than ours (mean similarity 0.61 vs. 0.55 and 0.35), so S8 is not harder because of its vocabulary (Appendix B).

The gap widens on real schemas. Seven current models execute 94.0–100% of real instances but are correct on only 75.0–89.7% (Table 6), and 93% of their failures are queries that execute and return wrong rows. Averaged over the twelve learned models, Equiv is 78.5% on S8 versus 86.9% on the synthetic rename suites. The failures are substitutions into neighboring structure: `impound_record_updates.resolved`, renamed to `processed`, is replaced by `resolved_at` from the joined table, and `user_last_seen_at` becomes its sibling `agent_last_seen_at` rather than `contact_last_seen_at`; 30% of silent failures still name the old identifier, which resolves against another table that retains it. Mixed renames are again hardest (49.2% vs. 90.4% for table renames).

Metadata closes most of the gap. The real migration's rename list (R3) raises every current model to 93.0–100% ($p<10^{-9}$ each, exact McNemar) and the re-run paper models to 79.0–97.7%. On S5, the hint helped only mid-tier models; on real schemas, it helps all of them because the missing information is the mapping itself. A heuristic that diffs both schema versions but matches columns by name reaches only 45.7%.

Synthetic rankings predict real rankings. Equiv on the held-out-vocabulary and hospital suites correlates with Equiv on S8 at ρ=0.70 and 0.80 (Table 7), and at ρ=0.88 for the held-out suite once the within-family scaling models are added (23 models, Appendix B); the default-vocabulary suite, on which current models cluster at 86–96%, does not discriminate (ρ=0.39). The harder synthetic variants are the informative proxies for real drift.

Current models, structural drift, and feedback. All seven current models solve the table split (89–100%), and none escapes the semantic-transform trap: each executes 100% of S4 with 46.2–86.5% Equiv, and Claude-Haiku-4.5 is 41 pp *below* Claude-3-Haiku there while outperforming it on every rename suite (Appendix B). Returning the database error for up to three turns raises execution to 99–100% but Equiv by at most 1.3 pp on average. A preview of the returned rows helps when the values expose the error: on S4, it raises Equiv from 52.7 to 72.1% when the model is asked only whether the query preserves the original intent, and from 50.3 to 87.7% when the request names "the meaning and units" of the values. However, that wording also leads GPT-4o-mini and GPT-6-Luna to revise correct S8 repairs into incorrect ones (-7.7 and -5.7 pp, $p<0.001$), which the neutral request does not (80.5→80.6%). Without the pre-drift result, feedback converges on queries that execute and appear plausible (scaling: Appendix B).

8 Discussion and Conclusion

Execution success is the wrong headline metric. Under the default prompt, the exec–equiv gap appears for every LLM on every rename and semantic-transform suite: 7.1–11.8 pp on S1, 5.9–19.9 pp on S2, and 10.3–35.7 pp on real migrations (Table 6), with 53.1 pp for the fuzzy heuristic on S2 and 93.3 pp for a deterministic rewriter on semantic transforms; only on table splits are the models that write the join exactly right. Execution feedback likewise makes incorrect repairs execute rather than making them correct. "The repaired query runs" is a diagnostic, not an accuracy measure; a repair pipeline requires a result-level check against a pre-migration snapshot.

Difficulty is set by the information regime. Adding drift operations barely changes strong models, whereas withholding the original schema costs the best LLM about 10 points relative to the oracle, a change description returns 17.6 points to a mid-tier model, and a real migration's rename list lifts every current model above 93%. The highest-leverage intervention is therefore a provenance channel from migrations to downstream consumers, not a better model.

Learned and deterministic repair are complements, and size decides neither. Deterministic rewriting is exact on renames given full history and scores 0% on table splits, where three models reach 100%: resolve deterministically first, then fall back to a model when the drift cannot be expressed as a mapping. Scale and recency do not determine the strongest model: Phi-4-14B beats Gemma3-27B on S1, and Claude-3-Haiku leads on semantic transforms, 41 pp ahead of its successor.

Limitations. The main suites are procedurally generated so that drift impact and oracle existence are certified for every instance. S8 contains real schemas and real renames but uses templated queries and synthesized rows, because the projects' own code contains almost no self-contained queries over the renamed identifiers. It also covers only Rails applications. Drift coverage spans three families, equivalence is determined on concrete data, and absolute scores depend on vocabulary, domain, harness, and serving stack (Appendix E).

Conclusion. **DriftBench** makes drift impact a guarantee, correctness a property of the pre-drift result set, and the information regime an experimental variable. Across synthetic and real migrations, it shows where LLM repair succeeds, where deterministic tooling is stronger, and where neither is safe without result-level validation.

Ethics Statement

All data rows in **DriftBench** are synthetic: customer-like, patient-like, and product records are generated from seeded pseudo-random samplers and contain no personal data, and no human subjects were involved. The real-migration slice adds public table and column names, commit identifiers, and rename operations from open-source repositories; its rows are synthesized in the same way, and nothing beyond provenance links to the public histories is redistributed. The main risk we see is misuse in automation. An LLM-proposed repair should not be deployed without result-level validation, and our results quantify why: systems that execute 98–100% of the time can be semantically wrong in the great majority of cases, and on a unit change a mapping-based rewriter executed on every instance while preserving semantics on 6.7% of them. We recommend human review and result-set auditing before any production use. The artifact is licensed Apache-2.0 for code and CC-BY-4.0 for data, and contains no API keys, endpoints, or credentials.

Reproducibility Statement

An anonymized artifact containing the benchmark instances, generation and evaluation code, experiment configurations, and documentation is provided as supplementary material; a public repository will be made available upon acceptance. Section 3 defines the task and information regimes, Sections 4–6 specify generation, validation, prompting, metrics, and statistics, and Table 9 (Appendix A) gives the drift family, size, seeds, and harness of every suite behind a reported number. Appendix F lists the artifact contents. Generation and decoding are deterministic, so every suite can be rebuilt exactly from its seeds. Two verification paths in the supplementary material need no model endpoint and no network, regenerating a structural suite and confirming its size, and re-deriving the column-transform result with the deterministic systems, and an offline test suite re-derives the reported suite sizes and rates from the shipped result files. The real-migration slice is rebuilt by scripts that clone the listed repositories, verify each rename against the schema history, and validate instances; the additional experiments record tokens, cost, and serving provider per call, and a re-scoring script reproduces both extraction protocols (Appendix B).

AI Use Statement

Large language models are the *object of study* in this paper: the models in Sections 6 and 7.6 are evaluated as repair systems, and their outputs constitute the experimental data. Separately, we used a generative AI assistant as a writing and engineering aid. It assisted with editing and restructuring the manuscript, cross-checking reported numbers against the underlying evaluation records, and implementing and refactoring parts of the released code, tests, documentation, and artifact. We did not use generative AI to generate or alter experimental results: all repairs were produced by the evaluated models and baselines, and all scores were computed by the evaluation harness. The supplementary material includes the per-evaluation records for S2–S8 and a sample of S1; the test suite re-derives the S2–S7 rates from the shipped files, and an included script regenerates the S8 and current-model tables. The research questions, experimental design, and interpretation of results are our own. We reviewed and verified all AI-assisted text and code and take full responsibility for the final text, claims, results, and artifacts.

References

Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

Jinsheng Ba and Manuel Rigger. Testing database engines via query plan guidance. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 2060–2071. IEEE, 2023.

Edmon Begoli, Jesús Camacho-Rodríguez, Julian Hyde, Michael J Mior, and Daniel Lemire. Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 221–230, 2018.

Jipeng Cen, Jiaxin Liu, Zhixu Li, and Jingjing Wang. Sqlfixagent: Towards semantic-accurate text-to-sql parsing via consistency-enhanced multi-agent collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 49–57, 2025.

Surajit Chaudhuri and Luis Gravano. Optimizing queries over multimedia repositories. In *Proceedings of the 1996 ACM SIGMOD international conference on management of data*, pp. 91–102, 1996.

Surajit Chaudhuri and Moshe Y Vardi. Optimization of real conjunctive queries. In *Proceedings of the twelfth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. 59–70, 1993.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Shumo Chu, Brendan Murphy, Jared Roesch, Alvin Cheung, and Dan Suciu. Axiomatic foundations and algorithms for deciding semantic equivalences of sql queries. *Proc. VLDB Endow.*, 11(11): 1482–1495, July 2018. ISSN 2150-8097. doi: 10.14778/3236187.3236200. URL https://doi.org/10.14778/3236187.3236200.

Carlo A Curino, Hyun J Moon, and Carlo Zaniolo. Graceful database schema evolution: The PRISM workbench. *Proceedings of the VLDB Endowment*, 1(1):761–772, 2008.

Carlo A Curino, Hyun J Moon, MyungWon Ham, and Carlo Zaniolo. The PRISM Workbench: Database schema evolution without tears. In *2009 IEEE 25th International Conference on Data Engineering*, pp. 1523–1526. IEEE, 2009.

Sheldon Finkelstein. Common expression analysis in database applications. In *Proceedings of the 1982 ACM SIGMOD international conference on Management of data*, pp. 235–245, 1982.

Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 31–40, 2007.

Odd Erik Gundersen and Sigbjørn Kjensmo. State of the art: Reproducibility in artificial intelligence. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Yun Guo, Nan Li, Jeff Offutt, and Amihai Motro. Automatically repairing sql faults. In *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 500–511. IEEE, 2018.

Ashish Gupta, Inderpal Singh Mumick, and Venkatramanan Siva Subrahmanian. Maintaining views incrementally. *ACM SIGMOD Record*, 22(2):157–166, 1993.

VI Lcvenshtcin. Binary coors capable or 'correcting deletions, insertions, and reversals. In *Soviet physics-doklady*, volume 10, 1966.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. Re-examining the role of schema linking in text-to-sql. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6943–6954, 2020.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 13067–13075, 2023.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Li Lin, Hongqiao Chen, Qinglin Zhu, Liehang Chen, Linlong Tang, and Rongxin Wu. Dlbench: A comprehensive benchmark for sql translation with large language models. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 854–866. IEEE, 2025.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *arXiv preprint arXiv:2012.12627*, 2020.

Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. Bao: Making learned query optimization practical. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 1275–1288, 2021.

Meikel Poess and Raghu Nambiar . TPC Benchmark H standard specification. *TPC*, 2010.

Hyun J Moon, Carlo A Curino, Alin Deutsch, Chien-Yi Hou, and Carlo Zaniolo. Managing and querying transaction-time databases under schema evolution. *Proceedings of the VLDB Endowment*, 1(1):882–895, 2008.

Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. Enhancing few-shot text-to-sql capabilities of large language models: A study on prompt design strategies. *arXiv preprint arXiv:2305.12586*, 2023.

Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C Mowry, Matthew Perron, Ian Quah, et al. Self-driving database management systems. In *CIDR*, volume 4, pp. 1, 2017.

Li Qian, Michael J Cafarella, and HV Jagadish. Sample-driven schema mapping. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 73–84, 2012.

Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

Manuel Rigger and Zhendong Su. Detecting optimization bugs in database engines via non-optimizing reference engine construction. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1140–1152, 2020.

John F. Roddick. Schema evolution in database systems - an annotated bibliography. *SIGMOD record*, 21(4):35–40, 1992.

John F Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1995.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*, 2021.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? 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)*, pp. 922–938, 2021.

Michael Stonebraker, Lawrence A Rowe, and Michael Hirohama. The implementation of POSTGRES. In *Making Databases Work: the Pragmatic Wisdom of Michael Stonebraker*, pp. 519–559. 2018.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8372–8388, 2020.

Tobiko Data. Sqlglot: A python sql parser and transpiler. `https://github.com/tobymao/sqlglot`, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 7567–7578, 2020.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. 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)*, pp. 1962–1979, 2019a.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al. Sparc: Cross-domain semantic parsing in context. *arXiv preprint arXiv:1906.02285*, 2019b.

Fred Zemke. What's new in SQL: 2011. *ACM SIGMOD Record*, 41(1):67–73, 2012.

Tianshu Zhang, Kun Qian, Siddhartha Sahai, Yuan Tian, Shaddy Garg, Huan Sun, and Yunyao Li. EvoSchema: Towards text-to-SQL robustness against schema evolution. *Proceedings of the VLDB Endowment*, 18(10):3655–3668, 2025.

Bolong Zheng, Lei Bi, Ruijie Xi, Lu Chen, Yunjun Gao, Xiaofang Zhou, and Christian S Jensen. Rhb-net: A relation-aware historical bridging network for text2sql auto-completion. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1458–1467, 2023.

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

A Extended Results

Table 8: Positioning against the closest benchmarks. **DriftBench** is the only one whose task is repair of an *existing* query and whose correctness criterion is equality with the pre-drift result set on migrated data.

Benchmark	Task	Input	Schema source	Correctness criterion	Drift impact
Spider / WikiSQL	NL → SQL	question, schema	curated, static	match to gold SQL	n/a
EvoSchema	NL → SQL	question, perturbed schema	BIRD, 10 perturbation types	exec. accuracy, table/col. F1	not enforced
DriftBench (ours)	broken SQL → repaired SQL	failing query, DB error, drifted schema	procedural, 2 domains	multiset equality with pre-drift result	enforced by execution

Table 9: Evaluation suites; every results table names its suite. Counts are post-validation, and S1 is balanced across drift category, query complexity, and difficulty.

Suite	Drift	N	Seeds	Harness
S1 Main	renames	2,000	42–46	PostgreSQL
S2 Frontier	renames	292	42–43	SQLite
S3 Split	table split	100	42–43	SQLite
S4 Transform	col. transf.	104	42–43	SQLite
S5 Ablation	renames	290†	42–46	SQLite
S6 Dictionary	renames	116×2	47–48	SQLite
S7 Hospital	renames	116	49–50	SQLite
S8 Real migr.	real renames	300	34 projects	SQLite

†paired: evaluated under both prompts.

A.1 Detailed Results on Suite S1

Table 10: Suite S1: semantic equivalence (%) by query template. Agg = aggregation, Sub = subquery, Win = window.

Model	Simple	Agg	Join	Sub	CTE	Win
Phi-4-14B	93.2	92.9	92.1	**95.9**	90.9	86.6
Gemma3-27B	91.8	85.9	96.5	92.1	89.1	87.2
Mistral-7B	78.5	56.2	74.4	74.1	77.8	56.9
DeepSeek-7B	40.9	51.5	48.8	58.8	63.8	50.0
Fuzzy base.	15.9	15.6	4.1	16.8	15.6	14.4

By query structure (Table 10), aggregations and window functions are the main failure modes for weaker models: Mistral drops from 74–78% on joins and CTEs to 56% on both. These templates require the same identifier to be updated consistently across structurally distinct positions, including aggregate arguments, `GROUP BY` keys, and `PARTITION BY` keys, and one missed occurrence is fatal. The heuristic fails almost completely on joins (4.1%), where table-qualified identifiers defeat surface matching. Difficulty matters far less than category or structure: Phi-4 and Gemma3 vary by at most 2.4 points from one to four drift operations, while Mistral degrades from 73.0% to 67.1%. Essentially all execution failures carry SQLSTATE 42P01 or 42703, precisely the errors caused by drift, indicating unresolved drift rather than incidental syntax damage (Table 13).

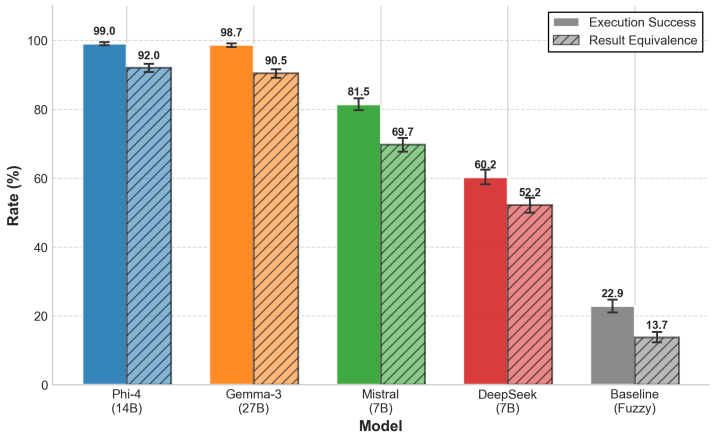


Figure 3: Suite S1: execution and semantic equivalence rates with 95% bootstrap confidence intervals.

Table 11 shows a clear stratification in robustness. Phi-4-14B and Gemma3-27B remain above 89.8% at every difficulty level, whereas Mistral-7B loses 5.9 pp and DeepSeek-R1-7B 2.4 pp between one and four drift operations. Difficulty is therefore a much weaker driver of failure than drift *category*

Table 11: Suite S1: semantic equivalence (%) by difficulty level (number of drift operations).

Model	Diff 1	Diff 2	Diff 3	Diff 4
Phi-4-14B	93.0	92.9	90.6	91.3
Gemma3-27B	90.7	90.8	89.8	90.4
Mistral-7B	73.0	69.6	68.5	67.1
DeepSeek-R1-7B	53.4	53.5	50.6	51.0
Fuzzy baseline	10.7	14.2	15.2	15.2

(Table 2), consistent with our interpretation that coordination across query scopes, rather than the raw number of changes, is what breaks weaker models.

Table 12: Suite S1: execution rate (%) by drift category.

Model	Single	Multi	Table	Mixed
Phi-4-14B	98.6	99.6	99.2	98.8
Gemma3-27B	98.4	98.8	99.4	98.0
Mistral-7B	80.6	82.6	91.2	71.4
DeepSeek-R1-7B	66.4	68.0	63.4	43.2
Fuzzy baseline	29.2	32.2	30.2	0.0

Table 12 gives the corresponding execution rates. DeepSeek-R1-7B drops to 43.2% execution on mixed drift and the fuzzy heuristic to 0.0%, mirroring the equivalence pattern: compound drift is where both non-reasoning and low-capability systems fail outright rather than subtly.

Table 13: Suite S1: dominant SQLSTATE codes among failed executions (counts over 2,000 instances).

System	SQLSTATE	Count
Fuzzy baseline	42P01 (undef. table)	849
Fuzzy baseline	42703 (undef. column)	693
DeepSeek-R1-7B	42P01	467
DeepSeek-R1-7B	42703	328
Mistral-7B	42P01	187
Mistral-7B	42703	184
Gemma3-27B	42703	14
Gemma3-27B	42P01	13
Phi-4-14B	42P01	10
Phi-4-14B	42703	9

Table 13 lists the SQLSTATE codes among failed executions. Essentially all are 42P01 (undefined table) and 42703 (undefined column), i.e. unresolved drift rather than incidental syntax damage. Phi-4-14B accumulates 19 such failures and Gemma3-27B 27 across 2,000 instances each. Figure 5 shows that oracle-score distributions are bimodal and concentrated near 1.0 for Phi-4 and Gemma3, while weaker models exhibit broader distributions, and that edit distance to the oracle predicts oracle score only weakly.

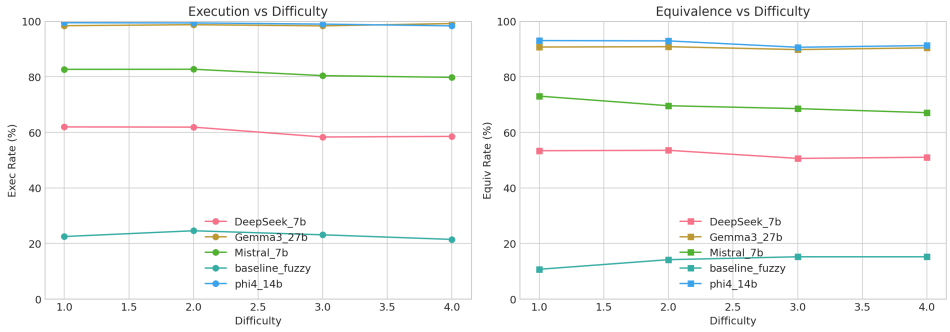


Figure 4: Suite S1: semantic equivalence by difficulty level. Top models are stable; weaker models degrade gradually.

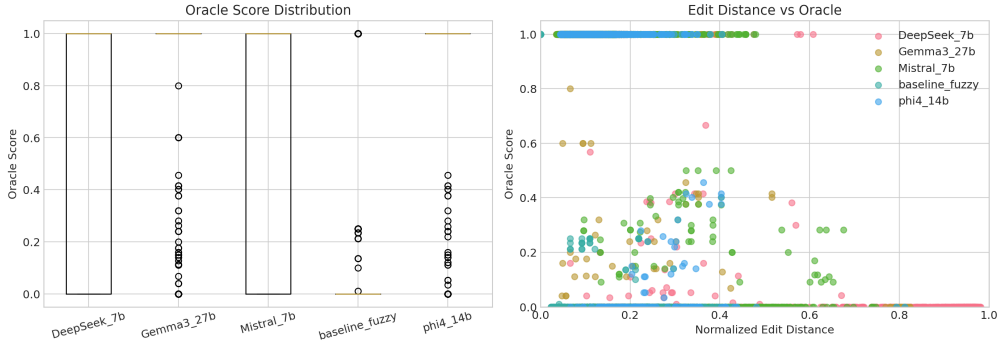


Figure 5: Suite S1. Left: oracle-score distributions by model. Right: edit distance vs. oracle score, showing a positive but noisy relationship; textual similarity is not a reliable correctness signal.

A.2 Frontier-Model Results on Suite S2

Table 14: Suite S2 (292 instances, SQLite): semantic equivalence (%) by drift category.

System	Single	Multi	Table	Mixed
Schema-diff oracle	100.0	100.0	100.0	100.0
Gemini-2.5-Flash-Lite	100.0	95.5	96.8	81.5
GPT-4o-mini	95.5	86.4	89.5	82.3
Claude-3-Haiku	68.2	68.2	98.4	78.2
Gemma3-27B	86.4	81.8	87.9	75.0
Phi-4-14B	86.4	63.6	88.7	75.8
Mistral-7B	63.6	40.9	77.4	52.4
DeepSeek-R1-7B	50.0	59.1	50.8	40.3
Fuzzy baseline	0.0	0.0	14.5	1.6

Tables 14 and 15 break suite S2 down along the same axes as the main benchmark. Two patterns replicate across the frontier models: mixed drift is the weakest category for four of the seven learned systems, and no system exceeds 82.3% on it; the fuzzy baseline remains near zero everywhere except isolated table renames (14.5%), where the error message names the missing relation outright. The exceptions are informative. Claude-3-Haiku exhibits an unusually uneven profile, reaching 98.4% on table renames but only 68.2% on both column categories. Phi-4 and Mistral are weakest on *multi*-column drift (63.6% and 40.9%), indicating that their errors concentrate in multi-site column updates rather than in compound table-plus-column changes.

Table 15: Suite S2 (292 instances, SQLite): semantic equivalence (%) by query complexity. Agg = aggregation, Sub = subquery, Win = window.

System	Simple	Agg	Join	Sub	CTE	Win
Oracle	100.0	100.0	100.0	100.0	100.0	100.0
Gemini-2.5-FL	93.2	77.3	95.2	92.5	87.5	92.5
GPT-4o-mini	86.4	75.0	92.9	92.5	77.5	90.0
Claude-3-H.	90.9	72.7	79.8	95.0	87.5	92.5
Gemma3-27B	84.1	75.0	83.3	85.0	72.5	90.0
Phi-4-14B	81.8	84.1	75.0	87.5	72.5	92.5
Mistral-7B	75.0	54.5	50.0	77.5	60.0	75.0
DeepSeek-7B	40.9	47.7	44.0	62.5	37.5	52.5
Fuzzy	2.3	6.8	4.8	12.5	12.5	5.0

B Real-Migration Slice and Current-Model Experiments

This appendix details suite S8 and the experiments of §7.6. Every table is generated by a script from the per-evaluation records in the supplementary material.

B.1 Construction of the Real-Migration Slice

Sources. We selected 40 open-source Ruby-on-Rails applications (government services, education, commerce, community, and publishing platforms) whose default branch commits `db/schema.rb` or `db/structure.sql`, and cloned their complete histories. Every migration ever added under `db/migrate/` or `db/post_migrate/` was scanned (Table 16). Six projects yield no verified rename: three contain no rename migration at all, two committed their schema snapshot only after their renames (no snapshot exists before the migrating commit), and one rename never appears in any later snapshot. S8 therefore draws on the remaining 34 projects, all of which contribute instances.

Candidate operations and verification. Only renames in the forward direction of a migration are extracted (the `change`, `up`, and `self.up` methods; `down` is ignored): `rename_column`, `rename_column_concurrently`, `rename_table`, `rename_table_safely`, and `t.rename` inside `change_table`. A candidate is accepted only if the snapshot at the parent of the migrating commit contains the old identifier and not the new one (for a column, in its owning table, resolved through a same-commit table rename if necessary), and a snapshot at that commit or one of the next 15 commits that touched a snapshot contains the new identifier in place of the old. Schema dumps are often committed after the migration, which the window accounts for; a parsing mistake can only cause an operation to be dropped.

Working schema, data, and queries. For each verified commit, the pre-migration snapshot is restricted to the renamed tables, up to three foreign-key neighbors (Rails `<singular>_id` convention), and distractor tables drawn from the same schema, up to eight tables (more when a commit renames more). Composite primary-key constraints are dropped; the columns are retained. Only the verified renames are applied, through a canonical plan keyed by original identifiers; other changes made in the same commit are excluded. Each table receives 36 deterministic rows synthesized from its column types; foreign keys are drawn from the referenced key range so that joins return rows, nullable columns receive occasional NULLs, and string values never contain column names. Queries instantiate six alias-qualified templates (simple, aggregation, join, subquery, CTE, window) over the renamed identifiers; the oracle is the same template rendered with post-drift names, and every instance must pass the three-way validation of §5.1 and return at least one row. We also searched each project for SELECT statements referencing a renamed identifier (views in the schema snapshot, SQL heredocs in Ruby sources, `.sql` files) and excluded statements with bind parameters or interpolation; only ten qualified, eight returned no rows on the synthesized data, one failed to execute, and one failed to parse, so S8 contains templated queries only. From 4,083 valid candidates, 300 instances are drawn round-robin across projects, stratified by template.

Table 16: Suite S8: mining, verification, and slice statistics.

Quantity	Value
Repositories cloned (full history)	40
Migration files scanned	18,039
Candidate rename operations (forward direction)	1,309
rejected: no schema snapshot before the commit	292
rejected: old identifier absent before	88
rejected: new identifier already present	18
rejected: not reflected within 15 schema commits	41
rejected: duplicate	15
Verified operations (column / table renames)	855 (627 / 228)
Commits with verified renames / contributing projects	508 / 34
Rename similarity, mean [IQR]	0.61 [0.48, 0.76]
Valid candidate instances / selected	4,083 / 300
Selected: projects, max per project	34, 10
Working schema: tables, median [range]	8 [4, 15]
Working schema: columns, median [IQR]	88 [69, 106]
Project SQL candidates / valid	10 / 0

Listing 1: A real-migration instance (bike_index, February 2021). The migration renamed `impound_record_updates.resolved` to `processed`; the joined table keeps a column `resolved_at`.

```
# Failing query (captured error: no such column: a.resolved)
SELECT a."resolved", b."status" FROM "impound_record_updates" a
JOIN "impound_records" b ON a."impound_record_id" = b."id"
WHERE a."resolved" IS NOT NULL;

# Oracle repair                                  (4 of 12 models)
SELECT a."processed", b."status" FROM ... WHERE a."processed" IS NOT NULL;

# Executes, not equivalent       (GLM-5.3-Flash, gpt-oss-120b, ...)
SELECT b."resolved_at", b."status" FROM ... WHERE b."resolved_at" IS NOT NULL;
```

B.2 Detailed Results on S8

Table 17: Suite S8 (300 instances): R2 results with 95% bootstrap CIs, and the paired R2→R3 difference with exact McNemar p.

System	Exec	Equiv	95% CI	Gap	R3	Δ	p
Schema-diff oracle (R1)	100.0	100.0	[100.0, 100.0]	0.0	–	–	–
GLM-5.3-Flash	100.0	89.7	[86.3, 93.0]	10.3	100.0	+10.3	$<10^{-9}$
DeepSeek-V4.1-Flash	99.3	88.3	[85.0, 91.7]	11.0	100.0	+11.7	$<10^{-10}$
GPT-6-Luna	99.7	88.0	[84.3, 91.7]	11.7	99.7	+11.7	$<10^{-10}$
Claude-Haiku-4.5	99.7	83.3	[79.0, 87.3]	16.3	96.3	+13.0	$<10^{-9}$
Gemini-3.1-FL	99.7	83.3	[79.0, 87.3]	16.3	98.7	+15.3	$<10^{-13}$
gpt-oss-120b	100.0	83.3	[79.0, 87.0]	16.7	98.0	+14.7	$<10^{-11}$
Ministral-8B	94.0	75.0	[70.0, 80.0]	19.0	93.0	+18.0	$<10^{-10}$
Claude-3-Haiku	92.3	76.3	[72.0, 81.0]	16.0	94.7	+18.3	$<10^{-12}$
Gemini-2.5-FL	95.3	76.0	[71.3, 80.3]	19.3	95.7	+19.7	$<10^{-13}$
GPT-4o-mini	98.0	74.3	[69.3, 79.0]	23.7	97.7	+23.3	$<10^{-19}$
Gemma3-27B	91.0	73.3	[68.3, 78.0]	17.7	97.0	+23.7	$<10^{-19}$
Phi-4-14B	87.0	51.3	[46.0, 56.7]	35.7	79.0	+27.7	$<10^{-12}$
Schema-diff heuristic	61.7	45.7	[40.3, 51.3]	16.0	–	–	–
Fuzzy baseline	15.7	13.7	[10.0, 17.7]	2.0	–	–	–

Table 18: Suite S8: Equiv (%) by drift category and query template, averaged over models (R2).

	Drift category				Query template					
	Single	Multi	Table	Mixed	Simple	Agg	Join	Sub	CTE	Win
Instances	183	32	74	11	44	42	50	49	61	54
Current models (7)	80.4	93.3	93.8	62.3	88.3	86.1	80.0	76.4	87.1	88.4
All 12 models	74.5	84.4	90.4	49.2	81.8	80.6	75.0	71.1	78.3	84.6

B.3 Current Models on the Synthetic Suites

Table 19: Current and re-served paper models on the synthetic suites (SQLite, same instances as Tables 3 and 5; R2 prompt; one serving endpoint).

	S6 def.	S6 held.	S7 hosp.	S3 split		S4 transform	
System	Equiv	Equiv	Equiv	Exec	Equiv	Exec	Equiv
GLM-5.3-Flash	95.7	94.0	95.7	100.0	100.0	100.0	86.5
DeepSeek-V4.1-Flash	91.4	93.1	90.5	100.0	100.0	100.0	70.2
GPT-6-Luna	94.0	95.7	91.4	100.0	100.0	100.0	66.3
Claude-Haiku-4.5	94.8	92.2	94.8	100.0	100.0	100.0	46.2
Gemini-3.1-FL	90.5	87.1	92.2	100.0	100.0	100.0	53.8
gpt-oss-120b	86.2	87.9	87.1	100.0	100.0	100.0	58.7
Ministral-8B	93.1	97.4	88.8	90.0	89.0	100.0	48.1
Claude-3-Haiku	81.9	87.1	75.0	84.0	34.0	100.0	87.5
Gemini-2.5-FL	91.4	87.1	84.5	100.0	100.0	100.0	15.4
GPT-4o-mini	90.5	81.0	80.2	100.0	100.0	100.0	58.7
Gemma3-27B	93.1	83.6	81.9	100.0	96.0	100.0	23.1
Phi-4-14B	88.8	51.7	35.3	96.0	87.0	97.1	22.1
Schema-diff oracle (R1)	100.0	100.0	100.0	–	–	–	–
Schema-diff heuristic	–	–	–	80.0	0.0	–	–
Determ. rewrite	–	–	–	–	–	100.0	6.7
Fuzzy baseline	10.3	0.0	2.6	80.0	0.0	0.0	0.0

Table 19 repeats the structural and robustness suites with current models. Table splits are solved (89–100%). Semantic transforms remain difficult: every current model executes all 104 instances, but Equiv ranges from 46.2% to 86.5%, with no relation to recency (Claude-Haiku-4.5 46.2% vs. Claude-3-Haiku 87.5%). Current models are strong on the rename suites (86–97%), which is why the default-vocabulary suite no longer distinguishes them.

B.4 Reproducing the Paper's Systems

Table 20: Paper results vs. re-runs through the common endpoint, paired by instance. Agree = per-instance agreement on Equiv; p from an exact McNemar test.

Suite	System	n	Equiv (paper → re-run)		Δ	Agree	p
S6 default dict.	Phi-4-14B	116	85.3	88.8	+3.4	89.7	0.388
S6 default dict.	Gemma3-27B	116	81.0	93.1	+12.1	86.2	$<10^{-3}$
S6 held-out dict.	Phi-4-14B	116	86.2	51.7	−34.5	56.9	$<10^{-8}$
S6 held-out dict.	Gemma3-27B	116	68.1	83.6	+15.5	81.0	$<10^{-3}$
S7 hospital	Phi-4-14B	116	80.2	35.3	−44.8	48.3	$<10^{-12}$
S7 hospital	Gemma3-27B	116	71.6	81.9	+10.3	84.5	0.008
S4 col. transform	GPT-4o-mini	104	59.6	58.7	−1.0	95.2	1.000
S4 col. transform	Claude-3-Haiku	104	86.5	87.5	+1.0	99.0	1.000
S4 col. transform	Gemini-2.5-FL	104	15.4	15.4	+0.0	100.0	1.000
S4 col. transform	Phi-4-14B	104	12.5	22.1	+9.6	75.0	0.076
S4 col. transform	Gemma3-27B	104	10.6	23.1	+12.5	87.5	$<10^{-3}$

The API models reproduce closely: on S4, their per-instance agreement with the paper's records is 95.2–100%, no rate changes by more than 1.0 pp, and Claude-3-Haiku scores 34.0% on S3 versus 33.0% in Table 3. The open models do not reproduce across serving stacks: when served through the common endpoint rather than the original local server, Gemma3-27B gains 10–16 pp while Phi-4 loses 34.5 and 44.8 pp on the held-out and hospital suites, respectively, mostly through repairs that execute but return wrong rows (45 of 56 and 59 of 75 failures). We therefore compare models only within a single serving stack, as in §7.6. Repeated runs through the same stack are stable: the two models evaluated in both experiments agree on 93–100% of instances, with rates within 3 pp.

B.5 Rank Transfer

Table 21: Spearman correlation between Equiv on S8 and on each synthetic suite; p from 20,000 permutations. "All learned models" adds the within-family scaling models where they were evaluated; the Qwen3 runs marked ‡ in Table 23 are excluded.

	12 main models			All learned models		
Synthetic suite	n	ρ	p	n	ρ	p
S6 default dict.	12	0.39	0.217	23	0.65	$<10^{-3}$
S6 held-out dict.	12	0.70	0.015	23	0.88	$<10^{-4}$
S7 hospital	12	0.80	0.002	12	0.80	0.003
S3 table split	12	0.62	0.034	12	0.62	0.037
S4 col. transform	12	0.65	0.025	23	0.72	$<10^{-3}$

B.6 Execution Feedback

In feedback mode, the model's first answer is executed on the drifted database. If it fails, the database error is returned with a request for a corrected query. In the *preview* condition, when a query executes, the model receives its column names, row count, and first five rows, together with a request to verify that it preserves the intent of the original query, "including the meaning and units of every returned value", and to return it unchanged if so. A *neutral preview* shows the same rows but asks only whether the query preserves the intent of the original query. The loop ends when a query executes (error condition), when the model repeats its previous query (preview conditions), or after three turns. The pre-drift result, oracle, and migration plan are never shown.

Error feedback changes Equiv on S8 and S4 by at most 2.3 pp for any model (Table 22). The effect of preview feedback depends on the wording: without the words "meaning and units", the average S4 gain falls from 37.3 to 19.4 pp and GPT-4o-mini no longer improves at all, while the S8 regressions of GPT-4o-mini and GPT-6-Luna (-7.7 and -5.7 pp) shrink to -2.0 and -1.0 pp (n.s.). Naming units therefore steers models toward rescaling, which is correct on S4 but sometimes incorrect on S8.

Table 22: Change in Equiv (pp) from the first to the final turn, per model and feedback condition (Neutral: preview with a neutral request). *: $p < 0.01$, exact McNemar on the paired first/final outcomes.

	S8 real slice			S4 col. transform		
Model	Error	Preview	Neutral	Error	Preview	Neutral
DeepSeek-V4.1-Flash	+0.0	+0.7	+1.0	+0.0	+25.0*	+14.4*
GPT-4o-mini	+0.3	−7.7*	−2.0	+0.0	+23.1*	+0.0
GPT-6-Luna	+0.3	−5.7*	−1.0	+0.0	+39.4*	+18.3*
Gemini-3.1-FL	+0.3	+2.7	+1.7	+0.0	+41.3*	+22.1*
Gemma3-27B	+2.3	+2.3	+2.3	+0.0	+49.0*	+24.0*
Llama-3.3-70B	+0.3	−0.3	−1.7	+0.0	+46.2*	+37.5*

B.7 Within-Family Scaling

S8 Equiv rises with size in every family (Llama-3 0.7→77.0%, Ministral-3 64.0→80.0%, Gemma-3 26.0→73.3%, Qwen3 68.7→83.0%), and so does S4 Equiv except in Gemma-3, where the 27B

Table 23: Equiv (%) by model size within families (R2, one serving endpoint). Qwen3 models are served by a provider that honors the no-thinking setting. ‡: first-run Qwen3 models routed to providers that ignored it (reasoning tokens were recorded), reported separately. Gemma3-4B completed 285 of 300 S8 instances (provider rate limits).

Family	Model	S8 real	S6 def.	S6 held.	S4 transf.
Llama-3	Llama-3.2-1B	0.7	4.3	9.5	9.6
	Llama-3.2-3B	26.3	57.8	45.7	27.9
	Llama-3.1-8B	51.7	72.4	63.8	28.8
	Llama-3.3-70B	77.0	87.9	84.5	45.2
Ministral-3	Ministral-3B	64.0	85.3	77.6	15.4
	Ministral-8B	75.3	92.2	96.6	51.0
	Ministral-14B	80.0	94.0	90.5	61.5
Gemma-3	Gemma3-4B	26.0	53.4	42.2	23.1
	Gemma3-12B	60.3	87.9	82.8	61.5
	Gemma3-27B	73.3	93.1	83.6	25.0
Qwen3	Qwen3-8B	68.7	91.4	75.9	23.1
	Qwen3-14B	76.0	87.1	81.9	31.7
	Qwen3-235B-A22B	83.0	88.8	86.2	73.1
Qwen3‡	Qwen3-14B‡	85.0	89.7	85.3	15.4
	Qwen3-32B‡	83.7	88.8	88.8	40.4

model falls to 25.0% against 61.5% for 12B. Letting Qwen3-14B reason (‡) raises its S8 Equiv from 76.0 to 85.0%.

B.8 Extraction Normalization

Some models copy the prompt's final line, `Corrected SQL:`, into their answer or leave a Markdown fence unterminated. The extraction procedure in §6 retains this text, so the candidate fails to parse and is counted as a non-executing repair even when the SQL itself is correct. For §7.6, we strip only such a leading label and dangling fences, never modifying the SQL, and re-execute the candidate with the same namespaces, data, and comparator. This changes 480 of the 33,213 scored model responses (1.4%), converts 440 non-equivalent outcomes into equivalent ones and none in the opposite direction, and matters primarily for the R3 prompt, where Gemini-3.1-Flash-Lite prefixed every answer with the label (Table 24). Re-executing 3,000 randomly sampled unchanged responses reproduced every recorded outcome on the original machine and all but one on a second machine, whose older SQLite build lacks a function used by that response. Applied to our original runs, the normalization changes only DeepSeek-R1-7B, by up to 11.2 pp; all other models, and every number in §7.1–§7.5, use the original extraction.

Table 24: Equiv (%) under strict and normalized extraction, for every model–suite pair that changes by more than 0.4 pp.

Model	Suite, prompt	n	Strict	Normalised
Claude-3-Haiku	S8 real slice, R3	300	93.7	94.7
GLM-5.3-Flash	S8 real slice, R3	300	66.0	100.0
Gemini-3.1-FL	S8 real slice, R3	300	0.0	98.7
Qwen3-32B‡	S4 col. transform, R2	104	39.4	40.4
Qwen3-32B‡	S6 default dict., R2	116	87.9	88.8
Qwen3-32B‡	S6 held-out dict., R2	116	85.3	88.8
Qwen3-32B‡	S8 real slice, R2	300	73.0	83.7
DeepSeek-R1-7B (paper)	S6 default dict., R2	116	39.7	50.9
DeepSeek-R1-7B (paper)	S6 held-out dict., R2	116	24.1	28.4
DeepSeek-R1-7B (paper)	S7 hospital, R2	116	32.8	37.9

B.9 Serving Configuration and Evaluation Cost

All additional evaluations used one OpenRouter endpoint with the prompts of §5, temperature 0 and top-p 1 wherever accepted (GPT-6-Luna accepts neither), and each model's default reasoning effort; reasoning tokens are recorded per call. DeepSeek-V4.1-Flash was routed to its lowest-price host (§). The 33,213 scored model evaluations (40,627 requests including feedback turns) cost $6.59 in total (Table 25); 15 Gemma3-4B requests failed after retries because of provider rate limits and are excluded.

Table 25: Models, serving, and cost of the additional experiments: 24 models in 25 rows, since Qwen3-14B was run twice (with thinking disabled, and ‡ with it left on by the provider).

Model	OpenRouter id	T=0	Jobs	Cost ($)
Claude-3-Haiku	`anthropic/claude-3-haiku`	yes	1152	0.25
Claude-Haiku-4.5	`anthropic/claude-haiku-4.5`	yes	1152	1.03
DeepSeek-V4.1-Flash	`deepseek/deepseek-v4.1-flash`§	yes	2796	1.00
Gemini-2.5-FL	`google/gemini-2.5-flash-lite`	yes	1152	0.09
Gemini-3.1-FL	`google/gemini-3.1-flash-lite`	yes	2796	0.98
Gemma3-12B	`google/gemma-3-12b-it`	yes	636	0.02
Gemma3-27B	`google/gemma-3-27b-it`	yes	3432	0.37
Gemma3-4B	`google/gemma-3-4b-it`	yes	621	0.02
GLM-5.3-Flash	`z-ai/glm-5.3-flash`	yes	1152	0.24
GPT-4o-mini	`openai/gpt-4o-mini`	yes	2796	0.47
GPT-6-Luna	`openai/gpt-6-luna`	no	2796	0.56
gpt-oss-120b	`openai/gpt-oss-120b`	yes	1152	0.10
Llama-3.1-8B	`meta-llama/llama-3.1-8b-instruct`	yes	636	0.02
Llama-3.2-1B	`meta-llama/llama-3.2-1b-instruct`	yes	636	0.02
Llama-3.2-3B	`meta-llama/llama-3.2-3b-instruct`	yes	636	0.03
Llama-3.3-70B	`meta-llama/llama-3.3-70b-instruct`	yes	2280	0.45
Ministral-14B	`mistralai/ministral-14b-2512`	yes	636	0.08
Ministral-3B	`mistralai/ministral-3b-2512`	yes	636	0.04
Ministral-8B	`mistralai/ministral-8b-2512`	yes	1788	0.17
Phi-4-14B	`microsoft/phi-4`	yes	1152	0.06
Qwen3-14B	`qwen/qwen3-14b`§	yes	636	0.10
Qwen3-14B‡	`qwen/qwen3-14b`	yes	636	0.12
Qwen3-235B-A22B	`qwen/qwen3-235b-a22b`§	yes	636	0.21
Qwen3-32B‡	`qwen/qwen3-32b`	yes	636	0.12
Qwen3-8B	`qwen/qwen3-8b`§	yes	636	0.05
Total			33213	6.59

C Benchmark Instance Examples

Listing 2 shows a mixed-drift rename instance, and Listing 3 a column semantic transform, which is the case where syntactic repair and semantic repair come apart.

Listing 2: Rename instance (mixed drift).

```
# Original query (valid on original schema)
SELECT c.customer_name, COUNT(o.order_id)
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_name;

# Drift operations
1. rename_table:  customers -> clients
2. rename_column: customers.customer_id -> uid

# Captured error (original query, drifted schema)
ERROR: relation "customers" does not exist
SQLSTATE: 42P01

# Oracle repair (AST rewriting)
SELECT c.customer_name, COUNT(o.order_id)
FROM clients c
JOIN orders o ON c.uid = o.customer_id
GROUP BY c.customer_name;
```

Listing 3: Column semantic transform: the syntactic repair executes but is wrong by a factor of 100.

```
# Drift operation
products.unit_price (DECIMAL) ->
products.price_cents (INTEGER), values x 100

# Original query
SELECT p.category, SUM(p.unit_price)
FROM products p GROUP BY p.category;

# Mapping-based repair: EXECUTES, NOT EQUIVALENT
SELECT p.category, SUM(p.price_cents)
FROM products p GROUP BY p.category;

# Oracle repair
SELECT p.category, SUM(p.price_cents / 100.0)
FROM products p GROUP BY p.category;
```

D PRACTICAL RECOMMENDATIONS

1. **Validate repairs by result, not by execution.** Compare the repaired query's output against the original query's output on a pre-migration snapshot. Our results show execution success overstates correctness by 7–12 pp even for the strongest models, and by up to 93 pp for mapping-based rewriting under semantic drift.
2. **Attempt deterministic resolution first.** If both schema versions are available and the drift is rename-only, a schema diff plus AST rewrite is exact (100%) and cheaper than any model.
3. **Route structural drift to a model.** When the change cannot be expressed as an identifier mapping, splits, merges, unit changes, deterministic rewriting is not merely weaker, it is systematically wrong (0–6.7% equivalence at 80–100% execution).
4. **Preserve and propagate migration metadata.** Explicit change descriptions are worth up to +17.6 pp for mid-tier models, though not for the strongest or weakest.
5. **Escalate compound drift for review.** Mixed table+column renames are the hardest rename category for every system tested.
6. **Do not treat execution feedback as validation.** Returning database errors to a model makes its repairs run, not makes them right; on real migrations 93% of current-model failures already execute.
7. **Watch for unit and type changes specifically.** These are invisible to syntactic checks and are the failure mode most likely to reach a dashboard undetected.

E EXTENDED LIMITATIONS

Generated workloads and the scope of the real slice. The main suites are generated rather than harvested from production, which is what lets us guarantee by execution that drift affects each query and that a correct repair exists, and avoids the contamination risk of scraped SQL. Suite S8 replaces the synthetic schema and synthetic renames with real ones, but not the workload: its queries are templated and its rows synthesized, because the projects' own code contained almost no self-contained SELECT statements over the renamed identifiers (Appendix B). It also covers renames only, in Rails applications with a committed schema snapshot, and applies each rename in isolation even when the same commit made other changes. Real analytic workloads over real data, and real structural migrations, remain open.

Drift coverage. We model renames, table splits, and column semantic transforms. Real evolution also includes type changes, column additions and deletions, constraint changes, table merges, and denormalization (Curino et al., 2008). The framework accommodates them, adding the two structural families required no change to generation, validation, or scoring, but they are not yet implemented, so our absolute numbers should be read as covering three drift families rather than schema evolution in full.

Semantic equivalence is empirical. Deciding SQL equivalence symbolically is undecidable in general (Chu et al., 2018); we compare result multisets on concrete data. A repair that differs from the oracle only on data our generator never produces would be scored as equivalent. We mitigate this with seeded, varied value distributions and by requiring positional equality for ordered queries, but the metric remains a sound proxy rather than a proof.

Schema and vocabulary scope. Results cover two schema domains (e-commerce, hospital) and two rename vocabularies. The cross-domain and held-out-dictionary studies show the ranking is stable and the strongest model robust, but they also show mid- and low-tier absolute scores drop by 5–15 pp when conventions change. Those absolute numbers are therefore vocabulary- and domain-dependent, and should not be quoted without their setting.

Query complexity. Templates are bounded: 2–3 table joins, standard aggregates, single-statement queries. Production queries can nest more deeply, span more tables, and use vendor-specific syntax and user-defined functions. Our results are best read as a lower bound on repair difficulty.

Oracle reliability and harness effects. Oracle generation depends on `sqlglot` parsing, which fails on roughly 2% of generated queries and falls back to a conservative regex path. Additionally, absolute rates differ between the PostgreSQL and SQLite harnesses for the same models (§7); cross-suite comparisons of absolute numbers are not meaningful and we avoid them throughout.

Model and serving variability. LLM outputs vary across API versions and serving stacks. We use deterministic decoding (temperature 0) and fixed prompts, and every recorded evaluation carries SHA-1 hashes of its prompt and response, but exact reproduction on a different deployment of the same architecture is not guaranteed.

F Artifact Contents

An anonymized artifact containing the benchmark instances, generation and evaluation code, experiment configurations, and documentation is provided as supplementary material; a public repository will be made available upon acceptance. It comprises: the validated suites S3–S7 as JSONL and a 50-instance sample of S1, each instance carrying its seed, drift operations, canonical migration plan, failing query, and oracle repair; per-evaluation records as CSV for S2–S7 and the S1 sample (execution success, semantic equivalence, latency, and the extracted repair, with the oracle score, edit distance, captured database error, and SHA-1 hashes of the prompt and response for S1–S3, S5, and S6), where the S2 records also carry each instance's failing query and oracle repair; machine-readable configurations for all seven suites; the generator and evaluation harness; and an offline test suite that checks the S2–S7 suite sizes and reported rates against these files. For S8 and §7.6 it adds the mined rename events with provenance (repository, commit, migration file), the validated slice, the mining, slice-building, and experiment scripts with their model configuration, the per-call records (including tokens, cost, and serving provider), the re-scoring script, and the script that generates the corresponding tables. Every record is keyed by seed and instance ID, and generation is deterministic, so any suite can be rebuilt exactly and any evaluation located (Gundersen & Kjensmo, 2018); the S3 records were produced with an earlier data-seed convention, documented in the artifact, which changes the synthetic rows but not the queries or oracle repairs. Decoding for S1–S7 is deterministic (temperature 0, top-p 1.0, 2,048-token budget; §7.6 uses temperature 0 wherever the provider accepts it and up to 16,000 tokens for reasoning models), and suite composition, seeds, and harness for every reported number are given in Table 9. Code is licensed Apache-2.0 and data CC-BY-4.0; all data rows are synthetic.