TABQUERYBENCH: A QUERY-CENTRIC BENCHMARK FOR SYNTHETIC TABULAR DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthetic tabular data support use cases like data sharing, model development under access restrictions, and rapid prototyping of analytical workflows. Modern generative models are evaluated by their statistical similarity, correlation structure, privacy, and downstream machine-learning utility. However, such evaluations leave a gap: they rarely test the structure that matters for analytical queries. We present TABQUERYBENCH, a query-centric benchmark for evaluating synthetic tabular data with SQL-shaped analytical queries. TABQUERYBENCH distills 12 public query sources into 44 reusable query templates and grounds them to each dataset through a policy-guided template-to-SQL pipeline. Across 49 datasets and 11 generative models, it produces more than 100 executable queries per dataset while preserving cross-model comparability. Our evaluation reveals a substantial gap between conventional and query-centric fidelity. Even the strongest model, RealTabFormer, reaches only 0.76 ± 0.23 query-centric fidelity compared with 1.00 for real data. Failures are most pronounced for high-cardinality attributes and local or tail queries, while model quality also trades off against generation cost. These results show that synthetic data can appear statistically realistic yet still fail to preserve the analytical behavior of real data, motivating query-centric fidelity as a necessary complement to existing evaluation metrics. TABQUERYBENCH is open-sourced and available for anonymous review. [1]

1 INTRODUCTION

Synthetic tabular data are now used for data sharing (Bates et al., 2019; Personal Data Protection Commission Singapore, 2024; Jiang et al., 2024), model development under access restrictions (Bates et al., 2019; Personal Data Protection Commission Singapore, 2024; Chu et al., 2024), and rapid algorithm or system prototyping (Patki et al., 2016; Walonoski et al., 2018; Kannan et al., 2026; Gupta et al., 2025). As these use cases mature, benchmarking matters more, because what makes synthetic data useful in these settings is not distributional resemblance alone, but the preservation of the analytical properties that the database community has long treated as the core measure of data quality (Poess & Floyd, 2000; Nambiar & Poess, 2006; Poess et al., 2002; Ding et al., 2021; Transaction Processing Performance Council, 1999; 2017; Jiang et al., 2023; Chu et al., 2026).

Synthetic data consumers often care less about synthetic records as standalone samples. They care more about the functions that those records make possible. *A useful synthetic dataset does not merely approximate marginal distributions. It should ideally expose a controlled approximation of hidden dataset attributes*, such as table structure, column relationships, valid joins, and executable queries Walonoski et al. (2018); Kannan et al. (2026). This structural view is important for (1) collaboration, where a partner wants to preview the structure and quality of a sensitive dataset before accessing the real data Bates et al. (2019); Personal Data Protection Commission Singapore (2024); (2) software and data-system testing, where engineers need realistic schemas, constraints, edge cases, and relational consistency Walonoski et al. (2018); and (3) query-centric analytics, where the target is not an individual row but the answer to aggregates, joins, SQL functions, and text-to-SQL workloads Kannan et al. (2026); Caferoğlu et al. (2025).

Our key observation is that existing synthetic tabular benchmarks have limited coverage of the analytical structures that occur in real analytical queries. Current benchmarks score column-level

[1] See anonymous code in github repository, the dataset and docker images in huggingface repository.

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.

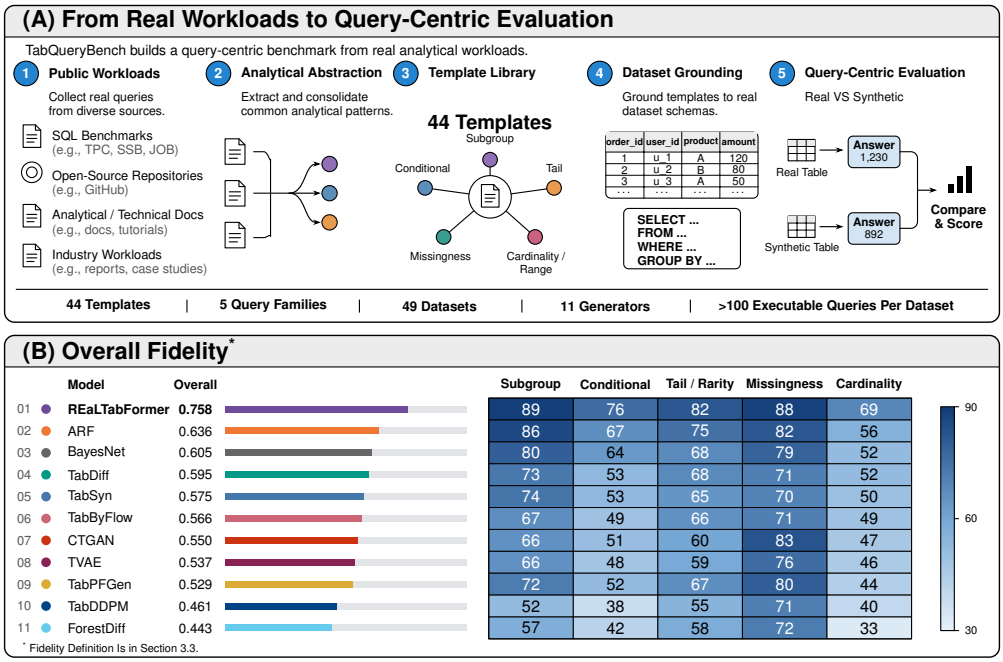


Figure 1: Overview of TABQUERYBENCH.

distance, predictive ML utility, or task-specific utility proxies (Lautrup et al., 2025; Snoke et al., 2018; El Emam et al., 2022; Hernandez et al., 2023; Kaabachi et al., 2025), none of which directly assesses the analytical properties that diverse downstream workflows care about. We also find that picking generative models based on conventional distance-based metrics may even be misleading for practitioners (van Breugel et al., 2023). A useful contrast comes from the database community, which has long treated occurring query sets as first-class evaluation objects: benchmarks such as TPC-H and TPC-DS encode representative decision-support SQL workloads designed to reflect recurring analytical patterns in industry practice (Poess & Floyd, 2000; Nambiar & Poess, 2006; Ding et al., 2021; Poess et al., 2002). Although designed to compare database engines, such recurring query patterns also provide natural *structural assessors* for synthetic data. In contrast, existing synthetic-data benchmarks do not yet use them in this role, which shows a natural reality/generation target mismatch.

Our answer: TABQUERYBENCH (Figure 1). We argue for a shift from the conventional benchmarking used for tabular generative models to *query-centric* benchmarking, where *domain-relevant queries* serve as structural assessors of synthetic-data fidelity. The key design challenge is to make these assessors both realistic and generic. TABQUERYBENCH addresses this challenge with a two-stage construction pipeline. First, it taxonomizes recurring analytical logic from public sources of analytical queries into reusable five-family templates (i.e., subgroup, conditional, tail/rarity, cardinality/range, and missingness). Second, it grounds these templates to each benchmark dataset through a constrained realization policy that binds template roles to schema-appropriate columns and validates the resulting queries. This design preserves schema-level realism while enabling comparable evaluation across datasets, domains, and generative models. TABQUERYBENCH also provides fine-grained feedback on specific queries and query families rather than only an aggregate score.

In this work, we identify the following key findings: (1) Current tabular generative models can have good distance-based fidelity, but they still fall short on query-centric fidelity; (2) Tabular generative models fail to synthesize columns with very high-cardinality discrete support; (3) SOTA generative models preserve good global conditional query-centric fidelity, but fail more on local queries; (4) Tail fidelity deteriorates as queries move toward the extreme tail, even for SOTA generative models; (5) There is a clear cost-fidelity tradeoff in tabular data generation and BayesNet achieves the best balance. We make three main contributions:

- TABQUERYBENCH is one of the first tabular data benchmarks that focus on workload-level fidelity evaluation.

Benchmark	Target setting	Eval. focus	Data	Model	①	②	③	④	⑤	Dist.	Cost
Synthcity (Qian et al., 2023)	Generic tabular	Metric suite	$18+u$	$5+u$	✗	✗	✗	△	✗	✓	✗
SDGym (DataCebo, n.d.)	Generic tabular	Global fidelity	$22+u$	u	✗	✗	✗	✗	✗	✓	✓
SyntheRela (Hudovernik et al., 2024)	Multi-table	Relational fidelity	39	6	✗	✗	✗	✗	✗	△	△
SynthRO (Santangelo et al., 2025)	Health domain	Domain validation	u	u	✗	✗	✗	✗	✗	✓	✗
SynthEval (Lautrup et al., 2025)	Generic tabular	Metric suite	u	u	✗	✗	✗	✗	✗	✓	✗
DP benchmark (Tao et al., 2021)	Private tabular	Privacy–utility	7	12	✗	✗	✗	✗	✗	✓	△
TabStruct (Jiang et al., 2026)	Generic tabular	Structural metrics	29	13	✗	△	✗	✗	✗	✓	✓
TABQUERYBENCH (ours)	Generic tabular	Query-centric fidelity	49	11	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of related benchmarks for synthetic tabular data and adjacent tabular evaluation tasks. ① subgroup, ② conditional, ③ tail, ④ missingness, and ⑤ high-cardinality. ✓ = direct, △ = partial, ✗ = absent, u = user-specified.

- TABQUERYBENCH provides an automated and practical framework for operators to generate realistic queries without dataset-specific handcrafting.
- TABQUERYBENCH reveals actionable diagnosis to operators, not just one fidelity score. Across 49 datasets and 11 generators, it exposes failures in high-cardinality support, local conditional structure, and extreme tails, while also showing fidelity–cost tradeoffs.

2 RELATED WORK AND MOTIVATION

2.1 RELATED WORKS

Tabular generative models. Tabular generative models learn distributions over heterogeneous records. Approaches range from Bayesian networks and tree-based synthesis to adversarial, variational, diffusion, transformer-based, and flow-matching models (Ping et al., 2017; Watson et al., 2023; Xu et al., 2019; Kotelnikov et al., 2023; Solatorio & Dupriez, 2023; Guzmán-Cordero et al., 2025). These approaches differ in how they represent dependencies; our focus is whether their synthetic tables preserve the answers to analytical queries. Appendix A.1 reviews the model families.

Tabular generative model benchmarks. Existing frameworks assess distributional fidelity, privacy, and predictive utility (Qian et al., 2023; Lautrup et al., 2025; Hansen et al., 2023). TabStruct further evaluates structural fidelity through conditional independence and *global utility*, which aggregates predictive utility across variables without requiring a ground-truth causal graph (Jiang et al., 2026). TABQUERYBENCH complements these perspectives by directly comparing dataset-grounded SQL answers across five analytical query families. Appendix A.2 details the distinctions and compares the evaluation settings.

Database query benchmarks. TPC-H, TPC-DS, ClickBench, and H2O db-benchmark use recurring analytical queries to evaluate database or data-processing systems (Transaction Processing Performance Council, 1999; 2017; ClickHouse, 2022; H2O.ai, 2020). We use public SQL patterns as sources of reusable templates, adapting them to assess answer preservation in synthetic tables. Appendix A.3 explains this connection.

2.2 MOTIVATION: DISTANCE–QUERY FIDELITY MISMATCH

Distance-based metrics summarize global resemblance between real and synthetic tables, but analytical queries often depend on query-relevant subsets, group identities, conditional rankings, or rare supports. This mismatch motivates evaluating synthetic tables with query-centric fidelity in addition to classical distance-based metrics (Swanberg et al., 2025). Figure 2 compares each model's Wasserstein distance fidelity with its official query score. Both axes are therefore higher-is-better. Appendix A.4 gives more details of this example.

3 BENCHMARK SCOPE AND DESIGN

3.1 SCOPE

TABQUERYBENCH evaluates whether synthetic tables preserve the answers to reusable analytical queries. The current benchmark focuses on single-table analysis, drawing on query patterns from public OLAP and decision-support collections (Poess & Floyd, 2000; Nambiar & Poess, 2006; Poess et al., 2007). These patterns capture analytical structure that can be assessed across datasets through

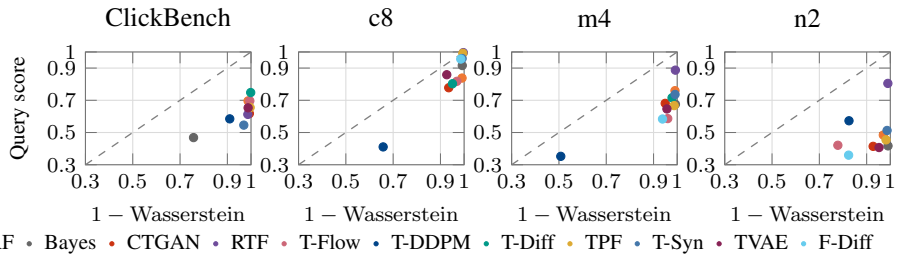


Figure 2: Distance-based and query scores do not align across generative models. The four panels show four datasets, respectively. Colors identify the models in the shared legend.

schema-aware grounding. Queries tied to specific row identifiers or private business rules, temporal forecasting beyond simple time-bucket aggregates, and relational joins are outside the scope of the current evaluation.

We consider *mimicry-style* synthetic-data generation: each generative model learns from a real training split and produces a synthetic table under the same schema, with the aim of preserving its analytical structure. We do not evaluate differentially private synthesis, generation under task-specific conditions, or text-to-table generation. The evaluation reports query-centric fidelity by analytical family, allowing different aspects of structure preservation to be examined separately.

3.2 Dataset and Generative Model Suites

Dataset suite. We curate 49 single-table datasets from UCI, Kaggle, OpenML, and HuggingFace (Kelly et al.; Kaggle; OpenML; Hugging Face), comprising 19 categorical-dominant, 19 numerical-dominant, and 11 mixed-type datasets. These regimes probe discrete support and subgroup structure, numerical ranges and tail behavior, and their interactions, respectively. The suite spans multiple application domains and includes high-cardinality and missingness-rich tables to test these structural challenges. Details of these datasets are provided in Appendix B.3, Table 6.

Generative model suite. We evaluate 11 tabular generative models spanning Bayesian networks, tree-based methods, adversarial and variational models, diffusion, transformers, and flow matching (Ping et al., 2017; Watson et al., 2023; Xu et al., 2019; Kotelnikov et al., 2023; Zhang et al., 2024; Shi et al., 2025; Jolicoeur-Martineau et al., 2024; Solatorio & Dupriez, 2023; Ma et al., 2023; Guzmán-Cordero et al., 2025). This selection enables comparisons across different modeling approaches.

For each dataset, all generative models use the same training split, schema, and synthetic sample-size requirement. The full model roster and supporting references are provided in Appendix Table 7. Detailed training, runtime, and hyperparameter settings are provided in Appendix B.2.

3.3 TabQueryBench Design

TabQueryBench constructs analytical queries by extracting and consolidating reusable templates from public SQL sources, grounding them to individual datasets, and scoring query answers.

Stage 1: Template construction. We extract recurring analytical query patterns from public SQL sources, including TPC-H, TPC-DS, ClickBench, and H2O db-benchmark (Transaction Processing Performance Council, 1999; 2017; ClickHouse, 2022; H2O.ai, 2020), with additional patterns from curated SQL repositories and database engine documentation. We consolidate these patterns into 44 reusable templates and deduplicate overlapping templates across sources. Appendix D.2 provides the detailed construction procedure and source-to-template attribution.

Query families. The resulting library contains 44 templates organized into five families, each probing a distinct aspect of analytical structure. The five families are derived from recurring single-table analytical patterns in our source registry and are kept separate because they probe distinct properties that synthetic data must preserve. **Subgroup** (16 templates) measures grouped summary statistics and subgroup sizes; **Conditional** (13 templates) measures conditional summaries at full grouped scope and within filtered slices; **Tail / rarity** (8 templates) measures rare-member coverage and tail magnitudes or sizes; **Missingness** (3 templates) measures marginal and bucket-conditioned missingness patterns; and **Cardinality / range** (4 templates) measures discrete support and rank behavior, together with numerical ranges. These families target structural properties whose preservation may not be captured

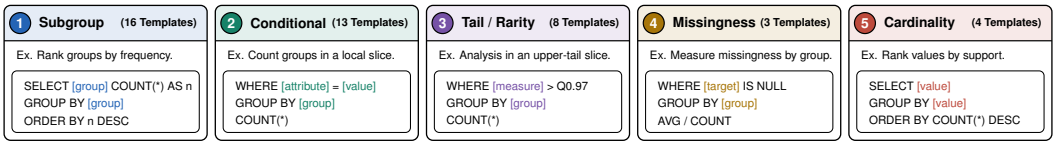


Figure 3: The five query families in TABQUERYBENCH and their representative examples.

by aggregate statistical distances. Figure 3 summarizes the five-family taxonomy and its representative analytical questions, while Appendix C.3 lists the complete template inventory, and Table 5 gives a grounded question and SQL example for each family.

Stage 2: Dataset grounding and SQL realization. For each dataset, the agent uses its schema and data profile to select applicable templates and eligible columns under fixed template policies, skipping templates without valid bindings. The agent performs constrained SQL realization while preserving the analytical structure specified by each template. Accepted SQL queries are accompanied by natural-language questions for readability. Across the 49 benchmark datasets, each dataset activates at least 10 templates and yields more than 100 executable SQL queries. Appendix D.3 describes the grounding and SQL realization procedure.

Query scoring. We execute the grounded SQL queries on real and synthetic data and compare their answers using scoring rules tailored to the different SQL result output types. The detailed scoring standards are provided in Appendix E.

4 EVALUATION

4.1 SETUP

We evaluate TABQUERYBENCH on 49 datasets and 11 synthetic tabular generative models. For each dataset, we split the real table into a training split and a held-out test split (4:1). Each generative model is trained only on the training split and is then asked to synthesize a table under the same schema. The synthetic table is generated with the same number of rows as the training split.

We then execute the same grounded SQL workload on both the real training table and the synthetic table, and compute query fidelity by comparing their query answers. The held-out test split is retained for future downstream evaluation and is not used in the current scoring pipeline. This setup ensures that all generators are evaluated under the same input schema, output contract, and workload.

The agent selects applicable templates and dataset columns. The profiler, binder, and validator are deterministic. Only the SQL realizer calls an LLM, and it does so only inside the constrained template-to-SQL realization step. We package each tabular generative model in its own Docker image and use the recommended configuration from the original implementation when available, otherwise tuning within a bounded search range chosen to balance quality and runtime. Full details on SQL grounding, validation, infrastructure, API cost, and hyperparameter ranges are provided in Appendix B.2.

4.2 HOW WELL DO STATE-OF-THE-ART TABULAR GENERATIVE MODELS PERFORM UNDER QUERY-CENTRIC EVALUATION?

Finding 1: Current state-of-the-art tabular generative models often appear faithful under distance-based metrics, but still fall well short on query-centric fidelity (Figure 4).

To provide an overview of TABQUERYBENCH results, we evaluate whether current tabular generative models preserve the analytical behavior exposed by dataset-grounded analytical queries. We are also curious about whether this query-centric approach provides a different view compared to conventional distance-based fidelity.

Figure 4 compares each generative model's distance-based overall score with its query overall score, with both scores computed against the same real training split and the same-size synthetic output. Every generative model lies below the diagonal, so the query-centric view is uniformly stricter than the distance-based one rather than reflecting a mismatch in reference splits. The ordering also shifts. RealTabFormer remains strongest overall, but ARF and TabPFGen look especially strong under

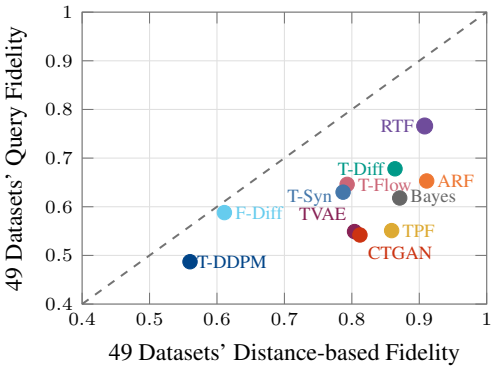


Figure 4: Distance-based and query overall scores do not align across generative models. Detailed values appear in Appendix Table 18.

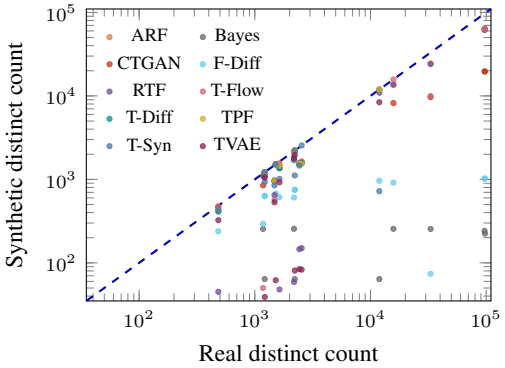


Figure 5: Real vs. synthetic distinct counts on high-cardinality columns. Current generative models still struggle to preserve high-cardinality support. Details in Table 2.

distance-based fidelity relative to their query-side scores, while BayesNet remains comparatively stronger once we ask whether the same synthetic tables preserve the structure that matters for analytical queries. The full per-generative-model summary is deferred to Appendix Table 18.

The appendix table also shows that the failure is not uniform across query families. RealTabFormer is strongest on the main query families, reaching $0.83_{\pm 0.20}$ on Tail/Rarity, $0.75_{\pm 0.26}$ on Conditional, and $0.81_{\pm 0.26}$ on Missingness, but weaker generators still drop sharply in these same families. Thus, distance-based fidelity can give an overly optimistic picture: it hides the subgroup, conditional, and rare-event failures that matter for real analytical use.

4.3 Where Do Failures Under Query-Centric Fidelity Concentrate?

A major goal for TabQueryBench is to localize failure modes rather than reporting only an aggregate query score. We study two complementary axes: query families, which reveal difficult structures to preserve, and data regimes, which reveal how categorical, numerical, and mixed schemas change the failure profile.

4.3.1 Breaking Down By Query Family

We break down query-centric fidelity across the query families and demonstrate analysis and insights. We first localize failures by query family. This view asks which kinds of analytical query objects are most fragile under synthetic generation, rather than averaging all query behavior together. In the main text, we focus on three representative hard regimes. Conditional queries test whether grouped structure remains reliable after the analysis is restricted to a filtered local slice. Tail and rarity queries test whether rare regions remain both identifiable and queryable as the support becomes more extreme. High-cardinality queries test whether synthetic data preserves the large discrete support that matters for analytical queries. For each family, we follow the same progression: we first define the query object, then show one grounded example, then summarize the benchmark evidence, and finally explain what the result implies about current synthetic tabular generative models.

Finding 2: Current SOTA tabular generative models often fail to preserve high-cardinality discrete support (Figure 5 and Table 2).

We study *cardinality* because it determines whether the synthetic table preserves the state space needed by downstream analytical queries. For discrete columns, cardinality measures whether the generated table retains the distinct values present in the real data. This is especially important for high-cardinality attributes, where a column may contain hundreds, thousands, or even more distinct states. If a generative model drops a large portion of these values, then later filtering, grouping, and subgroup queries no longer operate over the same support as the real table. In this case, the synthetic table may appear reasonable under aggregate distributional metrics while still failing to represent the set of values that downstream analyses depend on.

Dataset	Rows	Column Name	Kind	Real	ARF	Bayes	CTGAN	F-Diff*	RTF	T-Flow	T-DDPM*	T-Diff*	TPF*	T-Syn*	TVAE*
c14	300,000	7	discrete	1219	**1217**	64	**1219**	629	**1212**	**1219**	-	**1219**	**1219**	932	1049
c15	600,000	nom_9	discrete	2218	**2216**	**2216**	**2218**	751	2191	**2219**	-	**2217**	**2218**	2203	81
c17	8,809	listed_in	discrete	484	424	421	**474**	239	45	**467**	-	411	**434**	433	324
c17	8,809	date_added	discrete	1634	1366	1375	**1491**	612	48	**1562**	-	1355	**1459**	1016	924
c18	129,975	region_1	discrete	1179	**1114**	255	**848**	293	**1097**	50	-	-	-	-	-
c18	129,975	title	discrete	96777	**62598**	242	19630	1022	**61865**	**63708**	-	-	-	-	-
c18	129,975	winery	discrete	15786	**13729**	256	8207	915	**13594**	**15730**	-	-	-	-	-
c3	2,551	ATRINS-DONOR-521	discrete	2541	**1649**	1572	1581	-	150	**1609**	-	1579	1535	**2550**	83
c3	2,551	CCAGC	discrete	2426	1533	**1562**	**1559**	-	146	**1564**	-	1532	1547	1462	84
c19	48,697	channel_title	discrete	2181	2086	256	**2163**	607	59	**2181**	-	1834	**2168**	1699	1774

Table 2: Representative column-level cases for cardinality fidelity. Each generative-model entry is a distinct-value count. Entries for generative models are highlighted as **First**, **Second**, and **Third** within each row by closeness to the real statistics. Models marked with * have unavailable results due to distinct-value limitations or exceeding the 72-hour time limit for training or generation.

Figure 5 shows that this failure is common on high-cardinality discrete columns. Many points fall far below this line, especially when the real column contains thousands to tens of thousands of distinct values. This means that the synthetic table often preserves only a fraction of the real discrete support. The problem is also generative-model-dependent: some generative models remain close to the diagonal for several columns, while others collapse high-cardinality support to only hundreds of synthetic values.

Table 2 shows the support loss at the raw column level. For columns with relatively moderate cardinality, some generative models can preserve the distinct count almost exactly. Several generative models collapse the same columns to only a small fraction of the real support: BayesNet repeatedly generates only 64 or 256 distinct values, ForestDiffusion often reduces thousands of real states to hundreds, and TVAE can collapse high-cardinality columns even more severely.

Finding 3: Current SOTA generative models preserve global conditional structure more reliably than local-slice conditional structure (Figure 6).

For the local–global comparison, we define a conditional query containing a `WHERE` predicate as a *local* query. Its paired *global* counterpart is obtained by deleting only the `WHERE` clause, while keeping the remaining `SELECT`, `FROM`, `GROUP BY`, and `ORDER BY` structure unchanged. For example, `... WHERE Subtopic = 'Linear Transformations' GROUP BY ...` is paired with the same query without the `WHERE` clause. Thus, the comparison isolates the effect of evaluating the same conditional structure within a filtered slice rather than over the full table; see Appendix C.4.2 for the formal construction.

Figure 6 shows that local-slice conditional queries are often less faithfully preserved than their global counterparts. Each point represents one generative model on one dataset under one paired conditional query. The points below the diagonal correspond to cases where the same query structure becomes less faithful after restricting the analysis to a subpopulation. This result shows that high global conditional fidelity is not sufficient: a synthetic table may answer broad conditional queries accurately while still failing on the local slices that analysts use to study specific subgroups.

Finding 4: Current SOTA generative models fail to preserve rare cases and tail distributions, which leads to poor fidelity on tail-focused analytical queries (Figure 7).

We first ask whether current generative models preserve the rare regions of individual columns, before evaluating full tail queries. The tail-region definition and the two complementary scores, *tail size* and *tail coverage*, are documented in Appendix C.4.3.

Figure 7 shows that current generative models have limited ability to preserve rare regions, and this limitation becomes more severe as the tail threshold tightens. Figure 7 shows that this trend is not caused by a single weak generative model. However, the rare-region failure is broadly shared across current SOTA generative models.

Missingness. Missingness provides a useful contrast. Generative models often preserve the overall missing rate, but this does not guarantee that missingness remains associated with the same conditioning attributes. We therefore evaluate both marginal missingness and broad co-missingness. For example, one grounded query on dataset `c5` groups rows by `odor` and measures the missing rate of

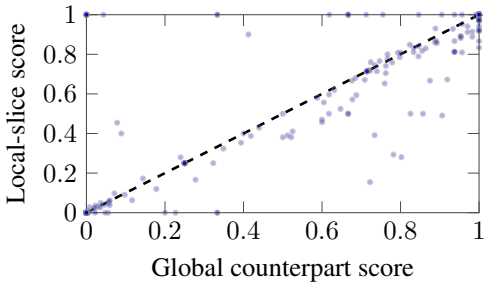


Figure 6: Local-slice vs. global-counterpart conditional scores. Most points lie below the diagonal, indicating that local-slice conditions are generally harder to preserve than global-counterpart conditions. Details appear in Appendix C.4.2.

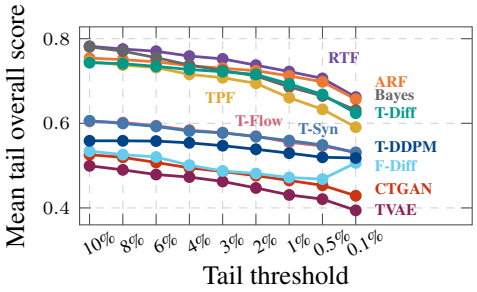


Figure 7: The broadly consistent decline across models shows that tail degradation is a general pattern, not one driven by only a few models.

Category	Overall	Subgroup		Conditional		Tail / Rarity		Missingness		Cardinality / Range	
	Query overall	Structure	Size	Global structure	Local slice	Tail coverage	Tail size	Marginal	Broad co-missing	Discrete	Continuous
Categorical	**0.68** ±0.07	**0.77** ±0.06	**0.78** ±0.06	**0.64** ±0.06	**0.63** ±0.10	**0.14** ±0.12	**0.31** ±0.24	**0.93** ±0.01	**0.93** ±0.01	**0.66** ±0.21	**0.88** ±0.16
Numerical	**0.59** ±0.06	**0.52** ±0.08	**0.54** ±0.10	**0.32** ±0.06	**0.15** ±0.09	**0.07** ±0.07	**0.60** ±0.11	**0.99** ±0.01	**0.99** ±0.01	**0.58** ±0.25	**0.92** ±0.08
Mix	**0.63** ±0.06	**0.55** ±0.05	**0.63** ±0.05	**0.51** ±0.06	**0.56** ±0.09	**0.10** ±0.12	**0.23** ±0.18	**0.94** ±0.02	**0.94** ±0.02	**0.85** ±0.14	**0.94** ±0.08

Table 3: **Query taxonomy by data regime.** Rows are ranked as **First**, **Second** and Third.

`stalk-root` within each group. A synthetic table may match the global missing fraction while still failing this grouped query if the relationship between the conditioning attribute and the missingness pattern is distorted. Detailed query definitions are provided in Appendix C.4.4.

4.3.2 Breaking Down by Categorical vs. Numerical vs. Mixed

We further separate results across categorical, numerical, and mixed datasets, which we define by the dominant column type in each schema. This analysis tests whether the same query family fails differently depending on the schema regime, rather than treating all tabular datasets as one homogeneous benchmark pool.

Table 3 shows that queries on categorical datasets are often about preserving discrete support, subgroup identity, and rare-state membership, whereas queries on numerical datasets more often test range structure, continuous conditional behavior, and tail sizes.

4.4 Fidelity-Cost Pareto Tradeoff

Finding 5: There is a clear cost-fidelity tradeoff in tabular data generation, with different tradeoff regimes for training and generation cost (Figures 8 and Table 18).

Figure 8 shows that query-centric fidelity and runtime separate into distinct practical regimes once training and generation cost are considered separately. RealTabFormer achieves the highest mean Query overall score across the 49-dataset benchmark, but it is also by far the most expensive generative model to train. BayesNet lies on the low-training-cost/strong-fidelity end of the training frontier, while ARF offers a stronger generation-time tradeoff because it has high query fidelity and sub-second median generation cost per 1k rows. Details are shown in Appendix Table 18.

4.5 The Stability of Query Generation Pipeline

We assess whether the choice of LLMs used for SQL realization affects the benchmark's comparative conclusions. We regenerate the query sets using three AI builders, including Claude Haiku-4.5, GLM-5.3 and GLM-5.3Flash, running three times for the two GLM models, under the same template library and dataset schemas. Figure 9 shows that the same four generative models occupy the top four positions across all seven query sets. The overall query-centric fidelity score, averaged across

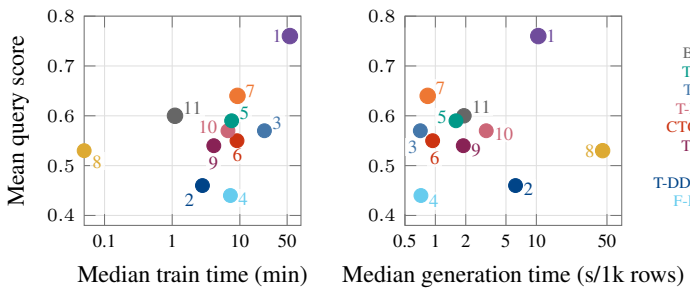


Figure 8: Query-centric fidelity versus runtime. The left panel shows median training time on a log scale, and the middle panel shows median generation time per 1k rows. Numbers mean different models align with Figure 9. Details in Table 18.

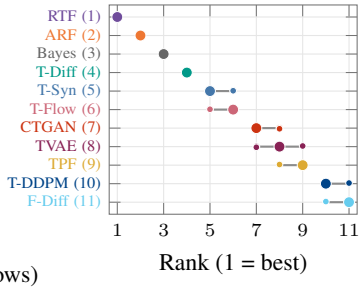


Figure 9: Generator ranks across seven AI workload builders. Segments show min–max rank; large dots show medians.

generative models, ranges from 0.572 to 0.588, an absolute spread of 0.016. These results indicate that the identification of the leading generative models is insensitive to the choice of SQL builder within the tested configurations. Complete scores are reported in Appendix Table 20.

4.6 Structural Coverage of the Generated Query Workload

Beyond ranking stability, we audit whether the complete generated workload covers the structural space represented by public analytical SQL workloads. We retain all 2,763 valid generated queries in the default query version. After schema-specific normalization and SQL parsing with SQLGlot (Mao & SQLGlot Contributors, 2026), these queries contain 44 unique query shapes, while the source corpus contains 286 unique shapes from 320 valid SQL queries. For each source shape, we find its closest generated shape using Tree Similarity of Edit Distance (TSED) (Song et al., 2024), an AST-based metric that normalizes tree-edit distance into a similarity score in [0, 1], with higher values indicating greater structural similarity. The resulting Generated→Source TSED score is 0.702, indicating substantial structural overlap between the generated and public workloads, while still leaving non-trivial source-side structures uncovered.

Complementing this structural view, we compare the query-score distributions of the official ClickBench workload ($n = 39$) and TQBClick ($n = 144$). We first normalize query scores to $[0, 1]$ and express query ranks as percentiles, making the two workloads comparable despite their different numbers of queries. We then compute the normalized 1-Wasserstein distance as the area of the absolute difference between their empirical quantile functions over the full percentile range. The resulting distance is 0.1702, corresponding to an average normalized-score separation of 17.02% points across ranked quantiles. More details lie in Appendix G.2.2.

5 Conclusions

TabQueryBench reframes synthetic tabular data evaluation around analytical query answers. Instead of asking only whether a synthetic table matches the real table under aggregate statistical distances, it asks whether the table preserves the answers to queries that downstream users are likely to run. We instantiate this idea with 49 datasets and 11 generative models. More concretely, TabQueryBench contributes a workload-grounded evaluation framework that distills recurring analytical structures into 44 reusable templates across five query families, automatically grounds them to diverse dataset schemas, and evaluates synthetic data through the resulting query answers at both query and family levels.

Our evaluation shows that distance-based fidelity does not reliably predict query-centric fidelity. Beyond an aggregate score, TabQueryBench localizes where synthetic data fails to preserve analytical behavior, revealing failures in high-cardinality support, local conditional structure, and extreme tails. In our released repository, users can also invoke the TabQueryBench API following the provided requirements to automatically generate an executable query workload for a target dataset. This enables workload-aware evaluation, model selection, and failure diagnosis beyond a single global resemblance score.

AI USE STATEMENT

During research and manuscript preparation, we used LLM-based assistants and coding agents to assist with software implementation and debugging, language editing and writing refinement, and the preparation or refinement of selected conceptual figures. These tools were used as assistants rather than as autonomous decision makers: all generated code, text, figures, and technical suggestions were reviewed and, where necessary, revised by the authors.

AI is also used as a constrained component of the benchmark construction pipeline. Within the benchmark, an LLM agent receives a dataset profile and a fixed template library, selects applicable templates and dataset columns, and produces the structured query specification under predefined constraints. The LLM does not freely generate the final SQL. The specification is validated against the template contract and deterministically rendered into executable SQL; invalid, non-executable, or contract-violating outputs are rejected and regenerated under the same constraints. Profiling, binding, validation, SQL rendering, scoring, and benchmark aggregation are deterministic and are not delegated to the LLM.

To evaluate the robustness and external validity of the agent-grounded query generation pipeline, we conduct three dedicated analyses. First, we run the same grounding pipeline with multiple AI models to examine whether its behavior and downstream conclusions are robust to the choice of underlying model. Second, we apply queries produced by our agent to ClickBench and compare their workload characteristics with those of the real analytical SQL workload provided by ClickBench. Third, we measure structural similarity between our generated queries and all available SQL queries collected from the 12 source workloads using Tree Similarity of Edit Distance (TSED), providing an external assessment of how closely the generated workload reflects the structural space of real-world analytical SQL.

ETHICS STATEMENT

This study evaluates synthetic tabular-data generators using publicly available datasets and does not recruit human participants or collect new personal data. The benchmark is intended to measure analytical fidelity and does not establish privacy protection. Synthetic data that achieves high fidelity may still retain sensitive or identifying information, so benchmark scores should not be interpreted as evidence of privacy, anonymity, or regulatory compliance. Users of the released artifacts should follow the licenses, terms, and usage restrictions associated with the original data sources and should conduct separate privacy and risk assessments before applying the benchmark or its outputs to sensitive data.

REPRODUCIBILITY STATEMENT

We provide the benchmark code, query-template specifications, grounding and validation procedures, model adapters, experiment configurations, and the scripts and data used to generate the reported figures and tables in the accompanying anonymous repository: https://github.com/TabQueryBench/TabQueryBench and https://huggingface.co/datasets/TabQueryBench2026/TabQueryBench. The evaluation protocol specifies the dataset splits, model interfaces, sampling budgets, query-grounding constraints, scoring rules, runtime measurements, and aggregation procedures. Each generator is isolated in a Docker environment and evaluated through a common fit–sample–restore–validate interface. We record random seeds where supported, together with model versions, prompts, inference settings, and other configurations needed to reproduce the experiments. The appendix documents the dataset roster, model settings, API usage, query-generation protocol, and scoring standard required to reproduce the reported results.

REFERENCES

Altinity. ClickHouse TPC-DS benchmark. https://github.com/Altinity/tpc-ds. URL https://github.com/Altinity/tpc-ds.

Apache Druid. *All Druid SQL Functions*. URL https://druid.apache.org/docs/latest/querying/sql-functions/.

Andrew G. Bates, Iva Špakulová, Iain Dove, and Andrew Mealor. ONS Methodology Working Paper Series Number 16: Synthetic Data Pilot. Technical report, Office for National Statistics, January 2019. URL https://www.ons.gov.uk/methodology/methodologicalpublications/generalmethodology/onsworkingpaperseries/onsmethodologyworkingpaperseriesnumber16syntheticdatapilot. Published 15 January 2019.

Hasan Alp Caferoğlu, Mehmet Serhat Çelik, and Özgür Ulusoy. SING-SQL: A Synthetic Data Generation Framework for In-Domain Text-to-SQL Translation, 2025. URL https://arxiv.org/abs/2509.25672.

Chao Chen, Jamie Twycross, and Jonathan M. Garibaldi. A new accuracy measure based on bounded relative error for time series forecasting. *PLOS ONE*, 12(3):e0174202, 2017. doi: 10.1371/journal.pone.0174202.

Andrew Chu, Xi Jiang, Shinan Liu, Arjun Bhagoji, Francesco Bronzino, Paul Schmitt, and Nick Feamster. Feasibility of state space models for network traffic generation. In *Proceedings of the 2024 SIGCOMM Workshop on Networks for AI Computing*, pp. 9–17, 2024.

Andrew Chu, Xi Jiang, Shinan Liu, Arjun Bhagoji, Francesco Bronzino, Paul Schmitt, and Nick Feamster. Netssm: Multi-flow and state-aware network trace generation using state-space models. *Proceedings of the ACM on Networking*, 4(CoNEXT1):6:1–6:24, 2026. doi: 10.1145/3786289.

ClickHouse. *Aggregate Functions*. URL https://clickhouse.com/docs/sql-reference/aggregate-functions/reference.

ClickHouse. ClickBench: A benchmark for analytical databases. https://github.com/ClickHouse/ClickBench, 2022. URL https://github.com/ClickHouse/ClickBench. GitHub benchmark.

CWI Database Architectures Group. Reproducing TPC-DS qualification results. https://github.com/cwida/tpcds-result-reproduction. URL https://github.com/cwida/tpcds-result-reproduction.

DataCebo. Welcome to SDGym!, n.d. URL https://docs.sdv.dev/sdgym. Official documentation; accessed 26 September 2026.

Bailu Ding, Surajit Chaudhuri, Johannes Gehrke, and Vivek R. Narasayya. DSB: A decision support benchmark for workload-driven and traditional database systems. *Proceedings of the VLDB Endowment*, 14(13):3376–3388, 2021. doi: 10.14778/3484224.3484234.

Khaled El Emam, Lucy Mosquera, Xi Fang, and Alaa El-Hussuna. Utility metrics for evaluating synthetic health data generation methods: Validation study. *JMIR Medical Informatics*, 10(4):e35734, 2022. doi: 10.2196/35734.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0519. URL https://openreview.net/forum?id=jZqCqpCLdU. Datasets and Benchmarks Track.

Alison L. Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *International Statistical Review*, 70(3):419–435, 2002. doi: 10.1111/j.1751-5823.2002.tb00178.x.

Google Cloud. *GoogleSQL Approximate Aggregate Functions*. URL https://cloud.google.com/bigquery/docs/reference/standard-sql/approximate_aggregate_functions.

Ragini Gupta, Shinan Liu, Ruixiao Zhang, Xinyue Hu, Xiaoyang Wang, Hadjer Benkraouda, Pranav Kommaraju, Phuong Cao, Nick Feamster, and Klara Nahrstedt. Generative active adaptation for drifting and imbalanced network intrusion detection. *arXiv preprint arXiv:2503.03022*, 2025.

Andrés Guzmán-Cordero, Floor Eijkelboom, and Jan-Willem van de Meent. Exponential family variational flow matching for tabular data generation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=kjtvCSkSsy.

H2O.ai. Database-like operations benchmark. https://h2oai.github.io/db-benchmark/, 2020. URL https://h2oai.github.io/db-benchmark/. Benchmark report and reproducibility repository.

Lasse Hansen, Nabeel Seedat, Mihaela van der Schaar, and Andrija Petrovic. Reimagining synthetic tabular data generation through data-centric ai: A comprehensive benchmark. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/6aa9a05b929fb08ff46a58cab6cf860d-Abstract-Datasets_and_Benchmarks.html. Datasets and Benchmarks Track.

Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic tabular data evaluation in the health domain covering resemblance, utility, and privacy dimensions. *Methods of Information in Medicine*, 62(S 01):e19–e38, 2023. doi: 10.1055/s-0042-1760247. URL https://doi.org/10.1055/s-0042-1760247.

Julian Höllig and Michaela Geierhos. Utility meets privacy: A critical evaluation of tabular data synthesizers. *IEEE Access*, 13:44497–44509, 2025. doi: 10.1109/ACCESS.2025.3549680.

Valter Hudovernik, Martin Jurković, and Erik Štrumbelj. Benchmarking the fidelity and utility of synthetic relational data. *arXiv preprint arXiv:2410.03411*, 2024. doi: 10.48550/arXiv.2410.03411. URL https://arxiv.org/abs/2410.03411.

Hugging Face. Hugging Face Datasets. https://huggingface.co/datasets. URL https://huggingface.co/datasets. Accessed: 2026-07-01.

Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Paul Schmitt, Francesco Bronzino, and Nick Feamster. Generative, high-fidelity network traces. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*, pp. 131–138, 2023.

Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. Netdiffusion: Network data augmentation through protocol-constrained traffic generation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 8(1):1–32, 2024.

Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. Tabstruct: Measuring structural fidelity of tabular data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XOPH34Extq. Oral presentation.

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 1288–1296. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/jolicoeur-martineau24a.html.

Bayrem Kaabachi, Jérémie Despraz, Thierry Meurers, Karen Otte, Mehmed Halilovic, Bogdan Kulynych, Fabian Prasser, and Jean Louis Raisaro. A scoping review of privacy and utility metrics in medical synthetic data. *npj Digital Medicine*, 8(1):60, 2025. doi: 10.1038/s41746-024-01359-3.

Kaggle. Kaggle. https://www.kaggle.com. URL https://www.kaggle.com. Accessed: 2026-07-01.

Shivasankari Kannan, Yeounoh Chung, Amita Gondi, Tristan Swadell, and Fatma Özcan. High-fidelity and complex test data generation for google SQL code generation services. In *2026 IEEE 42nd International Conference on Data Engineering (ICDE)*, pp. 3694–3707, 2026. doi: 10.1109/ICDE65706.2026.00274.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI Machine Learning Repository. https://archive.ics.uci.edu. URL https://archive.ics.uci.edu. Accessed: 2026-07-01.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17564–17579, 2023. URL https://proceedings.mlr.press/v202/kotelnikov23a.html.

Anton D Lautrup, Tobias Hyrup, Arthur Zimek, and Peter Schneider-Kamp. Syntheval: a framework for detailed utility and privacy evaluation of tabular synthetic data. *Data Mining and Knowledge Discovery*, 39(1):6, 2025. doi: 10.1007/s10618-024-01081-4.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115.

Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. TabPFGen – tabular data generation with tabPFN. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023. URL https://openreview.net/forum?id=4MkkNsAEmO.

Spyros Makridakis. Accuracy measures: Theoretical and practical concerns. *International Journal of Forecasting*, 9(4):527–529, 1993. doi: 10.1016/0169-2070(93)90079-3.

Toby Mao and SQLGlot Contributors. SQLGlot: Python SQL parser and transpiler, 2026. URL https://github.com/tobymao/sqlglot. Software, version 30.18.0.

Jr. Massey, Frank J. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. doi: 10.1080/01621459.1951.10500769.

Raghunath Othayoth Nambiar and Meikel Poess. The making of TPC-DS. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pp. 1049–1058. VLDB Endowment, 2006.

OpenML. OpenML. https://www.openml.org. URL https://www.openml.org. Accessed: 2026-07-01.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics*, pp. 399–410, 2016. doi: 10.1109/DSAA.2016.49.

Personal Data Protection Commission Singapore. Proposed Guide on Synthetic Data Generation. Guidance document, July 2024. URL https://www.pdpc.gov.sg/organisations/resources/guidance-by-topic/proposed-guide-on-synthetic-data-generation. Published 15 July 2024.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019. doi: 10.1561/2200000073.

Haoyue Ping, Julia Stoyanovich, and Bill Howe. DataSynthesizer: Privacy-preserving synthetic datasets. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, SSDBM '17, pp. 42:1–42:5, 2017. doi: 10.1145/3085504.3091117. URL https://doi.org/10.1145/3085504.3091117.

Meikel Poess and Chris Floyd. New TPC benchmarks for decision support and web commerce. *ACM SIGMOD Record*, 29(4):64–71, 2000. doi: 10.1145/369275.369291.

Meikel Poess, Bryan Smith, Lubor Kollár, and Per-Åke Larson. TPC-DS, taking decision support benchmarking to the next level. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 582–587, 2002. doi: 10.1145/564691.564759.

Meikel Poess, Raghunath Othayoth Nambiar, and David Walrath. Why you should run TPC-DS: A workload analysis. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 1138–1149, 2007. URL https://www.vldb.org/conf/2007/papers/industrial/p1138-poess.pdf.

Zhaozhi Qian, Rob Davis, and Mihaela van der Schaar. Synthcity: A benchmark framework for diverse use cases of tabular synthetic data. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/09723c9f291f6056fd1885081859c186-Abstract-Datasets_and_Benchmarks.html. Datasets and Benchmarks Track.

Gabriele Santangelo, Giovanna Nicora, Riccardo Bellazzi, and Arianna Dagliati. How good is your synthetic data? SynthRO, a dashboard to evaluate and benchmark synthetic tabular data. *BMC Medical Informatics and Decision Making*, 25(1):89, 2025. doi: 10.1186/s12911-024-02731-9. URL https://doi.org/10.1186/s12911-024-02731-9.

Aniruddha Sen, Christine Task, Dhruv Kapur, Gary Howarth, and Karan Bhagat. Diverse community data for benchmarking data privacy algorithms. *Advances in Neural Information Processing Systems*, 36:51409–51420, 2023.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. TabDiff: a mixed-type diffusion model for tabular data generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=swvURjrt8z.

Andrey Sidorenko, Michael Platzer, Mario Scriminaci, and Paul Tiwald. Benchmarking synthetic tabular data: A multi-dimensional evaluation framework. *arXiv preprint arXiv:2504.01908*, 2025. doi: 10.48550/arXiv.2504.01908. URL https://arxiv.org/abs/2504.01908.

Joshua Snoke, Gillian M. Raab, Beata Nowok, Chris Dibben, and Aleksandra Slavković. General and specific utility measures for synthetic data. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 181(3):663–688, 2018. doi: 10.1111/rssa.12358. URL https://doi.org/10.1111/rssa.12358.

Snowflake Inc. PERCENTILE_CONT — Snowflake Documentation. https://docs.snowflake.com/en/sql-reference/functions/percentile_cont, n.d.a. URL https://docs.snowflake.com/en/sql-reference/functions/percentile_cont. Accessed 26 September 2026.

Snowflake Inc. WIDTH_BUCKET — Snowflake Documentation. https://docs.snowflake.com/en/sql-reference/functions/width_bucket, n.d.b. URL https://docs.snowflake.com/en/sql-reference/functions/width_bucket. Accessed 26 September 2026.

Aivin V. Solatorio and Olivier Dupriez. REaLTabFormer: Generating realistic relational and tabular data using transformers. *arXiv preprint arXiv:2302.02041*, 2023. doi: 10.48550/arXiv.2302.02041. URL https://arxiv.org/abs/2302.02041.

Yewei Song, Cedric Lothritz, Daniel Tang, Tegawendé F. Bissyandé, and Jacques Klein. Revisiting code similarity evaluation with abstract syntax tree edit distance. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 38–46, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-short.3. URL https://aclanthology.org/2024.acl-short.3/.

Marika Swanberg, Ryan McKenna, Edo Roth, Albert Cheu, and Peter Kairouz. Is API access to LLMs useful for generating private synthetic tabular data? *arXiv preprint arXiv:2502.06555*, 2025. doi: 10.48550/arXiv.2502.06555. URL https://arxiv.org/abs/2502.06555.

Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Benchmarking differentially private synthetic data generation algorithms. *arXiv preprint arXiv:2112.09238*, 2021. doi: 10.48550/arXiv.2112.09238. URL https://arxiv.org/abs/2112.09238.

Christine Task, Karan Bhagat, and Gary Howarth. SDNist v2: Deidentified Data Report Tool, March 2023. URL https://data.nist.gov/od/id/mds2-2943.

Timescale. RTABench: A benchmark for real-time analytics applications. https://github.com/timescale/rtabench, 2025. URL https://github.com/timescale/rtabench. GitHub benchmark.

Van Tran, Shinan Liu, Tian Li, and Nick Feamster. Quantifying the privacy implications of high-fidelity synthetic network traffic. *arXiv preprint arXiv:2511.20497*, 2025. doi: 10.48550/arXiv.2511.20497. URL https://arxiv.org/abs/2511.20497. Accepted to ACM Internet Measurement Conference (IMC) 2026.

Transaction Processing Performance Council. TPC Benchmark H (TPC-H) standard specification, 1999. URL https://www.tpc.org/tpch/. TPC-H benchmark specification and query generator.

Transaction Processing Performance Council. TPC Benchmark DS (TPC-DS) standard specification, 2017. URL https://www.tpc.org/tpcds/.

Trino Software Foundation. Aggregate Functions — Trino Documentation. https://trino.io/docs/current/functions/aggregate.html, n.d. URL https://trino.io/docs/current/functions/aggregate.html. Accessed 26 September 2026.

Boris van Breugel, Zhaozhi Qian, and Mihaela van der Schaar. Synthetic data, real errors: How (not) to publish and use synthetic data. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 34793–34808, 2023. URL https://proceedings.mlr.press/v202/van-breugel23a.html.

Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An Approach, Method, and Software Mechanism for Generating Synthetic Patients and the Synthetic Electronic Health Care Record. *Journal of the American Medical Informatics Association*, 25(3):230–238, 2018. doi: 10.1093/jamia/ocx079.

Xinyue Wang, Hafiz Asif, and Jaideep Vaidya. Preserving missing data distribution in synthetic data. In *Proceedings of the ACM Web Conference 2023*, pp. 2110–2121, 2023. doi: 10.1145/3543507.3583297.

David S. Watson, Kristin Blesch, Jan Kapar, and Marvin N. Wright. Adversarial random forests for density estimation and generative modeling. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5357–5375, 2023. URL https://proceedings.mlr.press/v206/watson23a.html.

William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems*, 28(4):20:1–20:38, 2010. doi: 10.1145/1852102.1852106.

Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30:79–82, 2005. doi: 10.3354/cr030079.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html.

Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4Ay23yeuz0.

Appendix

TABQUERYBENCH: A Query-Centric Benchmark for Synthetic Tabular Data

Table of Contents

A Related Works . . . 17
A.1 Tabular Generative Models . . . 17
A.2 Tabular Generative Model Benchmarks . . . 17
A.3 Database Query Benchmarks . . . 18
A.4 Distance-Based vs. Query-Centric Fidelity . . . 19
B Synthetic Data Experiment . . . 20
B.1 Benchmark Scope . . . 20
B.2 Experimental Settings Details . . . 20
B.3 Dataset Catalog . . . 23
B.4 Models Catalog . . . 24
C Query Templates and Five Query Families . . . 25
C.1 Query-Centric Fidelity Overview . . . 25
C.2 Workload Sources . . . 25
C.3 44-Template Taxonomy Catalog . . . 25
C.4 Five Query Families . . . 26
C.4.1 Subgroup . . . 26
C.4.2 Conditional . . . 27
C.4.3 Tail / Rarity . . . 28
C.4.4 Missingness . . . 30
C.4.5 Cardinality / Range . . . 31
D Query Grounding . . . 32
D.1 Pipeline Overview . . . 32
D.2 Stage 1: Template Construction . . . 32
D.3 Stage 2: Dataset Grounding and SQL Realization . . . 32
D.4 Agent API Usage by Dataset . . . 33
E Query Scoring Standard . . . 36
E.1 Reporting and Applicability Rules . . . 36
E.2 Classical Distance-Based Scoring Rules . . . 37
F Overall Evaluation Analysis . . . 39
F.1 Overall Benchmark Summary . . . 39
F.2 Overall Query Score Heatmap . . . 39
G Extended Experiments . . . 40
G.1 Stability Across Query-Workload Rebuilds . . . 40
G.2 Structural Coverage of the Full Generated Workload . . . 40
H Discussion and Future Works . . . 43

A Related Works

This appendix expands the three strands of related work introduced in Section 2.1: tabular generative models, their evaluation benchmarks, and database query benchmarks. These strands distinguish how synthetic tables are generated, which properties are evaluated, and how analytical queries can provide reusable evaluation tasks.

A.1 Tabular Generative Models

Tabular synthesis models records whose columns may have different types and semantics. Preserving plausible values in each column is one part of the problem; maintaining dependencies between columns and support for less frequent values is also relevant to downstream use. Different generative approaches encode these relationships through different model structures. Their architectural differences motivate evaluation across multiple analytical properties rather than reliance on one summary score.

Bayesian and tree-based approaches. DataSynthesizer uses a Bayesian-network representation to model dependencies between attributes (Ping et al., 2017). Adversarial Random Forests (ARF) instead use tree-based partitions for density estimation and synthetic-data generation (Watson et al., 2023). These approaches provide alternatives to neural synthesis and are useful comparators when assessing whether different representations preserve grouped aggregates and conditional structure.

Adversarial and variational models. CTGAN and TVAE adapt adversarial and variational learning to mixed-type tabular data (Xu et al., 2019). CTGAN uses conditional generation, whereas TVAE learns a latent-variable representation. Both are established baselines for tabular synthesis. Our evaluation concerns the analytical answers supported by their outputs, rather than their training objectives.

Diffusion, transformer-based, and flow-matching models. TabDDPM, TabSyn, and TabDiff apply diffusion-based generation to tabular data, with different representations of mixed-type variables (Kotelnikov et al., 2023; Zhang et al., 2024; Shi et al., 2025). ForestDiffusion uses tree-based components in diffusion and flow-based generative modeling (Jolicoeur-Martineau et al., 2024). REaLTabFormer uses transformer-based generation, while TabPFGen explores generation using pretrained tabular foundation models (Solatorio & Dupriez, 2023; Ma et al., 2023). TabbyFlow provides a flow-matching approach to tabular synthesis (Guzmán-Cordero et al., 2025). These mechanisms need not preserve the same analytical properties to the same degree. TabQueryBench therefore evaluates their generated tables through a common set of dataset-grounded analytical queries.

A.2 Tabular Generative Model Benchmarks

Evaluation frameworks and standardized comparisons. Synthcity and SynthEval provide reusable tools for evaluating synthetic data through collections of metrics (Qian et al., 2023; Lautrup et al., 2025). The data-centric evaluation of Hansen et al. and SDGym support comparisons of tabular synthesis methods across datasets (Hansen et al., 2023; DataCebo, n.d.). Such resources make distributional fidelity, predictive utility, privacy, or computational cost easier to assess systematically. These dimensions answer different questions: distributional metrics compare statistical properties, predictive utility measures usefulness for specified learning tasks, and privacy evaluation concerns information exposure. None of these concepts should be treated as interchangeable with the preservation of a particular analytical query answer.

Specialized evaluation settings. SDNist and the associated Diverse Community Data resource support evaluation in public data-release scenarios (Sen et al., 2023; Task et al., 2023). SyntheRela addresses multi-table synthesis, while SynthRO focuses on health-oriented synthetic-data validation (Hudovernik et al., 2024; Santangelo et al., 2025). Sidorenko et al. consider multidimensional evaluation of mixed and contextual data, and Tao et al. benchmark differentially private tabular-data generation (Sidorenko et al., 2025; Tao et al., 2021). TabArena is an adjacent benchmark for tabular predictive learning rather than an evaluation of synthetic-table generators (Erickson et al., 2025). The interpretation of each comparison must therefore account for its target setting and evaluation object.

Structural fidelity and TabStruct. TabStruct explicitly adds structural fidelity to conventional evaluation dimensions and studies 13 tabular generative models across 29 datasets (Jiang et al., 2026). When a ground-truth structural causal model is available, it evaluates the preservation of implied conditional independence statements. For datasets without this information, its *global utility* metric treats each variable as a prediction target, predicts it from the remaining variables, and aggregates normalized predictive utility across targets. TabStruct reports empirical alignment between global utility and its conditional-independence-based structural scores. This is a broader structural assessment than predictive utility for a single designated target.

Our evaluation addresses a complementary object: the answers returned by dataset-grounded analytical SQL queries. In TABQUERYBENCH, a query family identifies analytical properties such as subgroup summaries, tail coverage, or missingness patterns. It is not a causal graph, and its conditional queries should not be equated with causal conditional independence tests. Likewise, a high query-centric fidelity score does not establish causal-structure recovery. The distinction concerns what is directly measured: prediction across variables and causal structural consistency in TabStruct, versus analytical answer preservation in TABQUERYBENCH.

Table 4: **Evaluation settings and objects of related resources.** Entries summarize the role of each resource, rather than asserting the absence of capabilities not listed here.

Resource	Evaluation setting and object
Synthcity (Qian et al., 2023)	Reusable synthetic-data framework with multiple evaluation metrics.
SynthEval (Lautrup et al., 2025)	Synthetic-data evaluation through a configurable metric suite.
Data-centric benchmark (Hansen et al., 2023)	Comparative evaluation of synthetic tabular data quality.
SDGym (DataCebo, n.d.)	Benchmarking infrastructure for tabular synthesis methods.
SDNist (Sen et al., 2023; Task et al., 2023)	Evaluation associated with public data-release scenarios.
SyntheRela (Hudovernik et al., 2024)	Multi-table synthetic data and relational fidelity.
SynthRO (Santangelo et al., 2025)	Health-oriented synthetic-data validation.
Sidorenko et al. (Sidorenko et al., 2025)	Multidimensional evaluation of mixed and contextual data.
DP benchmark (Tao et al., 2021)	Differentially private tabular synthesis and privacy–utility evaluation.
TabArena (Erickson et al., 2025)	Tabular predictive learning, rather than synthetic-data generation.
TabStruct (Jiang et al., 2026)	Structural fidelity through conditional independence and global utility, alongside conventional dimensions.
TABQUERYBENCH	Analytical SQL answer preservation across five query families, alongside distance-based metrics and cost.

A.3 DATABASE QUERY BENCHMARKS

Parameterized decision-support queries. TPC-H and TPC-DS encode analytical workloads through schemas, parameterized queries, and controlled execution procedures (Poess & Floyd, 2000; Nambiar & Poess, 2006; Poess et al., 2007; Transaction Processing Performance Council, 1999; 2017). Their central role is to evaluate database systems under specified decision-support workloads. For our purposes, their reusable query patterns provide public provenance for analytical operations, including filtering, grouping, aggregation, and ordering.

Public analytical workloads. ClickBench provides a web-analytics dataset and SQL queries for analytical database comparisons (ClickHouse, 2022). H2O db-benchmark compares data-processing systems through operations such as grouping and joins (H2O.ai, 2020). RTABench targets real-time application analytics (Timescale, 2025). These resources offer additional examples of recurring analytical operations. Their original workloads may include relational or system-specific features beyond the single-table setting evaluated in the current TABQUERYBENCH release.

From system workloads to synthetic-data evaluation. TABQUERYBENCH extracts reusable analytical templates from public SQL sources and grounds applicable templates to each dataset. It does not reproduce the complete database benchmarks or their execution protocols. The evaluation asks whether synthetic data preserves the answers induced by these grounded queries. Source attribution and construction details are provided in Appendix D.2; grounding and output-specific scoring are described in Appendices D.3 and E.

A.4 Distance-Based vs. Query-Centric Fidelity

This appendix gives additional context for the motivating ClickBench experiment in Section 2.2. The goal of this experiment is not to replace the full TabQueryBench evaluation, but to test in a realistic analytical workload whether distance-based fidelity and query-centric fidelity expose different behavior. It also checks whether the queries generated by our template-grounding pipeline resemble real analytical queries in terms of their score distribution.

Experimental purpose. We use ClickBench (ClickHouse, 2022) as a real-world analytical workload because it contains public queries written for analytical database evaluation. For each synthetic table, we compare two views of fidelity. The first view is the classical distance-based view, which summarizes global statistical similarity between the real and synthetic tables. The second view is the query-centric view, which asks whether the same analytical SQL queries return similar answers on the real and synthetic tables. The scoring rules for both views follow the definitions in Appendix E, with the distance-based metrics detailed in Appendix E.4.

Official queries and generated queries. We compare the official ClickBench queries with QClick2.4, our generated ClickBench-style query workload produced by the same query-construction pipeline used in TabQueryBench. Because the two workloads contain different numbers of scorable queries, we do not compare them by raw query index. Instead, we sort queries in each workload from high to low by their mean query score across the evaluated synthetic generators and align the two curves by ranked percentile. This lets us compare the score profile of real analytical queries with the score profile of our generated queries under the same synthetic-data evaluation setting.

Interpretation. This experiment serves two roles. First, it illustrates that distance-based fidelity and query-centric fidelity can diverge in a realistic workload: a synthetic table may look close under global distance metrics while still failing to preserve analytical query answers. Second, it provides a sanity check for our generated workload. If QClick2.4 produced only trivial or unrepresentative queries, its ranked score curve would behave very differently from the official ClickBench workload. Instead, comparing the two ranked curves helps assess whether our generated queries occupy a similar analytical difficulty regime while still providing broader coverage.

B Synthetic Data Experiment

B.1 Benchmark Scope

The released benchmark evaluates 11 synthetic tabular generators over 49 datasets. The dataset and model rosters are kept visible in one place below.

B.2 Experimental Settings Details

This appendix records the implementation details behind the compact evaluation setup in Section 4.1. The main text keeps only the common evaluation contract; the items below make the data split, SQL grounding, execution infrastructure, and model-tuning policy auditable.

Data split and synthetic output contract. For each dataset, the preparation pipeline creates disjoint training and test splits with proportions $80\%/20\%$ using a fixed split seed and stratification when the task is classificatory. Each generator observes only the training split and is asked to synthesize a table with the same schema and, by default, the same number of rows as that split. Query fidelity and distance-based fidelity both use the real training table as their reference. The validation and test splits are retained for model-development or future downstream evaluation and are not used in the current scoring pipeline.

Generator suite. The benchmark covers 11 tabular generators spanning Bayesian-network, tree-based, adversarial, variational, diffusion, transformer-based, and flow-matching families. The evaluated generators are BayesNet, ARF, CTGAN, TVAE, TabDDPM, TabSyn, TabDiff, ForestDiffusion, RealTabFormer, TabPFGen, and TabbyFlow. All generators use the same dataset split and output-size requirement.

SQL grounding and validation. For each dataset, TabQueryBench constructs a grounded SQL workload from the fixed template library described in Section 3.3. The agent selects applicable templates and dataset columns. The profiler, binder, and validator are deterministic. The SQL realizer is the only LLM-assisted component: it calls the GLM-5.3 API inside a constrained template-to-SQL realization step, with a mean API cost of 0.78 per dataset. The validator rejects invalid SQL, incompatible realizations, and queries that fail execution; failed realizations are retried under the same fixed template policy.

Runtime and infrastructure. Each generator is wrapped behind a common interface for fitting, sampling, schema restoration, and output validation. We package each generator in its own Docker image so that incompatible dependencies remain isolated while the produced synthetic tables follow

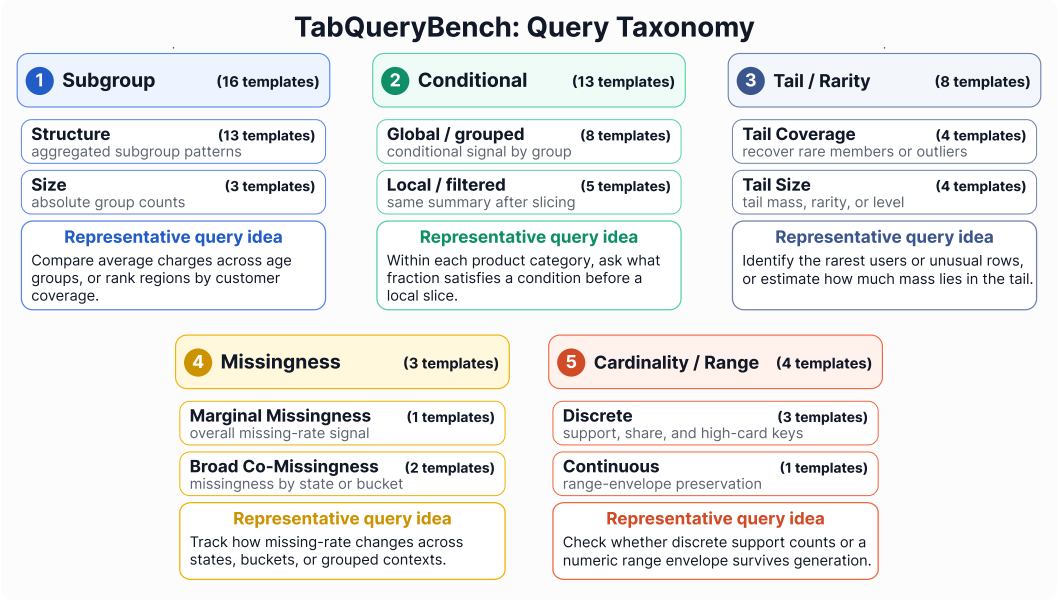


Figure 10: The Query Template Taxonomy.

a shared output contract. Runtime is decomposed into training time and generation time for the quality–cost analysis. The audited runtime experiments use the same hardware platform with an RTX Pro 6000 GPU.

Resolved model configurations. The benchmark runner uses model-specific resolved configurations rather than one common hyperparameter search space. Unless a run records an explicit override, ARF uses 30 trees, 10 adversarial iterations, minimum node size 5, `delta`=0, early stopping, and a fixed seed; BayesNet uses a Chow–Liu structure with mutual-information edge weights, up to 120K fitting rows and 25K structure-learning rows, with discretization bins and categorical-level caps reduced for wide or very large tables. CTGAN uses 50 epochs, embedding dimension 4, $(8, 8)$ generator and discriminator hidden dimensions, batch size 2, `pac`=1, and a 50K training-row cap. TVAE uses 300 epochs, batch size 500, embedding dimension 128, and $(128, 128)$ encoder/decoder dimensions. RealTabFormer uses 100 epochs in the reported default configuration.

For the diffusion and flow-based adapters, ForestDiffusion uses an XGBoost-backed model with 20 estimators, `n_t`=10, `duplicate_K`=5, maximum depth 4, and one worker in the default configuration. TabDDPM uses 1,000 diffusion timesteps, learning rate 10^{-3}, zero dropout and weight decay, and size-dependent training steps, batch size, and hidden-layer width; for example, tables below 10K training rows use 10K steps and $(512, 512)$ hidden layers, while larger tables use larger tiers. TabDiff and TabbyFlow use 500 training steps, learning rate 10^{-3}, batch size 4,096, and a sampling batch size of 10,000. TabDiff uses 50 diffusion timesteps, while TabbyFlow uses the flow-matching configuration with a 100-epoch learning-rate warmup.

TabPFGen uses pretrained TabPFN components without model fitting, caps the fit set at 50K rows, uses 1,000 SGLD steps with step size 0.01 and noise scale 0.01, and generates in memory-bounded chunks. TabSyn trains a VAE followed by latent diffusion on the full training split; its defaults are 4,000 VAE epochs and 10,001 diffusion epochs, with batch sizes adjusted for table width, and 50 function-evaluation steps at sampling time. When a run supplies an explicit epoch override, the TabSyn diffusion epoch count is resolved as approximately 2.5 times the VAE epoch count. These settings are recorded in the run artifacts when available; model-specific safety fallbacks, such as reduced sampling batches after GPU out-of-memory errors or learning-rate halving after numerical divergence, are treated as recovery behavior rather than additional tuning trials.

Preset and override policy. The main benchmark uses the resolved model configurations above. The repository also contains a `fast` preset for separate workload stress tests; it changes selected settings, including TabDiff and TabbyFlow training steps and ForestDiffusion's noise levels, duplicate factor, thread count, and training-row cap. Those stress-test overrides are not used to redefine the main benchmark results. Command-line arguments and explicit environment variables take precedence over preset values, and failed or technically invalid runs remain visible in the benchmark bookkeeping rather than being silently removed.

Subgroup
template S-2.1
dataset c2

QUESTION
Find subgroup counts and rank groups by size.

GROUNDED QUESTION
Which buying groups are most frequent?

GROUNDED SQL
```
SELECT buying [<group_field>],
COUNT(*) AS row_count
FROM c2 [<table>]
GROUP BY buying [<group_field>]
ORDER BY row_count DESC;
```

Conditional
template C-L1
dataset c6

QUESTION
Count rows per (x, y) cell inside a local slice.

GROUNDED QUESTION
Within rows where Subtopic equals Linear Transformations, how many rows fall into each Student Country and Question Level combination?

GROUNDED SQL
```
SELECT "Student Country" [<group_x>],
"Question Level" [<group_y>],
COUNT(*) AS row_count
FROM "c6" [<table>]
WHERE "Subtopic" = 'Linear Transformations'
[<slice_predicate>]
GROUP BY "Student Country" [<group_x>],
"Question Level" [<group_y>]
ORDER BY row_count DESC;
```

Tail / rarity
template T-2.2
dataset m8

QUESTION
Count rows by group inside an upper-tail slice.

GROUNDED QUESTION
Among the top 3% of balance, which previous values occur most often?

GROUNDED SQL
```
SELECT previous [<group_field>],
COUNT(*) AS row_count
FROM m8 [<table>]
WHERE balance > PERCENTILE(balance, 0.97)
[<tail_predicate>]
GROUP BY previous [<group_field>];
```

Missingness
template M-2.1
dataset c5

QUESTION
Compute a target missing rate for each state.

GROUNDED QUESTION
For each odor category, what fraction of rows has missing stalk-root?

GROUNDED SQL
```
SELECT "odor" [<state_col>],
AVG(CASE WHEN "stalk-root" [<target_col>] IS NULL
THEN 1.0 ELSE 0.0 END) AS missing_rate
FROM "c5" [<table>]
GROUP BY "odor" [<state_col>]
ORDER BY missing_rate DESC;
```

Cardinality / range
template K-1.1
dataset c18

QUESTION
Enumerate a high-cardinality attribute by support.

GROUNDED QUESTION
How many rows are associated with each wine title?

GROUNDED SQL
```
SELECT title [<value_col>] AS category_value,
COUNT(*) AS support
FROM c18 [<table>]
WHERE title [<value_col>] IS NOT NULL
GROUP BY title [<value_col>]
ORDER BY support DESC, category_value;
```

Table 5: **One question and SQL example for each of the five query families.** The blue text marks dataset-specific grounding, and the orange notes show which template slot each grounded SQL fragment instantiates. Details in Table 9.

B.3 Dataset Catalog

Table 6: 49 dataset catalog.

Series	No.	Original dataset name	Rows	Cols	Original source URL
C	c1	Jungle Chess 2pcs Raw Endgame Complete	44819	7	https://www.openml.org/d/41027
C	c2	Car Evaluation	1728	7	https://archive.ics.uci.edu/dataset/19/car+evaluation
C	c3	Splice junction Gene Sequences	3189	3	https://archive.ics.uci.edu/dataset/69/molecular+biology+splice+junction+gene+sequences
C	c4	Chess King Rook Vs King Pawn	3196	37	https://archive.ics.uci.edu/dataset/22/chess+king+rook+vs+king+pawn
C	c5	Mushroom	8416	23	https://archive.ics.uci.edu/dataset/73/mushroom
C	c6	Dataset For Assessing ML In Higher Education	9546	8	https://archive.ics.uci.edu/dataset/1031/dataset+for+assessing+mathematics+learning+in+higher+education
C	c7	Nursery	12960	9	https://www.openml.org/d/26
C	c8	Phishing Websites	11055	31	https://archive.ics.uci.edu/dataset/327/phishing+websites
C	c9	Amazon Employee Access Challenge	32769	10	https://www.kaggle.com/c/amazon-employee-access-challenge
C	c10	Poker Hand	1025010	11	https://archive.ics.uci.edu/dataset/158/poker+hand
C	c11	Connect 4	67557	43	https://archive.ics.uci.edu/dataset/26/connect-4
C	c12	Internet-Advertisements	3279	1559	https://www.openml.org/d/40978
C	c13	Us Census Data 1990	2458285	69	https://archive.ics.uci.edu/dataset/116/us+census+data+1990
C	c14	Cat In The Dat	300000	25	https://www.kaggle.com/c/cat-in-the-dat
C	c15	Cat In The Dat Ii	600000	25	https://www.kaggle.com/c/cat-in-the-dat-ii
C	c16	Fivethirtyeight Comic Characters Dataset	6896	13	https://www.kaggle.com/datasets/fivethirtyeight/fivethirtyeight-comic-characters-dataset
C	c17	Netflix Shows	8807	12	https://www.kaggle.com/datasets/shivamb/netflix-shows
C	c18	Wine Reviews	129971	14	https://www.kaggle.com/datasets/zynicide/wine-reviews
C	c19	Trending YouTube Video Statistics and Comments	40949	16	https://www.kaggle.com/datasets/datasnaek/youtube-new
M	m1	Remote Worker Productivity	1500	30	https://huggingface.co/datasets/nprak26/remote-worker-productivity
M	m2	Car Insurance Claim Prediction	58592	44	https://www.kaggle.com/datasets/ifteshanajnin/carinsuranceclaimprediction-classification
M	m3	Hotel Booking Demand	119390	32	https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand
M	m4	Medical Insurance Charges	2772	7	https://huggingface.co/datasets/rahulvyasm/medical_insurance_data
M	m5	Predict Students Dropout And Academic Success	4424	37	https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success
M	m6	Online Shoppers Purchasing Intention Dataset	12330	18	https://archive.ics.uci.edu/dataset/468/online+shoppers+purchasing+intention+dataset
M	m7	Stroke Prediction Dataset	5110	12	https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset
M	m8	Bank Marketing	45211	17	https://archive.ics.uci.edu/dataset/222/bank+marketing
M	m9	HR Analytics: Job Change of Data Scientists	19158	14	https://www.kaggle.com/datasets/arashnic/hr-analytics-job-change-of-data-scientists
M	m10	Mobile Price Classification	2000	21	https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification
M	m11	Health Insurance Cross Sell Prediction	381109	12	https://www.kaggle.com/datasets/anmolkumar/health-insurance-cross-sell-prediction
N	n1	Spambase	4601	58	https://www.openml.org/d/44
N	n2	Airfoil Self-Noise	1503	6	https://archive.ics.uci.edu/dataset/291/airfoil+self-noise
N	n3	Wine Quality	4898	12	https://archive.ics.uci.edu/dataset/186/wine+quality
N	n4	Communities And Crime	1994	128	https://archive.ics.uci.edu/dataset/183/communities+and+crime
N	n5	Superconductivity	21263	82	https://www.openml.org/d/44964
N	n6	BEED: Bangalore EEG Epilepsy Dataset	8000	17	https://archive.ics.uci.edu/dataset/1134/beed:+bangalore+eeg+epilepsy+dataset
N	n7	Anuran Calls Mfccs	7195	26	https://archive.ics.uci.edu/dataset/406/anuran+calls+mfccs
N	n8	Secom	1567	593	https://archive.ics.uci.edu/dataset/179/secom
N	n9	Pen Based Recognition Of Handwritten Digits	10992	17	https://archive.ics.uci.edu/dataset/81/pen-based+recognition+of+handwritten+digits
N	n10	Dry Bean Dataset	13611	17	https://archive.ics.uci.edu/dataset/602/dry+bean+dataset
N	n11	Magic Gamma Telescope	19019	11	https://archive.ics.uci.edu/dataset/159/magic+gamma+telescope
N	n12	Skin Segmentation	245057	4	https://archive.ics.uci.edu/dataset/229/skin+segmentation
N	n13	Combined Cycle Power Plant	9568	5	https://archive.ics.uci.edu/dataset/294/combined+cycle+power+plant
N	n14	First-Order Theorem Proving	2000	52	https://www.openml.org/d/44663
N	n15	COIL2000	9822	86	https://www.openml.org/d/298
N	n16	Credit Card Fraud Detection	284807	31	https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
N	n17	Statlog Shuttle	14500	10	https://archive.ics.uci.edu/dataset/148/statlog+shuttle
N	n18	APS Failure at Scania Trucks	76000	171	https://archive.ics.uci.edu/dataset/421/aps+failure+at+scania+trucks
N	n19	Fashion-MNIST	70000	785	https://www.openml.org/d/40996

B.4 Models Catalog

Model family	Model	Scope	Color
Transformer-based			
	REaLTabFormer (Solatorio & Dupriez, 2023)	Single + relational	#6F4C9B
Diffusion-based			
	TabDDPM (Kotelnikov et al., 2023)	Single table	#004488
	TabSyn (Zhang et al., 2024)	Single table	#4477AA
	ForestDiffusion (Jolicoeur-Martineau et al., 2024)	Single table	#66CCEE
	TabDiff (Shi et al., 2025)	Single table	#009988
Adversarial / energy-based			
	CTGAN (Xu et al., 2019)	Single table	#CC3311
	ARF (Watson et al., 2023)	Single table	#EE7733
	TabPFGen (Ma et al., 2023)	Single table	#DDAA33
VAE / latent / flow-based			
	TVAE (Xu et al., 2019)	Single table	#882255
	TabbyFlow (Guzmán-Cordero et al., 2025)	Single table	#CC6677
Probabilistic / synthetic database			
	BayesNet (DataSynthesizer) (Ping et al., 2017)	Single table	#666666

Table 7: Models currently instantiated in the benchmark roster, grouped by shared modeling backbone.

C Query Templates and Five Query Families

C.1 Query-Centric Fidelity Overview

TabQueryBench treats query-centric fidelity as a workload-facing measure: a synthetic table is useful for analytical workflows only when it preserves the answers to the types of queries users are likely to run. The benchmark therefore reports family-level signals instead of relying on a single aggregate resemblance score.

C.2 Workload Sources

The current appendix-facing template inventory surfaces 44 templates in total. Table 8 lists the public benchmark, repository, documentation, and paper sources that directly materialize into one or more released templates.

Direct source	Type	Templates
Benchmarks and benchmark specifications		
ClickBench (ClickHouse, 2022)	Benchmark	8
H2O db-benchmark (H2O.ai, 2020)	Benchmark	9
TPC-H / TPC-DS benchmark specs (TPC, 1999); (TPC, 2017)	Benchmark specs	1
TPC-DS benchmark family (TPC, 2017)	Benchmark spec	3
TPC-H qgen (TPC, 1999)	Benchmark	4
Repositories and benchmark-adjacent sources		
TPC-DS qualification repo (CWI, n.d.)	TPC-DS repository	2
TPC-DS Altinity repo (Altinity, n.d.)	TPC-DS repository	2
RTABench `order_events` slice (Timescale, 2025)	Benchmark + GitHub	2
Documentation sources		
BigQuery approximate aggregate docs (Google Cloud, n.d.)	Documentation	2
ClickHouse aggregate docs (ClickHouse, n.d.)	Documentation	1
Snowflake `PERCENTILE_CONT` docs (Snowflake, 2026)	Documentation	2
Snowflake `WIDTH_BUCKET` docs (Snowflake, 2026)	Documentation	1
Trino aggregate docs (Trino, 2026)	Documentation	3
Apache Druid SQL-function docs (Apache Druid, n.d.)	Documentation	1
Paper source		
Preserving Missing Data Distribution in Synthetic Data (Wang et al., 2023)	Paper	3

Table 8: Sources that directly materialize into one or more templates in the released library.

C.3 44-Template Taxonomy Catalog

The compact taxonomy summary in Appendix C.4 collapses related templates into family-level rows. The longtable below expands that same appendix-facing taxonomy back to the template level while keeping all 44 released templates in one consistent row format.

Table 9: **44-template taxonomy catalog.**

Template ID	No.	Canonical template	Family	Subfamily	Upstream source
`grp_cond_rate`	C-G1	Condition Rate by Group	Conditional	Global Structure	TPC-H, TPC-DS
`grp_ratio_2cond`	C-G2	Two-Condition Ratio by Group	Conditional	Global Structure	TPC-DS
`ds_within_grp_share`	C-G3	Item Share Within Each Group	Conditional	Global Structure	TPC-DS qualification
`win_part_avg`	C-G4	Partition Average by Group	Conditional	Global Structure	H2O db-benchmark
`2d_tgt_rate`	C-G5	2D Target-Rate Surface	Conditional	Global Structure	TPC-DS
`binned_num_grp_avg`	C-G6	Binned Group Average	Conditional	Global Structure	Snowflake WIDTH_BUCKET docs
`grp_disp_rank`	C-G7	Group Dispersion Rank	Conditional	Global Structure	Trino aggregate docs
`ds_base_gated_rank`	C-G8	Baseline-Gated Extreme Ranking	Conditional	Global Structure	TPC-DS Altinity
`flt_2d_grp_count`	C-L1	Filtered 2D Count Surface	Conditional	Local Slice	TPC-DS
`med_flt_num`	C-L2	Filtered Median Slice	Conditional	Local Slice	Snowflake PERCENTILE_CONT docs

Continued on next page

Table 9: **44-template taxonomy catalog (continued).**

Template ID	No.	Canonical template	Family	Subfamily	Upstream source
cond_grp_quants	C-L3	Filtered Group Quantiles	Conditional	Local Slice	ClickHouse aggregate docs
rta_time_bucket_cnt	C-L4	Filtered Time-Bucket Count	Conditional	Local Slice	RTABench order_events
rta_time_bucket_mavg	C-L5	Time-Bucket Moving Average by Group	Conditional	Local Slice	RTABench order_events
cb_grp_distinct_topk	S-1.1	Distinct-Coverage Ranking by Group	Subgroup	Structure	ClickBench
cb_flt_distinct_topk	S-1.2	Filtered Distinct-Coverage Ranking by Group	Subgroup	Structure	ClickBench
h2o_grp_sum	S-1.3	Total-Measure Ranking by Group	Subgroup	Structure	H2O db-benchmark
grp_avg_num	S-1.4	Mean-Measure Comparison by Group	Subgroup	Structure	H2O db-benchmark
guarded_grp_avg	S-1.5	Support-Gated Mean Comparison by Group	Subgroup	Structure	H2O db-benchmark
ds_topk_grp_sum	S-1.6	Filtered Total-Measure Ranking by Group	Subgroup	Structure	TPC-DS qualification
h2o_2d_grp_sum	S-1.7	Two-Dimensional Total-Measure Ranking	Subgroup	Structure	H2O db-benchmark
2d_grp_avg	S-1.8	Two-Dimensional Mean Comparison	Subgroup	Structure	H2O db-benchmark
cb_grp_summary_topk	S-1.9	Support-and-Mean Ranking by Group	Subgroup	Structure	ClickBench
tpch_2d_summary	S-1.10	Filtered Two-Dimensional Measure Summary	Subgroup	Structure	TPC-H qgen
h2o_2d_robust	S-1.11	Two-Dimensional Robust Measure Summary	Subgroup	Structure	H2O db-benchmark
tpch_max_agg_win	S-1.12	Top Aggregate Winner by Group	Subgroup	Structure	TPC-H qgen
wtd_topk_sum	S-1.13	Support-Gated Weighted Total Ranking by Group	Subgroup	Structure	BigQuery approx docs
cb_grp_count	S-2.1	Group Count Distribution	Subgroup	Size	ClickBench
cb_flt_grp_count	S-2.2	Filtered Group Count Ranking	Subgroup	Size	ClickBench
cb_2d_topk_count	S-2.3	Two-Dimensional Group Count Ranking	Subgroup	Size	ClickBench
quant_tail_slice	T-1.1	Top-Quantile Tail Members	Tail / Rarity	Tail Coverage	Snowflake PERCENTILE_CONT docs
global_z_outliers	T-1.2	Global Z-Score Outliers	Tail / Rarity	Tail Coverage	Trino aggregate docs
h2o_topn_in_grp	T-1.3	Within-Group Top-N Extremes	Tail / Rarity	Tail Coverage	H2O db-benchmark
ds_subgrp_base_outlier	T-1.4	Subgroup-Relative Outliers	Tail / Rarity	Tail Coverage	TPC-DS Altinity
tpch_rel_total_thr	T-2.1	Groups Above a Total-Share Threshold	Tail / Rarity	Tail Size	TPC-H qgen
tpch_thr_grp_rank	T-2.2	Above-Threshold Group Ranking	Tail / Rarity	Tail Size	TPC-H qgen
grp_pct_point	T-2.3	Group-wise Percentile Level	Tail / Rarity	Tail Size	BigQuery approx docs
thr_rarity_cdf	T-2.4	Threshold Exceedance Rarity	Tail / Rarity	Tail Size	Apache Druid SQL docs
miss_rate_marg	M-1.1	Marginal Missing Rate	Missingness	Marginal Missingness	Preserving Missing Data Distribution in Synthetic Data
miss_rate_disc_state	M-2.1	Missing Rate by Discrete State	Missingness	Broad Co-Missingness	Preserving Missing Data Distribution in Synthetic Data
miss_rate_cont_bucket	M-2.2	Missing Rate by Continuous Bucket	Missingness	Broad Co-Missingness	Preserving Missing Data Distribution in Synthetic Data
card_supp_rank_prof	K-1.1	Support Rank Profile	Cardinality / Range	Discrete	ClickBench
card_dist_share_prof	K-1.2	Distinct Share Profile	Cardinality / Range	Discrete	ClickBench
card_cont_range_env	K-1.3	Continuous Range Envelope Profile	Cardinality / Range	Continuous	Trino aggregate docs
card_hi_card_resp	K-2.1	High-Cardinality Response Stability	Cardinality / Range	Discrete	H2O db-benchmark

C.4 Five Query Families

C.4.1 Subgroup

Goal. Evaluate whether synthetic data preserves subgroup-level analytical behavior rather than only global marginals.

Template-side branches.

1. **Structure (13 templates).** Grouped summaries that keep the subgroup object fixed but vary the summary signal, including totals, means, distinct coverage, robust summaries, and winner-style aggregate views.
2. **Size (3 templates).** Absolute subgroup count or support views over one-dimensional and two-dimensional groupings.

Table 10 lists every Subgroup template and explains the analytical behavior it probes and the reason for its branch assignment.

Table 10: **Subgroup template inventory.** Each row states the analytical behavior tested by the template and why it belongs to the Structure or Size branch.

ID	Canonical template	What it probes	Why this branch
S-1.1	Distinct-Coverage Ranking by Group	Ranks groups by the number of distinct values or entities they cover, testing whether synthetic data preserves relative subgroup breadth.	*Structure*: the target is the ordering and profile of subgroup coverage, rather than the raw number of rows in each group.

Continued on next page

Table 10: **Subgroup template inventory (continued).**

ID	Canonical template	What it probes	Why this branch
S-1.2	Filtered Distinct-Coverage Ranking by Group	Repeats distinct-coverage ranking after applying a valid filter, testing whether subgroup breadth remains stable in a restricted analytical view.	*Structure*: the query compares group-level coverage behavior under a scoped view; the filter changes context but not the subgroup object.
S-1.3	Total-Measure Ranking by Group	Ranks groups by a total such as a sum, exposing whether the same subgroups dominate an aggregate measure.	*Structure*: it evaluates the relative aggregate pattern across groups, including which groups appear at the top of the ranking.
S-1.4	Mean-Measure Comparison by Group	Compares the average of a numeric measure across groups and tests whether group-level central tendency is preserved.	*Structure*: the score concerns the pattern of group means, not group population size alone.
S-1.5	Support-Gated Mean Comparison by Group	Compares group means while excluding groups with insufficient support, testing robust subgroup comparisons under a minimum-support rule.	*Structure*: the support gate controls which groups are eligible, while the measured object remains the cross-group mean pattern.
S-1.6	Filtered Total-Measure Ranking by Group	Ranks group-level totals inside a filtered view, testing whether aggregate dominance is preserved after restricting the rows considered.	*Structure*: it probes the ordering of subgroup aggregate behavior; the filter is a query scope rather than a separate conditional slice diagnostic.
S-1.7	Two-Dimensional Total-Measure Ranking	Ranks pairs of groups by a total measure, testing whether joint subgroup structure survives beyond one-dimensional grouping.	*Structure*: the central object is the two-dimensional aggregate surface and its ranking pattern.
S-1.8	Two-Dimensional Mean Comparison	Compares mean measures across combinations of two grouping attributes, testing preservation of joint subgroup-level averages.	*Structure*: it evaluates the arrangement of group-pair means, rather than the marginal support count of the pairs.
S-1.9	Support-and-Mean Ranking by Group	Ranks groups using a summary that combines subgroup support and a numeric measure, testing whether a multi-signal group ordering is kept.	*Structure*: the output is a group-level ranking driven by the summary profile, not a standalone count distribution.
S-1.10	Filtered Two-Dimensional Measure Summary	Summarizes a two-dimensional aggregate surface after a filter, testing whether joint subgroup relationships remain usable in a scoped query.	*Structure*: it preserves the shape of a filtered group-pair summary; the primary object is still the aggregate structure.
S-1.11	Two-Dimensional Robust Measure Summary	Compares robust summaries across group pairs to reduce sensitivity to extreme rows while testing joint subgroup behavior.	*Structure*: robustness changes the summary operator, while the evaluated object remains the two-dimensional subgroup pattern.
S-1.12	Top Aggregate Winner by Group	Identifies the group that wins under an aggregate comparison, testing whether the identity of the leading subgroup is preserved.	*Structure*: the query probes a winner identity and ordering relationship rather than the amount of mass in each group.
S-1.13	Support-Gated Weighted Total Ranking by Group	Ranks groups by a weighted total after enforcing a support condition, testing whether weighted subgroup dominance is preserved reliably.	*Structure*: the target is the weighted ranking among eligible groups; support affects validity but is not the outcome being measured.
S-2.1	Group Count Distribution	Measures the number of rows assigned to each group and the resulting distribution of subgroup sizes.	*Size*: the query directly targets how much mass each subgroup contains, without asking which numeric summary it produces.
S-2.2	Filtered Group Count Ranking	Ranks groups by row count inside a filtered view, testing whether relative subgroup sizes remain stable after restricting the data.	*Size*: the measured quantity is subgroup population, while the filter only defines the rows included in the count.
S-2.3	Two-Dimensional Group Count Ranking	Ranks group pairs by their row counts, testing whether joint support mass is preserved across a two-dimensional partition.	*Size*: the query focuses on the amount of support in each group pair, rather than on a value-based aggregate over those rows.

C.4.2 CONDITIONAL

Goal. Evaluate whether conditional behavior is preserved both at full grouped scope and after narrowing attention to local slices.

Template-side branches.

1. **Global structure (8 templates).** Grouped conditional summaries such as condition rates, grouped ratios, grouped averages, grouped dispersion, and other full-scope conditional views.
2. **Local slices (5 templates).** The same grouped conditional scaffold after adding a predicate-defined slice or local support gate.

Table 11 lists every Conditional template and explains the analytical behavior it probes and the reason for its branch assignment.

Table 11: **Conditional template inventory.** Each row states the analytical behavior tested by the template and why it belongs to the Global Structure or Local Slice branch.

ID	Canonical template	What it probes	Why this branch
C-G1	Condition Rate by Group	Measures how often a target condition holds within each group, testing whether conditional incidence varies across groups in the same way.	*Global Structure*: it evaluates a full-table grouped conditional relationship without a predicate-defined local slice.
C-G2	Two-Condition Ratio by Group	Compares the rates of two conditions within each group, testing whether relative conditional behavior is preserved across groups.	*Global Structure*: the query describes a grouped ratio over the full available scope rather than a filtered subpopulation.
C-G3	Item Share Within Each Group	Measures the share contributed by an item or category inside each group, testing within-group composition.	*Global Structure*: the outcome is a group-level composition pattern over the full table.
C-G4	Partition Average by Group	Computes a partition-aware average for each group, testing whether grouped central tendency remains aligned with the partition structure.	*Global Structure*: it probes a grouped conditional summary without introducing an explicit local filter.
C-G5	2D Target-Rate Surface	Builds a target-rate surface over pairs of grouping attributes, testing whether conditional behavior is preserved jointly across two dimensions.	*Global Structure*: the target is a full-scope two-dimensional conditional surface, not a local slice.
C-G6	Binned Group Average	Bins a numeric attribute and compares group averages across the bins, testing conditional trends over a discretized numeric axis.	*Global Structure*: binning organizes the full-scope conditional relationship rather than restricting rows to one selected slice.
C-G7	Group Dispersion Rank	Ranks groups by a dispersion statistic, testing whether conditional variability and group ordering are preserved.	*Global Structure*: it evaluates a grouped variability pattern at global scope.
C-G8	Baseline-Gated Extreme Ranking	Ranks groups by an extreme conditional value after applying a baseline eligibility rule, testing structured extremes across groups.	*Global Structure*: the gate controls valid groups, while the measured object remains the global grouped conditional ranking.
C-L1	Filtered 2D Count Surface	Counts group pairs inside a predicate-defined slice, testing whether joint conditional support survives local filtering.	*Local Slice*: the defining operation is a `WHERE`-based restriction before evaluating the two-dimensional grouped result.
C-L2	Filtered Median Slice	Computes a median-style numeric summary after restricting rows to a local subpopulation.	*Local Slice*: the query explicitly measures behavior inside a predicate-defined slice rather than across the full table.
C-L3	Filtered Group Quantiles	Measures grouped quantiles within a filtered view, testing local distributional structure beyond averages or counts.	*Local Slice*: the quantile summary is evaluated only after the local predicate has isolated a subpopulation.
C-L4	Filtered Time-Bucket Count	Counts rows in time buckets after filtering, testing whether local temporal support and grouped activity remain usable.	*Local Slice*: both the temporal aggregation and its answer are conditioned on a predicate-defined local view.
C-L5	Time-Bucket Moving Average by Group	Computes a moving average over time buckets for each group, testing whether temporal trends and group trajectories are preserved.	*Local Slice*: it belongs to the filtered temporal workload branch and evaluates a scoped group-by-time pattern.

For the paired local-vs-global diagnostic in Figure 6, we group filtered conditional queries by their SQL template rather than by dataset or model. Each local query first uses a `WHERE` predicate to isolate a subpopulation, then evaluates an aggregation inside that local slice. Its global counterpart is constructed by deleting only that `WHERE` predicate while keeping the remaining `SELECT`, `GROUP BY`, and `ORDER BY` structure unchanged. The five local-slice template parts are:

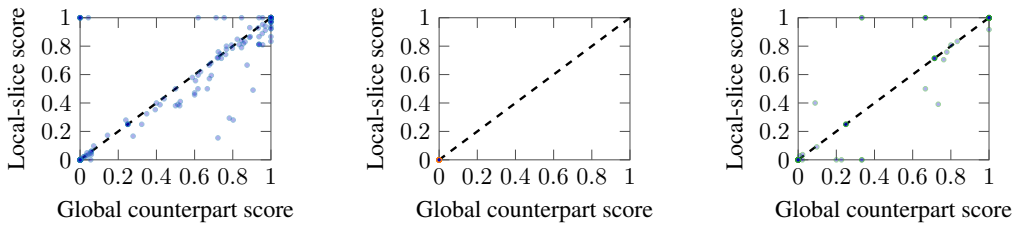
- **Part ①: Filtered two-dimensional group count** (228 paired query instances). Counts the joint support of two grouped columns inside a predicate-defined local slice.
- **Part ②: Filtered median slice** (29 paired query instances). Computes a median-style numeric summary after filtering to a local subpopulation.
- **Part ③: Filtered group quantile** (97 paired query instances). Measures whether grouped quantile structure is preserved inside a filtered conditional view.
- **Part ④: Filtered time-bucket count** (52 paired query instances). Evaluates time-bucketed counts under a local filter, matching dashboard-style filtered temporal summaries.
- **Part ⑤: Time-bucket moving average by group** (11 paired query instances). Computes a moving average over time-bucketed subgroup counts inside a predicate-defined local slice.

C.4.3 TAIL / RARITY

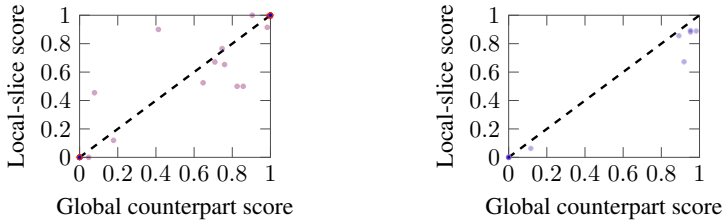
Goal. Evaluate whether synthetic data preserves low-support and rare-region behavior instead of washing rare cases into the distributional head.

Template-side branches.

1. **Tail Coverage (4 templates).** Queries that identify rare members, outliers, or extreme cases and test whether the same tail-support region remains recoverable after synthesis.



(a) Part ①: 2D count (b) Part ②: median slice (c) Part ③: group quantile



(d) Part ④: time-bucket count (e) Part ⑤: moving average

Figure 11: **Local-slice conditional query parts.** Each panel uses the same paired construction as Figure 6: the x-axis scores the query after deleting the local `WHERE` predicate, and the y-axis scores the original local-slice query. The diagonal marks equal fidelity between the global counterpart and the filtered local slice.

2. **Tail Size (4 templates).** Queries that measure how much mass, level, or threshold exceedance is assigned to a rare region and test whether synthetic data preserves the scale of that tail event.

Table 12 lists every Tail / Rarity template and explains whether it probes rare-region identity or the mass assigned to the rare region.

Table 12: **Tail / Rarity template inventory.** Each row states the rare-region behavior tested by the template and why it belongs to the Tail Coverage or Tail Size branch.

ID	Canonical template	What it probes	Why this branch
`T-1.1`	Top-Quantile Tail Members	Identifies members in an extreme quantile region and tests whether the same rare values remain recoverable after synthesis.	*Tail Coverage*: the primary object is the identity of the rare members or support region, not its aggregate mass.
`T-1.2`	Global Z-Score Outliers	Finds observations with unusually large standardized deviations from the global center, testing preservation of globally rare extremes.	*Tail Coverage*: the query asks whether the rare outlier support is still present and identifiable.
`T-1.3`	Within-Group Top-N Extremes	Finds the top extreme observations separately within each group, testing subgroup-relative rare support.	*Tail Coverage*: it evaluates which rare members occur in each group, rather than only how much total mass is extreme.
`T-1.4`	Subgroup-Relative Outliers	Detects observations that are extreme relative to their subgroup, testing whether conditional rare cases survive synthesis.	*Tail Coverage*: the target is the identity of subgroup-relative outliers and their recoverability.
`T-2.1`	Groups Above a Total-Share Threshold	Counts or identifies groups whose total exceeds a specified share of the overall total, testing the amount of mass in a rare high-contribution region.	*Tail Size*: the query emphasizes how much aggregate mass crosses the threshold rather than the exact identity of individual outliers.
`T-2.2`	Above-Threshold Group Ranking	Ranks groups whose aggregate values exceed a threshold, testing the scale and ordering of threshold-defined tail responses.	*Tail Size*: the main object is the magnitude of threshold-exceeding group behavior and its concentration.
`T-2.3`	Group-wise Percentile Level	Reports percentile levels within groups, testing whether the scale of upper or lower tail responses is preserved.	*Tail Size*: percentile values characterize the level or mass of a tail event rather than enumerating rare identities.
`T-2.4`	Threshold Exceedance Rarity	Measures how rarely a value exceeds a specified threshold, testing the amount of probability mass assigned to an extreme event.	*Tail Size*: it directly measures rare-event frequency or mass relative to the threshold.

Threshold-stress diagnostic. This appendix also gives the detailed definition and measurement protocol for the tail and rarity diagnostic used in the main-text threshold-stress analysis. Unless otherwise stated, the paper uses $\tau = 3\%$ as the default tail threshold; the threshold-stress figures additionally sweep $\tau \in \{10\%, 8\%, 6\%, 4\%, 3\%, 2\%, 1\%, 0.5\%, 0.1\%\}$. The diagnostic is complementary to the grounded Tail Coverage and Tail Size SQL templates described in the template taxonomy.

Column-level tail region. Let R denote the real training split, let S denote the synthetic table, and let τ be a tail-mass threshold. The real tail region $T_R^{(\tau)}$ is defined separately for each column. For a continuous column, the empirical CDF of R determines the lower and/or upper quantile interval(s) whose total probability mass is τ. For a categorical column, categories are ordered by increasing frequency in R, and the least frequent categories are accumulated until their cumulative frequency reaches τ. Thus, the categorical tail is a set of rare states, whereas the continuous tail is a rare interval or union of intervals.

Tail size. Tail size measures whether the synthetic table places the appropriate amount of mass in the real-defined tail. We first fix $T_R^{(\tau)}$ using the real data, then count the real and synthetic mass assigned to this same region. Let

$$m_R^{(\tau)} = \frac{\#\{x \in R : x \in T_R^{(\tau)}\}}{|R|}, \qquad m_S^{(\tau)} = \frac{\#\{x \in S : x \in T_R^{(\tau)}\}}{|S|}.$$

For categorical columns, the implementation uses the bounded relative-error score

$$\mathrm{Size}^{(\tau)} = \mathrm{clip}\left(1 - \frac{|m_S^{(\tau)} - m_R^{(\tau)}|}{m_R^{(\tau)}}, 0, 1\right),$$

when $m_R^{(\tau)} > 0$; when $m_R^{(\tau)} = 0$, the score is 1 if $m_S^{(\tau)} = 0$ and 0 otherwise. For continuous columns, the real lower and upper tail cutoffs define two target tail masses. If $m_{S,-}^{(\tau)}$ and $m_{S,+}^{(\tau)}$ are the synthetic masses below and above those real cutoffs, respectively, the implementation uses

$$\mathrm{Size}^{(\tau)} = \frac{1}{2}\left[\min\left(\frac{m_{S,-}^{(\tau)}}{\tau}, 1\right) + \min\left(\frac{m_{S,+}^{(\tau)}}{\tau}, 1\right)\right].$$

Thus the reported tail-size score is explicitly bounded in $[0, 1]$. It measures preservation of tail mass and does not require the synthetic table to identify exactly the same rare states or interval endpoints.

Tail coverage. Tail coverage measures whether the synthetic table recovers the identity of the rare region. We independently construct the corresponding synthetic tail $T_S^{(\tau)}$ using the same threshold rule and compute its overlap with the real tail:

$$\mathrm{Coverage}^{(\tau)} = \frac{\mu\left(T_R^{(\tau)} \cap T_S^{(\tau)}\right)}{\mu\left(T_R^{(\tau)}\right)}.$$

For categorical columns, μ is set cardinality, so coverage is the fraction of real tail categories recovered by the synthetic tail. For continuous columns, μ is interval length (or the corresponding one-dimensional measure), so coverage is the fraction of the real tail range covered by the synthetic tail range.

Tail overall and threshold stress. The column-level tail overall score is the equally weighted combination

$$\mathrm{TailOverall}^{(\tau)} = \frac{1}{2}\left(\mathrm{Size}^{(\tau)} + \mathrm{Coverage}^{(\tau)}\right).$$

Because both component scores are bounded in $[0, 1]$, the resulting TailOverall score is also bounded in $[0, 1]$. The threshold-stress analysis evaluates these quantities over $\tau \in \{10\%, 8\%, 6\%, 4\%, 3\%, 2\%, 1\%, 0.5\%, 0.1\%\}$. The resulting curves report absolute mean tail overall scores at each threshold, not scores normalized to the 10% baseline. Tightening τ therefore tests whether a model continues to preserve increasingly rare support, while the separate size and coverage curves distinguish loss of tail mass from loss of rare-region identity.

C.4.4 MISSINGNESS

Goal. Treat native missingness as an analyzable signal rather than as noise to be removed before fidelity assessment.

Template-side branches.

1. **Marginal Missingness (1 template).** Leaves the missing-rate signal unconditional and measures only the overall missing-rate profile.
2. **Broad Co-Missingness (2 templates).** Reuses the same missingness target but conditions it on either a discrete state or a continuous bucket, producing broad state-conditioned missingness profiles.

Table 13 lists every Missingness template and explains whether it measures marginal missingness or state-conditioned co-missingness.

Table 13: **Missingness template inventory.** Each row states the missingness behavior tested by the template and why it belongs to the Marginal Missingness or Broad Co-Missingness branch.

ID	Canonical template	What it probes	Why this branch
`M-1.1`	Marginal Missing Rate	Measures the overall fraction of missing values in a target column, testing whether the global missingness level is preserved.	*Marginal Missingness*: it has no conditioning attribute and therefore probes only the unconditional missing-rate profile.
`M-2.1`	Missing Rate by Discrete State	Measures the missing rate of a target column separately for each discrete state, testing structured co-missingness across categories.	*Broad Co-Missingness*: missingness is conditioned on a categorical state, exposing dependence between the target and another attribute.
`M-2.2`	Missing Rate by Continuous Bucket	Measures the missing rate of a target column across buckets of a continuous conditioning attribute.	*Broad Co-Missingness*: bucketing creates a structured conditional profile without requiring exact continuous-value matching.

C.4.5 CARDINALITY / RANGE

Goal. Evaluate whether observed support richness and numerical coverage envelopes remain analytically usable after synthesis.

Template-side branches.

1. **Discrete (3 templates).** Support/rank/concentration views over observed discrete values, including support ranking, support-share profiling, and high-cardinality response summaries.
2. **Continuous (1 template).** Min/max and range-envelope views over numerical coverage.

Table 14 lists every Cardinality / Range template and explains whether it probes discrete support richness or continuous numerical coverage.

Table 14: **Cardinality / Range template inventory.** Each row states the support or range behavior tested by the template and why it belongs to the Discrete or Continuous branch.

ID	Canonical template	What it probes	Why this branch
`K-1.1`	Support Rank Profile	Ranks observed discrete values or categories by support, testing whether the ordering of common and rare states is retained.	*Discrete*: the target is the identity and rank of discrete support values, not a continuous interval.
`K-1.2`	Distinct Share Profile	Measures how support mass is distributed across distinct values, testing concentration and share structure within a discrete domain.	*Discrete*: it describes shares over enumerated categories or values and their support concentration.
`K-1.3`	Continuous Range Envelope Profile	Measures the lower and upper numerical envelope of a continuous column, testing whether its usable range remains covered.	*Continuous*: the analytical object is an interval or numerical range rather than a finite set of category identities.
`K-2.1`	High-Cardinality Response Stability	Tests whether responses remain stable when the query operates over a large discrete support with many distinct values.	*Discrete*: the challenge comes from preserving a rich finite support and its response behavior, even when the cardinality is high.

D Query Grounding

D.1 Pipeline Overview

The query-generation pipeline has two stages. It first extracts reusable analytical templates from public SQL workloads and assigns them to five query families—subgroup, conditional, tail/rarity, missingness, and cardinality/range. It then grounds applicable templates to each dataset schema, realizes executable SQL under fixed constraints, and records the resulting semantic answer contract. The source inventory and taxonomy are detailed in Appendix C.2 and Appendix C.3; the corresponding answer-scoring rules are given in Appendix E.

D.2 Stage 1: Template Construction

Source acquisition and provenance. We construct the query-template library from public analytical workloads, including TPC-H, TPC-DS, ClickBench, H2O db-benchmark, and RTABench. Each source query is retained with its workload identifier, source-query identifier, and bibliographic provenance. The purpose of this step is to preserve reusable analytical intent rather than to reproduce the complete execution protocol of any individual database benchmark.

Canonical template construction. We abstract each source query into a single-table SQL skeleton whose dataset-dependent elements are represented by typed roles and bounded parameters. Typical roles include grouping columns, measure columns, predicate columns, condition columns, entity columns, and time columns. The skeleton preserves the analytical structure, while column names, observed values, thresholds, and other dataset-specific choices remain unbound until the grounding stage.

Template records and family assignment. Each template record stores a stable identifier, analytical intent, SQL skeleton, required and optional roles, binding constraints, query-family assignment, source provenance, portability metadata, and release status. Templates are assigned to the five query families described in Appendix C.4; the family label describes the primary analytical object of the template and does not change its source provenance. The concrete source-to-template mappings are listed in Appendix C.2.

Screening and materialization policy. The construction process screens templates for duplicate identifiers, missing provenance, invalid family mappings, unsupported SQL features, and violations of the single-table benchmark contract. Templates that satisfy the portability and execution requirements enter the core library. Other well-specified templates may be retained as extensions or prior-only material, but are not silently included in the released workload.

Versioned construction and auditability. Template construction is deterministic and versioned. Each library build records the hashes of its input catalogs and source-query bank, materialized template counts, skipped mappings, duplicate identifiers, and workload coverage. This makes every released grounded workload traceable to a specific source library and provides an audit trail for later library updates.

D.3 Stage 2: Dataset Grounding and SQL Realization

Dataset profiling. For each input dataset, we first construct a typed profile containing the schema, row count, column types, cardinalities, missingness indicators, observed categorical values, and candidate numeric thresholds. This profile defines the admissible search space for grounding and prevents the realization step from inventing schema elements or predicate values.

Applicability screening and binding. For every ready template, the grounding procedure checks whether all required roles have compatible candidates and whether structural constraints, such as distinct grouping columns or ordered temporal roles, can be satisfied. Templates with unavailable roles, incompatible types, or unrealizable predicates are skipped with an explicit reason. For applicable templates, the agent selects dataset columns and the configured binding procedure selects values from the admissible candidate set; the template skeleton and its required constraints remain fixed.

Deterministic validation and realization. Bindings are validated against the template contract before SQL generation. The validator checks required and unknown placeholders, candidate-column membership, allowed operators and values, observed categorical values, bounded numeric parameters,

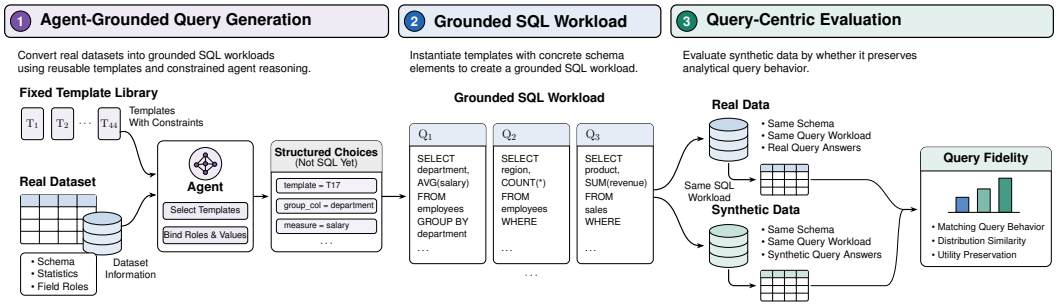


Figure 12: Agent query grounding pipeline.

distinct-role constraints, and temporal ordering. A deterministic renderer then instantiates the SQL skeleton with the validated bindings. Thus, dataset-specific choices are separated from the reusable analytical structure.

Execution filtering and retry policy. Each realized query is executed in an isolated SQLite environment under a fixed timeout. We retain only read-only, executable queries that return at least one row and are not entirely NULL. Invalid bindings, invalid SQL, timeouts, execution failures, empty results, and NULL-only results are rejected and may be retried under the configured grounding budget. Accepted queries are kept in deterministic template order.

Released artifacts and audit records. The generated bundle contains the accepted SQL, a machine-readable query registry, a manifest, and the SQLite materialization used for execution checks. Each accepted query records its template identifier, source provenance, template hash, concrete role bindings, semantic result contract, output columns, and preview rows. Skipped templates retain explicit failure reasons, while the manifest records input and template-library hashes, runtime metadata, timings, and usage counters. These records allow a released workload to be audited and regenerated without exposing raw model responses or server-local paths.

Algorithm 1 Template Grounding and Constrained SQL Realization

Require: dataset schema and profile; template pool with `can_vary` and `must_fix`; retry budget B
Ensure: accepted grounded problems and accepted SQL realizations

```
for all templates that are bindable on the dataset do
    construct admissible dataset-specific bindings
    for all admissible bindings do
        for b = 1 to B do
            realize a grounded problem and its main/check SQL pair
            validate type compatibility, must-fix preservation, and executability
            if both SQL queries pass validation then
                accept the grounded problem and stop retrying
                break
            end if
        end for
    end for
end for
```

Algorithm 1 summarizes the Stage 2 grounding and constrained SQL realization loop used after a template has been selected for a dataset.

D.4 Agent API Usage by Dataset

This appendix records dataset-level API usage.

This table group reports the recorded token usage of the GLM V11 workload builders in the repository release manifests and per-dataset grounding inventories. Costs are computed per million tokens using the provider rates shown below: GLM-5.3-Flash uses $0.15, $0.03, and $0.50; and GLM-5.3 uses $1.40, $0.26, and $4.40, respectively. These are standard API rates, not the separate Z.AI Coding Plan quota accounting. The repository contains three released runs for each GLM model; the GPT-5.4 grounding usage is reported in the table immediately above.[2]

Table 15: **Total recorded V11 grounding usage and estimated API cost.** Each row is one released run; repeated suffixes denote independent reruns. Dataset-level training rows and column counts are reported in the three tables below.

[2]Pricing page checked on 2026-09-22: `https://docs.z.ai/guides/overview/pricing`.

Run	Model	Queries	Input tok.	Cached in.	Output tok.	Total tok.	Cost (USD)
v11.2.1-2_glm-5.3-flash	GLM-5.3 Flash	2,729	25,976,126	2,331,072	672,327	26,648,453	3.95
v11.2.1-3_glm-5.3-flash	GLM-5.3 Flash	2,734	25,972,319	3,159,360	695,009	26,667,328	3.86
v11.2.1_glm-5.3-flash	GLM-5.3 Flash	2,734	25,986,950	581,696	696,387	26,683,337	4.18
v11.2.2-2_glm-5.3	GLM-5.3	2,761	25,957,914	1,272,256	642,610	26,600,524	37.72
v11.2.2-3_glm-5.3	GLM-5.3	2,760	25,947,520	3,328,704	700,987	26,648,507	35.62
v11.2.2_glm-5.3	GLM-5.3	2,763	25,917,017	697,088	628,253	26,545,270	38.25
All recorded GLM runs		16,481	155,757,846	11,370,176	4,035,573	159,793,419	123.58

Table 16: **Dataset-level recorded usage for GLM-5.3 Flash.**

Dataset	Train rows	Cols	Queries	Input tok.	Cached in.	Output tok.	Total tok.	Cost (USD)
c1	35,855	7	78	196,279	41,344	35,143	231,422	0.04
c2	1,382	7	74	111,577	14,272	28,515	140,092	0.03
c3	2,551	3	53	99,837	14,656	48,144	147,981	0.04
c4	2,557	37	102	350,215	71,616	28,398	378,613	0.06
c5	6,732	23	188	357,025	27,392	30,634	387,659	0.07
c6	7,636	8	115	200,077	33,664	42,892	242,969	0.05
c7	10,368	9	84	128,403	9,280	27,709	156,112	0.03
c8	8,844	31	103	446,067	40,704	37,367	483,434	0.08
c9	26,215	10	158	294,305	41,280	45,961	340,266	0.06
c10	820,008	11	102	279,888	12,928	48,916	328,804	0.06
c11	54,045	43	107	533,205	10,304	32,496	565,701	0.09
c12	2,623	1,559	134	16,275,641	1,811,264	54,088	16,329,729	2.25
c13	1,966,628	69	118	1,385,199	194,304	44,800	1,429,999	0.21
c14	240,000	25	144	527,476	58,816	47,094	574,570	0.10
c15	480,000	25	253	534,431	59,136	42,172	576,603	0.09
c16	5,516	13	255	290,332	34,752	43,887	334,219	0.06
c17	7,045	12	231	270,733	6,912	37,710	308,443	0.06
c18	103,976	14	251	379,469	6,848	36,275	415,744	0.07
c19	32,759	16	217	651,774	35,904	42,952	694,726	0.11
m1	1,200	30	171	607,711	38,400	41,891	649,602	0.11
m2	46,873	44	152	747,716	116,544	47,448	795,164	0.12
m3	95,512	32	242	680,027	59,968	48,583	728,610	0.12
m4	2,217	7	107	173,539	6,976	34,245	207,784	0.04
m5	3,539	37	179	763,702	47,424	41,089	804,791	0.13
m6	9,864	18	168	392,160	75,584	39,145	431,305	0.07
m7	4,088	12	227	245,685	18,368	36,079	281,764	0.05
m8	36,168	17	249	327,447	38,336	39,174	366,621	0.06
m9	15,326	14	245	318,108	45,376	40,045	358,153	0.06
m10	1,600	21	164	443,730	6,656	49,438	493,168	0.09
m11	304,887	12	160	279,180	31,040	34,648	313,828	0.06
n1	3,680	58	171	1,155,956	68,672	53,153	1,209,109	0.19
n2	1,202	6	111	184,436	21,824	32,630	217,066	0.04
n3	3,918	12	173	329,433	15,360	36,948	366,381	0.07
n4	1,595	128	283	3,109,939	328,640	41,941	3,151,880	0.45
n5	17,010	82	177	2,415,926	83,584	42,230	2,458,156	0.37
n6	6,400	17	176	429,287	61,184	41,786	471,073	0.08
n7	5,756	26	180	839,130	29,248	44,448	883,578	0.14
n8	1,253	593	284	13,525,624	384,320	72,680	13,598,304	2.02
n9	8,794	17	176	428,065	39,360	43,966	472,031	0.08
n10	10,888	17	180	563,229	7,232	42,177	605,406	0.10
n11	15,215	11	168	325,431	26,176	44,603	370,034	0.07
n12	196,045	4	95	142,027	6,976	37,189	179,216	0.04
n13	7,654	5	111	174,840	14,720	39,776	214,616	0.04
n14	1,600	52	175	1,326,726	89,728	50,912	1,377,638	0.21
n15	7,857	86	115	1,805,712	153,984	39,153	1,844,865	0.27
n16	227,845	31	173	971,557	7,360	47,047	1,018,604	0.17
n17	11,600	10	155	267,126	19,520	46,601	313,727	0.06
n18	60,800	171	287	4,189,781	238,528	50,006	4,239,787	0.62
n19	56,000	785	176	17,460,232	1,465,664	49,539	17,509,771	2.47

Table 17: **Dataset-level recorded usage for GLM-5.3.**

Dataset	Train rows	Cols	Queries	Input tok.	Cached in.	Output tok.	Total tok.	Cost (USD)
c1	35,855	7	78	196,283	52,544	46,535	242,818	0.42
c2	1,382	7	76	118,037	25,536	29,787	147,824	0.27
c3	2,551	3	54	102,366	12,544	48,260	150,626	0.34

Continued on next page.

Table 17 continued.

Dataset	Train rows	Cols	Queries	Input tok.	Cached in.	Output tok.	Total tok.	Cost (USD)
c4	2,557	37	102	350,190	46,336	24,222	374,412	0.54
c5	6,732	23	186	335,115	93,952	21,420	356,535	0.46
c6	7,636	8	120	205,088	36,288	39,724	244,812	0.42
c7	10,368	9	88	144,437	25,344	30,933	175,370	0.31
c8	8,844	31	104	446,636	51,648	32,411	479,047	0.71
c9	26,215	10	157	286,835	49,024	33,692	320,527	0.49
c10	820,008	11	102	279,286	48,384	33,323	312,609	0.48
c11	54,045	43	106	518,286	101,760	28,802	547,088	0.74
c12	2,623	1,559	134	16,227,063	1,390,528	48,828	16,275,891	21.35
c13	1,966,628	69	120	1,386,089	179,776	36,799	1,422,888	1.90
c14	240,000	25	146	527,723	20,928	36,429	564,152	0.88
c15	480,000	25	255	533,377	34,560	38,334	571,711	0.88
c16	5,516	13	255	289,615	58,304	27,786	317,401	0.46
c17	7,045	12	231	270,920	19,008	42,857	313,777	0.55
c18	103,976	14	252	379,466	65,152	49,557	429,023	0.68
c19	32,759	16	219	651,207	18,944	39,613	690,820	1.06
m1	1,200	30	171	607,398	66,496	57,781	665,179	1.03
m2	46,873	44	155	746,126	77,312	46,164	792,290	1.16
m3	95,512	32	249	679,295	89,920	36,594	715,889	1.01
m4	2,217	7	108	177,667	32,896	36,114	213,781	0.37
m5	3,539	37	180	763,990	59,648	32,997	796,987	1.15
m6	9,864	18	168	392,311	47,552	41,299	433,610	0.68
m7	4,088	12	233	244,448	29,056	49,876	294,324	0.53
m8	36,168	17	249	326,717	41,792	39,118	365,835	0.58
m9	15,326	14	248	316,952	41,088	43,408	360,360	0.59
m10	1,600	21	168	454,575	42,048	35,289	489,864	0.74
m11	304,887	12	162	279,459	7,488	37,016	316,475	0.55
n1	3,680	58	179	1,155,936	69,056	38,547	1,194,483	1.71
n2	1,202	6	114	193,787	11,968	44,510	238,297	0.45
n3	3,918	12	176	329,330	17,728	41,813	371,143	0.62
n4	1,595	128	287	3,101,626	188,224	39,977	3,141,603	4.30
n5	17,010	82	177	2,417,208	215,680	38,353	2,455,561	3.31
n6	6,400	17	179	428,796	50,432	39,779	468,575	0.72
n7	5,756	26	180	838,625	84,736	37,900	876,525	1.24
n8	1,253	593	286	13,489,251	396,608	62,033	13,551,284	18.71
n9	8,794	17	180	427,235	28,992	40,828	468,063	0.74
n10	10,888	17	178	563,041	46,784	47,332	610,373	0.94
n11	15,215	11	168	325,158	18,240	47,565	372,723	0.64
n12	196,045	4	94	141,539	23,616	42,079	183,618	0.36
n13	7,654	5	111	174,613	25,664	43,229	217,842	0.41
n14	1,600	52	179	1,325,251	125,568	52,230	1,377,481	1.94
n15	7,857	86	118	1,809,425	165,376	44,168	1,853,593	2.54
n16	227,845	31	178	971,465	70,528	44,433	1,015,898	1.48
n17	11,600	10	156	265,683	25,280	35,926	301,609	0.50
n18	60,800	171	288	4,180,327	361,856	39,773	4,220,100	5.61
n19	56,000	785	180	17,447,198	505,856	46,407	17,493,605	24.05

E QUERY SCORING STANDARD

This appendix documents the scoring protocol used by TABQUERYBENCH to compare query answers on real and synthetic tables. The protocol is query-centric: each valid query answer is routed to one semantic answer type, scored by one primary metric, and represented as a similarity in $[0, 1]$, where larger is better. Auxiliary diagnostics, such as key overlap or raw distance values, are reported for interpretation but are not blended into the primary query score.

E.1 REPORTING AND APPLICABILITY RULES

Scorable query objects. The scoring protocol applies to semantic answers rather than raw table rows: scalar aggregates, grouped aggregates, grouped counts, bounded rates or shares, and ranked selections whose identity columns are visible in the output. It does not score `SELECT *`, raw column projections without aggregation, metadata queries, write statements, multi-table queries, or non-deterministic queries. Top-k queries are scorable only when the ranked identity and ordering are deterministic.

Single-primary rule. Each valid query has exactly one primary semantic scorer. The query-level score is the similarity produced by that scorer:

$$S_q = S_{\text{semantic}}(q) \in [0, 1].$$

The benchmark does not average several semantic metrics inside a single query. The scorer is selected from template metadata and the semantic answer type, not from the observed numeric range of the answer. For example, a numeric aggregate that happens to lie in $[0, 1]$ is not treated as a rate unless the query semantics declare it to be a rate, share, probability, or proportion.

Validity and failures. A query is invalid, rather than assigned a semantic zero, when execution fails, columns are missing or unexpected, values have incompatible types, a value is non-finite, a count is negative or null, a rate falls outside its declared scale, ranking identities are duplicated, or no scorer can be resolved. Invalid queries are excluded from means but reported through valid-query counts and invalid-reason counts, separating failures from valid but poor semantic agreement.

Key alignment. For keyed answers, scoring uses the union of real and synthetic keys, $K = K_R \cup K_S$, not only their intersection. A key present on both sides receives the normal per-key similarity; a key present on only one side receives similarity 0 for numeric and rate answers, or probability mass 0 on the missing side for count distributions. Duplicate keys are summed only for count/support distributions; duplicate numeric, rate, or ranking keys are invalid unless the query contract gives an explicit resolution rule.

Nulls and non-finite values. For numeric and rate-valued cells, SQL `NULL` is treated as a semantic missing value: `NULL` versus `NULL` scores 1, while `NULL` versus a finite value scores 0. NaN and $\pm\infty$ are invalid inputs, not missing values. Count outputs must be finite, numeric, and non-negative.

E.2 QUERY-CENTRIC SEMANTIC SCORES

Numeric magnitude. Scalar numeric aggregates and keyed numeric aggregates are scored by normalized sMAPE similarity, following the established sMAPE literature (Makridakis, 1993; Chen et al., 2017). For finite real values r and s,

$$S_{\text{num}}(r, s) = 1 - \frac{|r - s|}{|r| + |s|}.$$

If both values are zero, the similarity is 1; if exactly one value is zero, the similarity is 0. Signed values are allowed. For keyed answers, the final score is the macro-average of per-key similarities over K.

Count and support distributions. Grouped counts and support profiles are first normalized into distributions over the union key space:

$$P_R(k) = \frac{c_R(k)}{\sum_j c_R(j)}, \qquad P_S(k) = \frac{c_S(k)}{\sum_j c_S(j)}.$$

The primary score is the complement of total variation distance, a standard probability metric (Gibbs & Su, 2002),

$$S_{\text{count}} = 1 - \frac{1}{2} \sum_{k \in K} |P_R(k) - P_S(k)|.$$

Thus this scorer evaluates whether support mass is allocated to the same keys, not whether the absolute total row count is identical. Total mass is retained as a diagnostic only.

Rates, shares, and proportions. Rates, probabilities, shares, proportions, and missingness rates have a fixed bounded scale. For values in $[0, 1]$, the score is one minus mean absolute error (Willmott & Matsuura, 2005):

$$S_{\text{rate}} = 1 - \frac{1}{|K|} \sum_{k \in K} |r_k - s_k|.$$

For scalar rates, K contains one implicit key. Percent-scale outputs are converted only when the query metadata explicitly declares that scale.

Top-k rankings. Ranking queries compare ordered identities, not attached magnitudes. The primary score is finite rank-biased overlap, introduced for comparing indefinite rankings, with persistence $p = 0.9$ (Webber et al., 2010):

$$S_{\text{rank}} = (1 - p) \sum_{d=1}^{D} p^{d-1} A_d + p^D A_D,$$

where A_d is the overlap fraction between the real and synthetic prefixes of length d, and D is the requested ranking depth or the observed comparison depth. Identical rankings score 1 and disjoint non-empty rankings score 0. Queries with ambiguous tie-breaking are not used for official ranking scores unless the SQL or template defines a deterministic order.

E.3 Aggregation Across Queries

Query scores are first computed at the valid query level. Family-, dataset-, and model-level summaries then average the corresponding valid query scores in the reported group. Invalid queries are excluded from the numerical mean but are counted separately, so a high score is interpreted together with its valid-query coverage. When scorer-type summaries are reported, the benchmark uses macro means within each semantic scorer type; when an overall query score is reported, it is the mean over valid query scores in the stated reporting group.

E.4 Classical Distance-Based Fidelity

The classical distance-based baseline compares the real training split against the synthetic output at the table-distribution level. The held-out test split is reserved for future downstream-task studies and is not used in this block.

Categorical distributions. For categorical columns, the benchmark reports Jensen–Shannon distance and total variation distance between aligned value distributions. Jensen–Shannon distance is computed as the square root of Jensen–Shannon divergence with logarithm base 2 (Lin, 1991). Total variation distance is

$$D_{\text{TV}}(P, Q) = \frac{1}{2} \sum_{x} |P(x) - Q(x)|.$$

We use the standard probability-metric interpretation of total variation (Gibbs & Su, 2002).

Numerical distributions. For numerical columns, the benchmark reports Kolmogorov–Smirnov distance and a normalized Wasserstein distance. Kolmogorov–Smirnov distance is the maximum gap between empirical cumulative distribution functions over the combined value grid (Massey, 1951). The normalized Wasserstein distance is computed from mean absolute quantile differences on a fixed quantile grid, normalized by the real-column range with a standard-deviation fallback, and capped to $[0, 1]$. Wasserstein distance is used in its modern optimal-transport sense (Peyré & Cuturi, 2019).

Distance-derived fidelity. For paper-facing comparisons, each lower-is-better distance is converted to higher-is-better form as $1 - D$. The overall classical fidelity score is the mean of available Jensen–Shannon, Kolmogorov–Smirnov, total variation, and normalized Wasserstein components.

Generator	Query Fidelity						Cost	
	Overall ↑	Subgroup ↑	Conditional ↑	Tail / Rarity ↑	Missingness ↑	Cardinality / Range ↑	Train min. ↓	Gen. s/1k ↓
REAL	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	—	—
RealTabFormer	$\mathbf{0.76}_{\pm 0.23}$	$\mathbf{0.88}_{\pm 0.17}$	$\mathbf{0.75}_{\pm 0.26}$	$\mathbf{0.83}_{\pm 0.20}$	$\mathbf{0.81}_{\pm 0.26}$	$\mathbf{0.74}_{\pm 0.26}$	54.93	10.45
ARF	$\mathbf{0.64}_{\pm 0.23}$	$\mathbf{0.85}_{\pm 0.22}$	$\mathbf{0.67}_{\pm 0.29}$	$\mathbf{0.77}_{\pm 0.21}$	$\mathbf{0.75}_{\pm 0.19}$	$\mathbf{0.60}_{\pm 0.27}$	9.22	0.84
BayesNet	$\mathbf{0.60}_{\pm 0.24}$	$\mathbf{0.80}_{\pm 0.22}$	$\mathbf{0.64}_{\pm 0.27}$	$\mathbf{0.69}_{\pm 0.25}$	$0.69_{\pm 0.20}$	$0.56_{\pm 0.28}$	1.10	1.92
TabDiff	$0.59_{\pm 0.20}$	$0.72_{\pm 0.27}$	$0.51_{\pm 0.29}$	$0.69_{\pm 0.24}$	$0.55_{\pm 0.26}$	$\mathbf{0.58}_{\pm 0.22}$	7.56	1.60
TabSyn	$0.57_{\pm 0.24}$	$0.73_{\pm 0.32}$	$0.52_{\pm 0.28}$	$0.67_{\pm 0.27}$	$0.55_{\pm 0.26}$	$0.55_{\pm 0.26}$	23.07	0.71
TabbyFlow	$0.57_{\pm 0.23}$	$0.66_{\pm 0.32}$	$0.49_{\pm 0.32}$	$0.67_{\pm 0.25}$	$0.57_{\pm 0.25}$	$0.54_{\pm 0.26}$	6.67	3.19
CTGAN	$0.55_{\pm 0.23}$	$0.66_{\pm 0.28}$	$0.52_{\pm 0.26}$	$0.62_{\pm 0.23}$	$\mathbf{0.75}_{\pm 0.15}$	$0.51_{\pm 0.28}$	9.08	0.94
TVAE	$0.54_{\pm 0.24}$	$0.65_{\pm 0.27}$	$0.49_{\pm 0.27}$	$0.60_{\pm 0.26}$	$0.65_{\pm 0.18}$	$0.50_{\pm 0.27}$	4.12	1.89
TabPFGen	$0.53_{\pm 0.24}$	$0.71_{\pm 0.29}$	$0.53_{\pm 0.29}$	$0.69_{\pm 0.23}$	$0.71_{\pm 0.21}$	$0.49_{\pm 0.28}$	0.00	45.29
TabDDPM	$0.46_{\pm 0.27}$	$0.52_{\pm 0.36}$	$0.37_{\pm 0.31}$	$0.57_{\pm 0.31}$	$0.56_{\pm 0.34}$	$0.45_{\pm 0.28}$	2.80	6.22
ForestDiffusion	$0.44_{\pm 0.23}$	$0.56_{\pm 0.31}$	$0.41_{\pm 0.29}$	$0.60_{\pm 0.28}$	$0.58_{\pm 0.22}$	$0.39_{\pm 0.24}$	7.25	0.72

Table 18: **Overall benchmark summary across 11 synthetic tabular generative models averaged over 49 datasets.** ↑ means higher-is-better and ↓ means lower-is-better. REAL is the self-comparison reference row, and top generative models are highlighted as **First**, **Second**, and **Third**. The last two columns report median training time in minutes and median generation time in seconds per 1k generated rows.

F Overall Evaluation Analysis

F.1 Overall Benchmark Summary

F.2 Overall Query Score Heatmap

Dataset	ARF	Bayes	CTGAN	F-Diff	RTF	T-Flow	T-DDPM	T-Diff	TPF	T-Syn	TVAE
c1	0.970	0.982	0.844	0.823	0.995	0.915	0.775	TF	0.909	0.925	0.883
c2	0.932	0.974	0.867	0.953	0.949	0.874	0.063	0.872	0.774	0.921	0.949
c3	0.508	0.458	0.442	0.245	0.294	0.266	0.268	0.269	0.371	0.292	0.297
c4	0.982	0.937	0.879	0.776	0.987	0.956	0.977	0.970	0.860	0.926	0.966
c5	0.982	0.933	0.921	0.742	0.986	0.936	0.938	0.925	0.946	0.928	0.917
c6	0.685	0.677	0.542	0.448	0.888	0.585	0.419	0.566	0.691	0.599	0.571
c7	0.957	0.971	0.877	0.052	0.990	0.929	0.964	0.922	0.726	0.052	0.865
c8	0.943	0.974	0.882	0.973	0.995	0.926	0.628	0.888	0.994	0.944	0.869
c9	0.276	0.185	0.126	0.099	0.893	0.465	0.141	0.459	0.207	0.434	0.137
c10	0.983	0.887	0.896	0.969	1.000	0.931	0.963	0.840	0.814	0.864	0.658
c11	0.992	0.987	0.923	0.899	0.998	0.982	0.977	0.977	0.897	0.989	0.985
c12	0.816	0.840	0.778	0.596	0.932	0.039	TF	TF	0.671	0.748	0.792
c13	0.860	0.678	0.467	0.603	0.859	0.731	0.526	0.651	0.081	0.802	0.474
c14	0.905	0.894	0.709	0.391	0.857	0.568	0.156	0.596	0.881	0.566	0.399
c15	0.858	0.773	0.671	0.544	0.926	0.649	TF	0.653	0.797	0.654	0.526
c16	0.742	0.563	0.633	0.381	0.790	0.280	TF	0.476	TF	0.469	0.481
c17	0.668	0.667	0.660	0.235	0.366	0.214	0.098	0.290	0.614	0.284	0.511
c18	0.691	0.508	0.497	0.261	0.806	0.290	TF	TF	TF	0.170	0.456
c19	0.830	0.338	0.490	0.174	0.279	0.162	TF	0.178	0.637	0.263	0.410
m1	0.475	0.674	0.561	0.465	0.582	0.588	0.411	0.614	0.496	0.577	0.419
m2	0.752	0.715	0.660	0.506	0.583	0.643	0.460	TF	0.723	0.701	0.608
m3	0.740	0.622	0.526	0.441	0.962	0.704	TF	0.728	0.590	0.782	0.536
m4	0.667	0.656	0.623	0.504	0.853	0.536	0.239	0.702	0.637	0.731	0.592
m5	0.608	0.491	0.405	0.424	0.911	0.737	0.425	0.668	0.571	0.716	0.360
m6	0.880	0.431	0.327	0.246	0.767	0.614	0.431	0.683	0.461	0.696	0.337
m7	0.650	0.689	0.631	0.650	0.821	0.701	0.676	0.708	0.673	0.710	0.591
m8	0.675	0.574	0.576	0.589	0.952	0.757	0.305	0.301	0.616	0.316	0.573
m9	0.833	0.826	0.804	0.577	0.913	0.588	0.668	0.653	0.733	0.765	0.667
m10	0.551	0.765	0.652	0.606	0.806	0.617	0.342	0.641	0.598	0.651	0.689
m11	0.627	0.503	0.641	0.496	0.896	0.687	0.755	0.805	0.514	0.708	0.577
n1	0.424	0.255	0.096	0.255	0.452	0.363	0.433	0.570	0.110	0.406	0.091
n2	0.493	0.374	0.391	0.332	0.742	0.417	0.544	0.423	0.450	0.505	0.396
n3	0.287	0.296	0.154	0.225	0.832	0.474	0.258	0.493	0.237	0.528	0.156
n4	0.367	0.350	0.272	0.329	0.790	0.391	0.286	0.429	0.368	0.444	0.293
n5	0.457	0.428	0.366	0.380	0.500	0.468	0.148	0.454	0.419	0.410	0.381
n6	0.299	0.631	0.346	0.260	0.809	0.558	0.558	0.538	0.301	0.559	0.752
n7	0.515	0.571	0.414	0.473	0.350	0.547	0.180	0.503	0.483	0.546	0.469
n8	0.430	0.467	0.395	0.378	0.544	0.325	TF	0.422	0.286	0.424	0.356
n9	0.356	0.879	0.818	0.343	0.944	0.705	0.726	0.709	0.313	0.714	0.835
n10	0.440	0.339	0.442	0.330	0.325	0.457	0.283	0.463	0.454	0.394	0.439
n11	0.405	0.427	0.250	0.387	0.518	0.391	0.372	0.404	0.388	0.413	0.290
n12	0.825	0.372	0.844	0.279	0.956	0.581	0.627	0.589	0.337	0.777	0.861
n13	0.234	0.276	0.160	0.148	TF	0.272	0.079	TF	0.254	0.202	0.159
n14	0.353	0.376	0.268	0.276	0.381	TF	0.267	0.428	0.305	0.177	0.252
n15	0.694	0.865	0.385	0.489	0.857	0.576	0.376	0.743	0.559	0.730	0.689
n16	0.281	0.224	0.267	0.174	0.211	0.274	0.166	0.266	0.269	0.268	0.217
n17	0.426	0.628	0.573	0.232	0.653	TF	0.593	0.616	0.231	0.494	0.570
n18	0.394	0.307	0.402	0.315	0.996	0.437	TF	0.436	0.284	0.420	0.322
n19	0.435	0.407	0.591	0.422	0.710	0.477	0.394	0.636	0.326	TF	0.685

Table 19: Dataset-by-generative-model overall query score heatmap for the v11.2.2_glm-5.3 workload. TF indicates technical failure due to unsupported generation or prohibitively long runtime.

G Extended Experiments

G.1 Stability Across Query-Workload Rebuilds

This appendix branch is reserved for additional analysis artifacts that do not fit into the main benchmark summary, including comparisons between benchmark-derived queries and TabQueryBench queries.

Generator	Haiku-4.5	GLM-5.3F-1	GLM-5.3F-2	GLM-5.3F-3	GLM-5.3-1	GLM-5.3-2	GLM-5.3-3	Mean	SD	Range
RealTabFormer	0.779	0.777	0.780	0.766	0.766	0.773	0.761	0.772	0.008	0.019
ARF	0.653	0.657	0.669	0.664	0.653	0.662	0.649	0.658	0.007	0.020
BayesNet	0.615	0.625	0.625	0.615	0.618	0.623	0.610	0.619	0.006	0.016
TabDiff	0.596	0.596	0.614	0.607	0.593	0.606	0.591	0.600	0.009	0.023
TabSyn	0.569	0.579	0.593	0.584	0.582	0.584	0.577	0.581	0.007	0.024
TabbyFlow	0.581	0.582	0.589	0.579	0.573	0.578	0.576	0.580	0.005	0.016
CTGAN	0.546	0.561	0.558	0.544	0.543	0.561	0.549	0.552	0.008	0.018
TabPFGen	0.544	0.549	0.539	0.531	0.542	0.544	0.525	0.539	0.008	0.024
TVAE	0.541	0.561	0.555	0.541	0.544	0.552	0.541	0.548	0.008	0.020
TabDDPM	0.438	0.467	0.477	0.478	0.467	0.475	0.469	0.467	0.014	0.041
ForestDiffusion	0.451	0.461	0.464	0.455	0.460	0.459	0.448	0.457	0.006	0.016

Table 20: **Query-score stability across six AI workload builders.** Scores are averaged over the 49-dataset benchmark set for each generator, excluding `dataevolve`. Haiku-4.5 denotes `v11.1.4`; Flash-1/2/3 denote `v11.2.1`, `v11.2.1-2`, and `v11.2.1-3`; GLM-1/2/3 denote `v11.2.2`, `v11.2.2-2`, and `v11.2.2-3`.

G.2 Structural Coverage of the Full Generated Workload

G.2.1 Purpose and comparison universes

The query-grounding pipeline is designed to produce executable, schema-aware queries, but executable queries can still cover only a narrow part of the structural space found in public analytical workloads. We therefore perform a separate workload-level audit. This audit evaluates the query workload itself, rather than synthetic-table fidelity or agreement between query answers.

The source corpus contains 334 queries from twelve public sources. Of these, 320 parse successfully and yield 286 unique normalized Detail-Level shapes. For the main result, we use the complete `v11.2.2_glm-5.3` generated workload: all 2,763 valid generated SQL queries and its 44 unique normalized shapes, including queries without an explicit public-source provenance mapping. This is the result reported as the full generated workload.

For comparison with the earlier source-grounded protocol, we also retain a provenance-matched subset. This subset contains 229 generated queries and 25 unique shapes whose template provenance resolves to one of the public source workloads `S01`–`S12`. The remaining 2,534 valid generated queries are not invalid or unparseable; they are simply outside that provenance filter. The matched subset is therefore reported only as a sensitivity analysis, not as the complete generated workload.

G.2.2 Detail-Level normalization and TSED construction

Before computing TSED, each SQL query is converted to a schema-masked Detail-Level skeleton. Dataset names, column names, and other schema-specific identifiers are replaced by placeholders, while SQL operators, clause roles, aggregate functions, nesting, and ordering structure are retained. Identical skeleton strings are deduplicated before the frequency-independent comparison, so each unique shape receives one equal weight. The audit therefore compares 286 source shapes against 44 generated shapes, rather than comparing all $320 \times 2{,}763$ SQL pairs.

Each skeleton is parsed into an ordered labeled tree. The primary parser is SQLGlot (Mao & SQLGlot Contributors, 2026). Every SQL expression becomes a node labeled by its expression class, such as `SELECT`, `WHERE`, `GROUP`, `SUM`, or `WINDOW`. Each populated expression argument is represented by an explicit role node, such as `ARG:WHERE` or `ARG:GROUP`, and child expressions preserve the original SQL tree order. Lists of expressions preserve their order as multiple children. Scalar values are represented by typed labels rather than raw database-specific values. If SQLGlot cannot parse a dialect-specific skeleton, the implementation uses a deterministic parenthesis-aware token tree as a conservative fallback and records the backend used for that pair.

Comparison universe	Generated SQL	Generated shapes	Source → Generated	Generated → Source
Full generated workload	2,763	44	0.478	0.702

Table 21: Full-workload structural coverage. The main result retains every valid generated SQL query from `v11.2.2_glm-5.3`. Scores are computed over unique normalized Detail-Level shapes.

G.2.3 TSED COMPUTATION

For a source tree T_s and a generated tree T_g, we compute the ordered tree-edit distance with APTED (Song et al., 2024). The allowed operations are node deletion, node insertion, and node rename. All three operations have fixed unit cost:

$$c_{\text{delete}} = c_{\text{insert}} = c_{\text{rename}} = 1.$$

The costs are not tuned on the benchmark data. Let $d_{\text{APTED}}(T_s, T_g)$ be the minimum total cost over all valid edit sequences, and let n_s and n_g be the numbers of nodes in the two trees. The normalized edit distance and TSED similarity are:

$$d_{\text{norm}}(T_s, T_g) = \min\left(1, \frac{d_{\text{APTED}}(T_s, T_g)}{\max(n_s, n_g)}\right), \qquad \text{TSED}(T_s, T_g) = \max\left(0, 1 - d_{\text{norm}}(T_s, T_g)\right).$$

Thus, identical normalized skeletons have TSED $= 1$, and increasingly different ordered SQL structures receive values closer to 0. The normalization by the larger tree size makes scores comparable for trees of different lengths. TSED measures structural relatedness of the normalized SQL trees; it does not establish semantic equivalence, equality of query answers, or equality of database execution plans.

Exact computation and caching. The full cross-product contains $286 \times 44 = 12{,}584$ possible shape pairs. We parse each source and generated skeleton once and persist every evaluated pair in a resumable SQLite cache. To reduce runtime without changing the answer, we use the node-count bound

$$\text{TSED}(T_s, T_g) \le \frac{\min(n_s, n_g)}{\max(n_s, n_g)}.$$

This follows because at least $|n_s - n_g|$ insertions or deletions are necessary when the tree sizes differ. A pair is skipped only when this upper bound cannot improve the current best match for either direction. The remaining pairs are evaluated with the exact APTED computation. In the full run, $3{,}225$ pair distances were computed exactly, and the other were skipped by this bound; no approximate distance was substituted for a skipped pair.

G.2.4 DIRECTIONAL AGGREGATION AND RESULTS

Let S be the set of unique source shapes and G the set of unique generated shapes. We report two directional nearest-neighbor summaries:

$$\text{S}{\to}\text{G} = \frac{1}{|S|}\sum_{s \in S} \max_{g \in G} \text{TSED}(s, g), \qquad \text{G}{\to}\text{S} = \frac{1}{|G|}\sum_{g \in G} \max_{s \in S} \text{TSED}(g, s).$$

The first score measures source-space coverage: for every source shape, a closest generated shape is selected. The second measures containment: for every generated shape, a closest source shape is selected. We summarize them with the descriptive bidirectional TSED-Chamfer score

$$\text{Chamfer}_{\text{TSED}} = \frac{1}{2}\left(\text{S}{\to}\text{G} + \text{G}{\to}\text{S}\right).$$

This symmetric summary is not merged with query-centric fidelity because the two experiments evaluate different objects.

Interpretation and limitations. For the full generated workload, Source → Generated is 0.478 and Generated → Source is 0.702. At threshold TSED ≥ 0.5, $96/286$ source shapes have a generated counterpart, while $45/49$ generated shapes have a source counterpart. The asymmetry indicates that the generated workload is mostly inside the structural envelope represented by the sources, but it covers only part of the broader source space. The largest coverage gap comes from long TPC-DS

queries with common-table expressions, nested subqueries, window functions, and set operations that are not well represented by the current single-table template inventory.

The result is frequency-independent after shape deduplication: a shape appearing many times does not receive more weight in the directional means. This makes the audit complementary to frequency-sensitive workload metrics. It also means that the score should not be interpreted as a fidelity score for the synthetic tables themselves. Documentation sources with one or two parseable queries, incomplete SQL snippets, and source-side parse failures are reported in the accompanying audit files rather than silently converted to zero similarity. The complete directional matrices, pair cache, parse audit, and both comparison universes are frozen in the corresponding `Bench2Tem` experiment bundle.

H Discussion and Future Works

H.1 Discussion and Limitation

Query-centric fidelity and distance-based fidelity. Query-centric fidelity is not intended to replace distance-based fidelity. The two views measure different aspects of synthetic data quality. Distance-based metrics summarize global resemblance over columns or distributions, while query-centric fidelity asks whether the synthetic table preserves the answers to analytical operations that downstream users are likely to run. Our results show why both views are needed. This makes query-centric fidelity a complementary evaluation layer rather than a substitute for statistical similarity. In practice, the right evaluation depends on the downstream use case. Users who need mock data for coarse schema exploration may care more about global resemblance, while users who need data system testing or analytics on synthetic data may need stronger query-centric fidelity.

Toward query-aware synthetic-data generation. TabQueryBench also suggests a model-development direction: future tabular generative models can use query families as training, validation, or model-selection signals. The family-level breakdown identifies where current models lose analytical structure, rather than only reporting an aggregate query score. For example, high-cardinality results point to the need for better discrete support preservation. Tail and rarity results point to the need for better rare-region generation. Local conditional results point to the need for models that preserve filtered subpopulation structure. These targets are concrete because they correspond to executable SQL queries and measurable query answers. A query-aware generative model could therefore optimize not only global distributional resemblance, but also take these analytical queries as part of the loss function or reward signals so that the generated table can support these operations.

Multi-table query-centric evaluation. The current benchmark is designed to support the single-table setting, where reusable analytical query patterns can be grounded across heterogeneous schemas through schema-aware realization. A natural next step is multi-table evaluation. In that setting, synthetic data must preserve not only column distributions and single-table query answers, but also joins, foreign-key structure, join selectivity, fanout behavior, and query answers across related tables. Multi-table query-centric fidelity would extend the same principle used in TabQueryBench: synthetic data should preserve the analytical operations that downstream users are likely to run. This extension would be especially important for analytical questions which cross table boundaries.

Privacy vs. fidelity. Query-centric fidelity also has a natural connection to privacy evaluation, where synthetic-data releases are often judged through privacy-utility tradeoffs, differential-privacy mechanisms, or threat-specific leakage analyses (Tao et al., 2021; Höllig & Geierhos, 2025; Personal Data Protection Commission Singapore, 2024; Tran et al., 2025). The same structural assessors that measure whether a synthetic table preserves analytical behavior can reveal how much rare support, local subgroup structure, or high-cardinality information survives generation. Query-centric fidelity is not a formal privacy metric, and it is not a substitute for membership-inference, attribute-inference, or differential-privacy analyses. Instead, it highlights a fidelity-privacy tension that users must interpret for their specific settings. In particular, privacy risk is often contextual: some releases may tolerate accurate aggregates, while others may treat specific query outputs as sensitive information. When query outputs are the objects users want to protect, query-centric evaluation provides a useful diagnostic because the same query structure can be used to measure how much protected signal leaks through the synthetic table. A synthetic table that preserves rare values or high-cardinality keys may be more useful for analytics, but such preservation should be read together with privacy reports and threat models.

LLM-assisted SQL grounding. TabQueryBench uses an LLM only inside the constrained template-to-SQL realization step: the profiling, binding, and validation are deterministic, and the validation rejects invalid SQL, incompatible realizations, and queries that fail execution. Our stability study suggests that the resulting conclusions are broadly stable across regenerated query sets, but LLM-assisted SQL grounding remains a source of benchmark-instrument variance. The current study reruns the same grounding pipeline, but it does not yet compare multiple SQL realization models or multiple prompting policies. Future work could audit SQL grounding across different LLMs, compare family-level and model-level rankings under these alternative realizations, and include additional human or rule-based checks for ambiguous schema bindings.