Sparks of Tabular Reasoning via Text2SQL Reinforcement Learning

Josefa Lia Stoisser [*]
Novo Nordisk [†]
ofsr@novonordisk.com

Marc Boubnovski Martell [*]
Novo Nordisk[†]
mbvk@novonordisk.com

Julien Fauqueur
Novo Nordisk[†]
jlzf@novonordisk.com

Abstract

This work reframes the Text-to-SQL task as a pathway for teaching large language models (LLMs) to reason over and manipulate tabular data—moving beyond the traditional focus on query generation. We propose a two-stage framework that leverages SQL supervision to develop transferable table reasoning capabilities. First, we synthesize detailed chain-of-thought (CoT) traces from real-world SQL queries, providing step-by-step, clause-level supervision that teaches the model how to traverse, filter, and aggregate table fields. Second, we introduce a Group Relative Policy Optimization (GRPO) reinforcement learning objective that connects SQL execution accuracy to generalizable reasoning by encouraging steps that extend beyond task-specific syntax and transfer across datasets.

Empirically, our approach improves performance on standard Text-to-SQL benchmarks and achieves substantial gains on reasoning-intensive datasets such as BIRD, CRT-QA and Tablebench, demonstrating enhanced generalization and interpretability. Specifically, the distilled-quantized LLaMA-8B model achieved a 34% relative increase in exact match scores on CRT-QA when trained on Text-to-SQL tasks, while Qwen-2.5-7B achieved a 10% and Qwen-2.5-14B a 6% relative increase. These results suggest that SQL can serve not only as a target formalism but also as an effective scaffold for learning robust, transferable reasoning over structured data.

1 Introduction

Recent advancements in LLMs have substantially improved performance on Text-to-SQL tasks, translating natural language into executable SQL queries over relational databases (Gao et al., 2023).

Progress has been driven primarily by supervised fine-tuning (SFT) on SQL-focused datasets (e.g., Spider (Yu et al., 2018), BIRD (Li et al., 2023)) or prompt-based adaptation (Sun et al., 2023). However, these methods often narrowly optimize for syntactic correctness or execution accuracy, overlooking deeper reasoning over underlying data structures—resulting in degraded performance in real-world settings (Liu et al., 2024; Nascimento et al., 2025).

This highlights a broader issue: Text-to-SQL is frequently treated as a standalone task, rather than as a facet of the more general challenge of reasoning over tabular data (Liu et al., 2024). SQL, as a formal language, provides a vehicle for structured reasoning over relational tables; thus, models generating SQL should ideally also support broader forms of table-based question answering (e.g., TabFact (Chen et al., 2019), WikiTQ (Pasupat and Liang, 2015), FinQA (Chen et al., 2021)).

Yet, models fine-tuned exclusively for Text-to-SQL often exhibit degraded performance on related tasks, suggesting overfitting to SQL-specific patterns at the expense of flexible reasoning (Abhyankar et al., 2024). Methods like H-STAR (Abhyankar et al., 2024) integrate symbolic and semantic reasoning for improved table comprehension, while Plan-of-SQLs (POS) (Brugere et al., 2024) emphasize interpretability and QA performance. However, both approaches tend to bias the model toward SQL-centric reasoning, potentially limiting generalization (Nascimento et al., 2025). Inspired by DeepSeek-R1 (Guo et al., 2025), we explore whether reinforcement learning (RL) can foster emergent reasoning capabilities that connect Text-to-SQL with general tabular QA.

We propose a two-stage approach depicted in Figure 1. First, we introduce a supervised fine-tuning phase leveraging synthetically generated CoT reasoning traces to provide structured guidance between the natural language input and its corresponding SQL representation. Unlike SynSQL-2.5 (Li et al., 2025b), which emphasizes data scale,

[*]Equal contribution
[†]One Pancras Square, Pancras Rd, London N1C 4AG

Proceedings of the 4th Table Representation Learning Workshop, pages 229–240

our approach focuses on generating high-quality CoT traces grounded in real data points. Second, we apply GRPO (Shao et al., 2024), a reinforcement learning method that compares multiple candidate outputs, aligning SQL execution accuracy and query structure with broader reasoning fidelity.

While prior work (e.g., Reasoning-SQL (Pourreza et al., 2025), SQL-R1 (Ma et al., 2025)) has applied RL to SQL generation, our key contribution lies in bridging Text-to-SQL with general tabular reasoning. We show that models trained with our two-stage framework outperform SFT baselines not only on SQL benchmarks but also on reasoning-intensive QA datasets such as CRT-QA (Zhang et al., 2023) and Tablebench (Wu et al., 2025), illustrating that SQL generation, when properly framed, can serve as a foundation for broader structured data reasoning.

Our key contributions are:

1. **Synthetic CoT Supervision:** We present a method for generating synthetic reasoning traces tailored to the SQL domain, offering structured and interpretable supervision during fine-tuning. The synthetic data is made publicly available[1].

2. **Reinforcement Learning with GRPO for Generalization:** We apply GRPO not only to improve SQL execution accuracy, but also to regularize model behavior toward more generalizable table reasoning.

3. **Empirical Evidence of Cross-Task Gains:** Our two-stage method improves performance on standard Text-to-SQL benchmarks while enhancing reasoning ability on diverse QA datasets such as CRT-QA and Tablebench.

The training and evaluation code is made publicly available[2].

2 Background

2.1 Reasoning in Language Models

LLMs have demonstrated strong capabilities in general-purpose reasoning tasks, including arithmetic, logic, and multi-step decision-making. These capabilities are often enhanced by prompting techniques, tool integration, and reinforcement learning (Jaech et al., 2024; Guo et al., 2025). A growing line of work has focused on intermediate reasoning structures, such as CoT prompting, which guide models through decomposed, interpretable inference steps (Zhao et al., 2025).

In particular, long-form CoT reasoning—requiring detailed, iterative solutions—has shown benefits in domains like mathematics, program synthesis, and multi-hop question answering (Team et al., 2025). Unlike short-form CoT, long-form reasoning involves planning, reflection, and consistency across intermediate steps. Recent studies have shown that such behavior can be learned through data-efficient supervised fine-tuning and parameter-efficient adaptation methods such as low-rank updates (LoRA) (Li et al., 2025a). Beyond training-time learning, test-time methods like self-consistency and re-ranking over multiple generations have been shown to improve reasoning reliability (Wei et al., 2022; Wang et al., 2022).

Complementary to these approaches, reinforcement learning has been explored as a way to promote reasoning beyond imitation, allowing models to discover extended inference patterns through reward-driven optimization (Qin et al., 2024; Chen et al., 2025; Shinn et al., 2023).

2.2 LLMs on Text-to-SQL

Mapping natural language to executable SQL involves three principal challenges: interpreting user intent, understanding database schema, and generating syntactically and semantically correct queries (Hong et al., 2024; Stoisser et al., 2025). LLMs have shown strong performance on this task, supported by progress in semantic parsing and schema linking (Liu et al., 2024; Shi et al., 2020). Recent work continues to refine LLMs across subcomponents of the task, including question understanding (Pourreza and Rafiei, 2023), schema comprehension (Yuan et al., 2025), and SQL generation (Lee et al., 2024).

To move beyond supervised fine-tuning, reinforcement learning has been proposed as a means of aligning model behavior with downstream performance objectives (Jiang et al., 2025). GRPO compares multiple candidate outputs, offering a denser learning signal that mitigates the limitations of sparse or binary rewards (Pourreza et al., 2025). SQL-R1 builds on this idea by integrating reinforcement learning with synthetic CoT supervision, achieving competitive results on benchmarks such

[1] https://huggingface.co/datasets/jls205/synthetic_cot_traces_clinton/blob/main/cot.csv

[2] https://github.com/josefastoisser/sparks_of_tabular_reasoning

as BIRD and WikiSQL (Ma et al., 2025; Li et al., 2025b).

These approaches suggest that supervision grounded in SQL execution can serve not only as a means of training for query generation, but as a proxy for inducing structured reasoning in LLMs.

2.3 LLMs on Tabular Question Answering

LLMs have increasingly been applied to question answering over structured tabular data—a task that combines natural language understanding with symbolic reasoning. In the typical formulation, models receive a serialized table and a natural language query, and are tasked with producing an accurate answer. While this setting is straightforward, it presents several challenges, including query intent disambiguation, context-aware retrieval, numerical reasoning, and robust handling of multi-turn interactions (Pal et al., 2023).

Recent work has introduced frameworks that extend LLM capabilities in this domain. The Chain-of-Command approach, for instance, reformulates user queries into structured commands that guide table interaction (Zha et al., 2023). Other strategies improve retrieval through query-based sampling or adaptive search mechanisms (Sui et al., 2023). Multi-turn dialogue settings have also gained attention, where task decomposition and iterative refinement have shown improvements in reasoning depth and consistency (Yu et al., 2025).

Benchmarks such as CRT-QA provide a foundation for evaluating LLM performance on table reasoning tasks (Zhang et al., 2023; Ashury-Tahan et al., 2025). These settings demand not only the ability to parse structured inputs, but also to integrate logical, numerical, and contextual cues across diverse formats. Together, these developments suggest that tabular question answering offers a rich and challenging testbed for evaluating the reasoning capabilities of LLMs.

3 Methodology

Our methodology is outlined in Figure 2, where we see the breakdown into 6 steps.

3.1 Generating Synthetic Reasoning Traces for SQL Tasks

In the first stage, we construct synthetic CoT traces for Text-to-SQL questions using a structured prompting pipeline. The core generation process employs a LLMs trained on 25 diverse datasets (see Appendix A), following the methodology of Boubnovski et al. (2025). Specifically, we prompt the o3-mini model to answer SQL-related questions while producing intermediate reasoning steps in natural language as shown in Appendix B.1. A second language model is used as a verifier to assess both the correctness of the final answer and the internal reasoning trace (prompt details in Appendix B.2).

This framework yields a dataset of 3,174 examples containing only correctly reasoned outputs, which we use as high-quality supervision during model fine-tuning.

3.2 Training and Reward Design

To promote tabular reasoning in large-scale language models for natural language to SQL tasks, we adopt a two-stage training approach inspired by DeepSeek-R1 (Guo et al., 2025). In the first stage, we apply supervised fine-tuning on synthetic reasoning traces generated by o3-mini. This step improves the model's ability to follow instructions, decompose complex tasks, and generate interpretable outputs within the SQL domain.

In the second stage, we apply reinforcement learning to refine the model's reasoning behavior and align it more closely with execution-based performance objectives. This training encourages consistency between intermediate reasoning steps and the final executable output, enabling the model to generalize beyond dataset-specific patterns in the data.

3.2.1 Reinforcement Learning

To refine model behavior beyond supervised learning, we employ GRPO, a reinforcement learning method originally introduced in Deepseekmath (Shao et al., 2024). This approach enables more stable optimization by comparing multiple outputs for the same input and assigning relative rewards. By evaluating groups of candidate outputs rather than individual sequences in isolation, the model receives finer-grained feedback that encourages consistent and generalizable reasoning.

Formally, for a given natural language question q and its associated database schema, the model generates a set of G candidate SQL queries $\{o_1, o_2, \ldots, o_G\}$. Each candidate is scored using a task-specific reward function, and the relative advantage A_i is computed for each output. The optimization objective is given by:

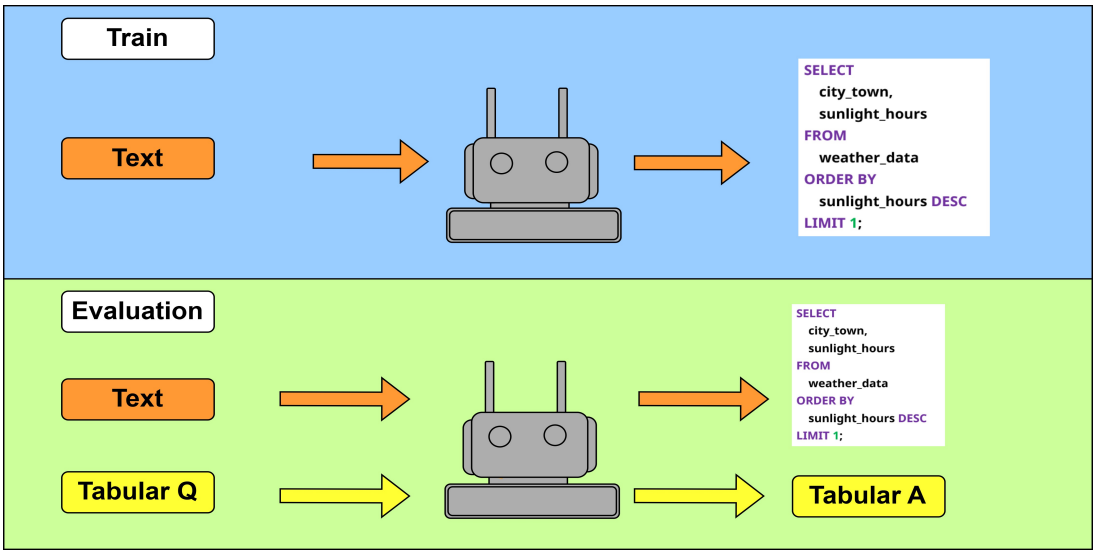

Figure 1: **Training on Text-to-SQL, Evaluating on Dual Tasks.** Our framework is trained solely on Text-to-SQL data, using structured supervision from CoT traces and reinforcement learning objectives. At evaluation time, we assess performance on both Text-to-SQL benchmarks and tabular question answering tasks. This setup tests whether SQL-centered training can induce reasoning capabilities that generalize beyond query generation to broader table-based inference.

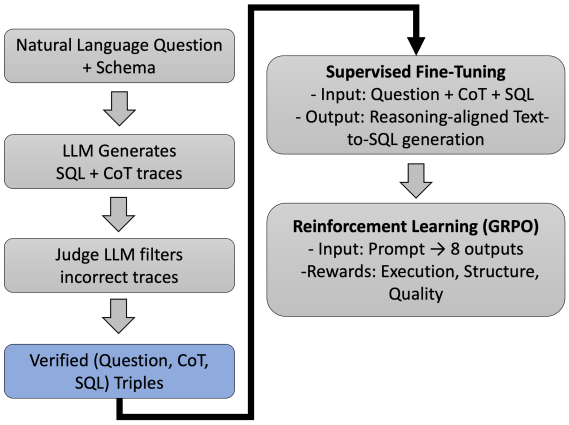

Figure 2: **Overview of the training pipeline.** Given a natural language question and schema, we generate SQL queries and CoT traces using a pretrained o3-mini. A second model filters these outputs by judging correctness and consistency. Verified traces are used for supervised fine-tuning on Clinton, followed by GRPO on the BIRD dataset. This two-stage training process promotes generalization across both SQL generation and tabular question answering.

$$J_{GRPO}(\Theta) = E\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i,\right.\right.$$
$$\left.\left.\text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\epsilon, 1+\epsilon\right)A_i\right)\right]$$
$$-\beta D_{KL}(\pi_\theta||\pi_{\text{ref}}) \quad (1)$$

Here, π_θ denotes the current policy, $\pi_{\theta_{\text{old}}}$ is the policy before the update, and π_{ref} is a frozen reference policy used for regularization. The hyperparameters ϵ and β control the clipping threshold and divergence penalty, respectively.

3.2.2 Reward Design

We define several reward functions tailored to the Text-to-SQL task, each capturing different dimensions of query quality. These rewards guide the optimization process during reinforcement learning with GRPO.

1. **Execution-Based Reward:** The primary objective in Text-to-SQL is to generate queries that execute to the correct result. Traditional binary execution rewards offer no gradient for near-correct predictions. To address this, we implement a reward function that leverages a language model to count orthographic changes—textual mutations between the predicted and reference queries, such as token insertions, deletions, or substitutions. The corresponding prompt can be found in B.3. The reward is computed as:

$$R_{\text{exec}} = \frac{1}{x+1}, \quad (2)$$

where x is the number of detected changes. This formulation provides a smoother feedback signal that penalizes incorrect queries

proportionally, even when they are close to correct.

2. **String Matching Reward:** This reward compares the predicted and gold SQL strings by identifying the longest contiguous matching subsequence. It is computed as the ratio of matching characters to the total number of characters across both sequences, thereby encouraging partial correctness even when queries are not exact matches.

3. **Component-Level Matching Reward:** To capture semantic equivalence beyond surface form, we compute overlap between query components such as `SELECT`, `WHERE`, and `GROUP BY` – using the F1 score as in the component matching metric (Yu et al., 2018). This allows the model to be rewarded for capturing the correct logical structure, even when query formatting varies.

4. **LLM Judge Reward with Classes:** Pretrained language models exhibit strong sensitivity to syntactic correctness and logical coherence. Building on the literature that utilizes pretrained language models to provide continuous rewards based on these criteria for SQL queries (Pourreza et al., 2025), we extend this approach to categorize model outputs into ordinal quality classes—*Very Bad*, *Bad*, *Average*, *Above Average*, *Good*, and *Excellent*, see Appendix B.4. This categorical scoring is adapted from Xin et al. (2024) and enables more interpretable and consistent supervision, particularly in filtering low-quality outputs during training.

All language model-based evaluations are performed using OpenAI's o3-mini model (Jaech et al., 2024), which serves as both a scorer and judge for reward construction.

4 Experiments

We design our experiments to investigate the following research questions:

- **RQ1:** How does the use of synthetic reasoning traces during supervised fine-tuning impact Text-to-SQL performance?
- **RQ2:** Can our two-stage framework—combining supervised fine-tuning and GRPO—facilitate the induction of transferable tabular reasoning capabilities?
- **RQ3:** Which reward functions in GRPO contribute most significantly to improved table-based reasoning?

4.1 Setup

Evaluation Benchmarks: We evaluate our framework across two primary tasks: Text-to-SQL and tabular question answering. For Text-to-SQL, we utilize the Clinton A and BIRD minidev [3] datasets. For tabular question answering, we evaluate performance on the Tablebench Fact Checking dataset (Wu et al., 2025), as it provides a comprehensive estimate of model understanding of tables across 18 fields. Additionally, to emphasize complex reasoning, we utilize the CRT-QA dataset (Zhang et al., 2023), which focuses on complex table-based reasoning, incorporating multi-step operations and informal reasoning techniques.

Evaluation Metrics: We employ task-appropriate evaluation metrics for each benchmark. For Text-to-SQL tasks, we report execution accuracy, defined as the exact match between the predicted and reference SQL query results. Given the limited access to the full database within Clinton, we utilize OpenAI's o3-mini model as a proxy for execution for this dataset, assessing query correctness based on structural and semantic alignment. For CRT-QA, we use Exact Match to compare the predicted answer with the ground truth. For Tablebench, we employ the ROUGE score as outlined in the original paper (Wu et al., 2025).

Training Settings: We utilize three base models: Qwen-2.5-7B-Instruct, Qwen-2.5-14B-Instruct, and a 4-bit quantized version of the distilled DeepSeek-R1-Distill LLaMA 8B model. This selection enables us to evaluate both distilled and quantized architectures, as well as smaller and larger models. For supervised fine-tuning, we use a learning rate of 2×10^{-4} and a batch size of 1. During reinforcement learning with GRPO, we fix the learning rate at 1×10^{-6}. Each GRPO training instance consists of a natural language question and its associated schema; for each prompt, the model generates 8 candidate completions used to compute group-based rewards. Further implementation details can be found in Appendix C.

[3] https://github.com/bird-bench/mini_dev

4.2 Tabular Aha-Moments

During reinforcement learning with GRPO, we observe instances of emergent tabular reasoning, which we term *Tabular Aha-Moments*. These moments, inspired by the *Aha Moment* concept from DeepSeek-R1 (Guo et al., 2025), occur when the model, provided only a natural language question and schema (but no table content), implicitly reconstructs the structure of the underlying table and uses this to solve the query. An example of this behavior is shown in Figure 4, where the model demonstrates schema-grounded inference without explicit tabular context during the training process.

When evaluated on tabular question answering tasks, the model often invokes SQL-like structures as intermediate reasoning tools—even when SQL output is not required. This is illustrated in Figure 3, where the model constructs an internal SQL representation to derive a binary answer. This reflects a bidirectional inductive bias: the model not only learns to generate SQL from questions but also learns to use SQL representations to support reasoning over tables. These findings highlight the potential for GRPO to induce transferable, structure-aware reasoning in LLMs.

4.3 Benefit of CoT Supervision

Table 1 reports the performance of our supervised models across Text-to-SQL and tabular question answering tasks. Comparing models fine-tuned with (SFT-CoT) and without (SFT) CoT supervision, we observe that including reasoning traces slightly reduces performance on the in-domain Clinton dataset, but improves generalization to unseen SQL benchmarks (BIRD) and table-based reasoning tasks (CRT-QA, Tablebench).

We attribute this to the inductive bias introduced by reasoning supervision: models exposed to intermediate inference steps are more likely to learn transferable patterns rather than overfitting to schema-specific templates. Moreover, fine-tuning with CoT traces provides a more structured initialization for reinforcement learning, ensuring that the GRPO stage begins from semantically grounded outputs.

CoT supervision yields markedly different gains for LLaMA and Qwen due to their architectural disparities. In our experiments, a distilled and quantized LLaMA model received a substantially larger performance boost from CoT supervision than the uncompressed Qwen model. We attribute this discrepancy to LLaMA's compressed nature: distillation and low-precision quantization reduce its representational capacity and can weaken its innate reasoning ability. Consequently, providing explicit step-by-step reasoning guidance during training allows LLaMA to compensate for these lost details, resulting in outsized improvements. In contrast, Qwen—being neither distilled nor quantized—retains a higher precision and fuller pretrained capacity for reasoning, which means it already performs strongly on complex tasks before CoT fine-tuning. As a result, Qwen's robust baseline reasoning ability leaves less headroom for dramatic gains. This contrast highlights that CoT supervision is especially critical for enhancing compressed models like LLaMA.

4.4 Text-to-SQL Performance

The combination of supervised fine-tuning with CoT (SFT-CoT) and GRPO yields marked improvements in Text-to-SQL performance. While the gains on the BIRD dataset—where GRPO was explicitly trained—are anticipated, the enhancement on the Clinton dataset is more notable. This indicates that GRPO not only fine-tunes models to specific tasks but also encourages broader SQL comprehension and reasoning capabilities, facilitating generalization within the Text-to-SQL domain.

In particular, the SFT-CoT + GRPO model shows a strong ability to generalize, demonstrating that models trained on real-world tasks can effectively perform even on data they haven't seen during training, provided they have a strong foundational understanding of SQL reasoning.

4.5 Zero-shot Question Answering Tabular Reasoning Performance

Table 1 demonstrates that our combined approach of SFT and GRPO, originally fine-tuned on Text-to-SQL data, also enhances tabular reasoning performance in zero-shot settings. Specifically, when evaluated on CRT-QA and Tablebench, we observe improved reasoning across the model, showcasing that the model's exposure to SQL structures helps it tackle general tabular question answering tasks even when SQL generation is not explicitly required.

The zero-shot performance is indicative of the transferability of the reasoning skills learned during SQL task training. By implicitly learning to reason over structured tables in the SQL framework, the model becomes better at navigating more com-

Model	Clinton (LLM-EXE)	Bird (EXE)	CRT-QA (EM)	Tablebench (Rouge)
o1	**60.7**	**28.3**	**61.3**	**64.4**
LLaMA Base	44.4	8.1	43.3	57.1
LLaMA SFT	**62.1↑ 17.7**	3.0	33.7	49.9
LLaMA SFT-CoT	56.3	9.1	47.7	57.2
LLaMA SFT-CoT-GRPO	57.0	**14.2↑ 6.1**	**58.1↑ 14.8**	**61.1↑ 4.0**
Qwen-2.5-7B-Instr Base	56.1	18.9	49.0	61.6
Qwen-2.5-7B-Instr SFT	**66.6 ↑ 10.5**	9.3	45.3	52.2
Qwen-2.5-7B-Instr SFT-CoT	59.6	19.1	46.2	53.8
Qwen-2.5-7B-Instr SFT-CoT-GRPO	59.9	**23.1 ↑ 4.2**	**54.0 ↑ 5.0**	**63.2↑ 1.6**
Qwen-2.5-14B-Instr Base	55.1	22.9	56.1	60.7
Qwen-2.5-14B-Instr SFT	**68.6 ↑ 13.5**	19.7	52.2	57.8
Qwen-2.5-14B-Instr SFT-CoT	58.6	23.5	52.8	60.6
Qwen-2.5-14B-Instr SFT-CoT-GRPO	59.2	**27.2 ↑ 4.3**	**59.2 ↑ 3.1**	**63.3 ↑ 2.6**

Table 1: Performance comparison of OpenAI o1, the 4-bit quantized version of the distilled Deepseek-R2 LLaMA 8B model, the Qwen-2.5-7B-Instruct model (Qwen-2.5-7B-Instr), and the Qwen-2.5-14B-Instruct model (Qwen-2.5-14B-Instr) evaluated across various datasets. This table compares the performance of untrained models (Base), those supervised fine-tuned on the Clinton Dataset (SFT), models fine-tuned with Chain-of-Thoughts (SFT-CoT) on the Clinton Dataset, and models that have undergone SFT-CoT on the Clinton Dataset and GRPO on the BIRD Dataset. Evaluation scores include execution accuracy (EXE), execution accuracy determined by an OpenAI o3-mini LLM judge (LLM-EXE), exact match scores (EM), and Rouge score (Rouge).

plex question answering tasks, further underlining the value of using SQL as a foundational tool for structured data reasoning.

4.6 Reward Ablation

In this section, we investigate the contribution of various reward functions in our GRPO training. Table 2 presents the results of our ablation study, evaluating the impact of different reward configurations on the model's performance on the BIRD, CRT-QA and Tablebench tasks. Specifically, we analyze the effect of different combinations of rewards—including execution-based, string matching, component-level matching, and LLM-based judgment rewards—on the accuracy of SQL execution and tabular question answering. Due to computational costs, we utilize the 7B and 8B models.

Ablation studies indicate that string matching serves as the most effective single reward due to its continuous nature, facilitating initial learning. However, exclusive reliance on string matching can lead to diminished performance in later training stages. We observe that combining string matching with additional reward mechanisms enhances overall effectiveness, as the initial continuous reward provides a substantial learning advantage. The most promising two-reward combination identified is string matching coupled with the LLM Judge Reward with classes. This synergistic approach effectively merges the continuous evaluation of string accuracy with the discrete assessment of general SQL quality, thereby creating a robust framework for improved model performance.

From the results in Table 2, we observe that incorporating a broader range of reward functions

Reward Configuration	BIRD	CRT-QA	Tablebench
Best Reward (LLaMA)	11.5	57.8	60.1
Best Reward (Qwen-2.5-7B-Instr)	19.6	53.9	62.7
Best 2 Rewards (LLaMA)	12.1	56.9	60.3
Best 2 Rewards (Qwen-2.5-7B-Instr)	20.0	53.2	64.5
Best 4 Rewards (LLaMA)	14.2	**58.1**	61.1
Best 4 Rewards (Qwen-2.5-7B-Instr)	**23.1**	54.0	**63.2**

Table 2: **Ablation study of reward configurations.** The models initially underwent SFT on Chain-of-Thought traces on Clinton, followed by GRPO on BIRD, where specific reward functions were applied. Performance is evaluated across the best GRPO reward configurations (best one, two, and four rewards) for each model. Evaluation scores include execution accuracy for BIRD, exact match for CRT-QA and Rouge for Tablebench.

generally improves model performance. For instance, the best four rewards configuration shows significant improvements on CRT-QA for the LLaMA model, indicating that a more diverse set of feedback signals enhances generalization across tasks. This suggests that combining different reward signals allows the model to better capture both syntactic correctness (in SQL) and logical coherence (in tabular reasoning), leading to a more balanced and accurate reasoning process.

5 Conclusion

In conclusion, our experiments demonstrate that integrating reinforcement learning with a supervised pretraining phase significantly enhances the model's ability to reason over tabular data. Notably, the distilled quantized LLaMA-8B model achieved a 34.2% relative performance improvement on the CRT-QA dataset, while the Qwen-2.5-7B model saw a 10.2% increase and the Qwen-2.5-14B model 5.5% relative increase, underscoring the efficacy of our two-stage framework in optimizing SQL

Question: Was there a difference in points earned in years when Pedro Rodríguez drove a Lotus compared to when he drove a Ferrari? Answer with only 'Yes' or 'No' that is most accurate and nothing else.

<think>
```sql SELECT pt FROM table WHERE entrant = "team lotus" - pt FROM table WHERE entrant = "scuderia ferrari" ``` I compared the points earned when Pedro Rodríguez drove a Lotus versus a Ferrari. Since the points differed (0 vs 1, 2, or 3), there was a difference.
</think>
Yes """

Figure 3: **SQL-Structured Reasoning in Tabular QA.** An LLM answering a natural language question over a table. While the output is a binary response ("Yes"), the model's internal reasoning implicitly follows an SQL-like logic: it compares subsets of rows filtered by different conditions to support its answer. This illustrates how models may invoke formal query structures even when the task does not explicitly require SQL, reflecting an internal alignment between table QA and SQL semantics.

First, let me understand what the tables are involved here.
...
Wait, so I need to:

Find the maximum occurrence in biwords.
Find the biword with that maximum occurrence.
Return w2nd from that biword.
In SQL:
SELECT w2nd FROM biwords WHERE occurrences = (SELECT MAX(occurrences) FROM biwords)
Yes, that query will return w2nd for the biword that has the highest occurrence.

But wait, if the maximum occurs multiple times, the query returns all such w2nd. So the query, without any LIMIT, would return all w2nd for biwords entries with the maximum occurrence.
But the question is about the word ID for the second word, so perhaps the requirement is to list all and then pick one?
So the final SQL query is:
SELECT w2nd FROM biwords WHERE occurrences = (SELECT MAX(occurrences) FROM biwords)

Let me test it. Suppose we have:
biwords:
lid | w1st | w2nd | occurrences
1 | 1 | 2 | 5
2 | 3 | 4 | 5
3 | 5 | 6 | 4
4 | 1 | 3 | 10
Then, the maximum occurrence is 10 in row 4. The query would select w2nd which is 3.
So the query returns the w2nd from the biword with the highest occurrence, which is 3.
So in SQLite, it's correct to just get w2nd from the row where occurrence is max.

Yes, that seems correct.

Figure 4: **Table-Guided CoT in LLMs for SQL Generation.** A reasoning trace from an LLM translating a natural language question into SQL. The model first interprets the task by examining the schema and example table rows, breaks the logic down into actionable steps, and validates the final SQL query through hypothetical execution. This illustrates how structured table understanding can guide accurate SQL synthesis.

execution and fostering transferable reasoning for complex question answering tasks. These findings suggest that SQL serves not only as a task-specific format but also as a foundational scaffold for developing robust tabular reasoning skills in LLMs.

Limitations

Our study focuses on medium-scale foundation models—distilled LLaMA 8B, Qwen-7B, Qwen-14B —whose exact pretraining corpora are undocumented. As a result, we cannot determine coverage or gaps across domains, languages, or proprietary material. This opacity complicates any analysis of domain blind spots, spurious correlations, or memorization risks. Moreover, the relatively modest parameter counts of these models may limit performance on tasks requiring deep domain expertise, such as biomedical or legal reasoning.

We evaluate tabular reasoning using CRT-QA and Tablebench, with o3-mini serving as an automated judge. While expedient, this setup lacks the nuance of human evaluation, particularly for complex reasoning and semantic alignment. Additionally, standard Text-to-SQL and tabular QA benchmarks may under-represent the complexity, ambiguity and noise present in real-world data, making our results more indicative of structured reasoning progress than deployment readiness.

Our current framework employs only two training stages. In contrast, multi-phase pipelines such as R1 leverage up to four stages, including instruction tuning and iterative CoT refinement. While our approach prioritizes simplicity and efficiency, it may sacrifice opportunities for deeper alignment or curriculum structuring.

Future research should address these limitations by exploring larger, better-documented models, human-in-the-loop evaluation, and more diverse datasets. Additional training stages—such as pre-CoT bootstrapping or domain-adaptive pretraining—may further enhance generalization and robustness in real-world table reasoning.

Acknowledgments

References

Nikhil Abhyankar, Vivek Gupta, Dan Roth, and Chandan K Reddy. 2024. H-star: Llm-driven hybrid sql-text adaptive reasoning on tables. *arXiv preprint arXiv:2407.05952*.

Shir Ashury-Tahan, Yifan Mai, Ariel Gera, Yotam Perlitz, Asaf Yehudai, Elron Bandel, Leshem Choshen, Eyal Shnarch, Percy Liang, Michal Shmueli-Scheuer, and 1 others. 2025. The mighty torr: A benchmark for table reasoning and robustness. *arXiv preprint arXiv:2502.19412*.

Marc Boubnovski, Kaspar Märtens, Lawrence Phillips, Daniel Keitley, Maria Dermit, and Julien Fauqueur. 2025. A scalable llm framework for therapeutic biomarker discovery: Grounding q/a generation in knowledge graphs and literature. In *ICLR 2025*
```

Workshop on Machine Learning for Genomics Explorations.

Ivan Brugere, Shubham Sharma, Sanjay Kariyappa, Anh Totti Nguyen, Freddy Lecue, and 1 others. 2024. Interpretable llm-based table question answering. *arXiv preprint arXiv:2412.12386*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wangxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. *arXiv preprint arXiv:2109.00122*.

Yilu Fang, Betina Idnay, Yingcheng Sun, Hao Liu, Zhehuan Chen, Karen Marder, Hua Xu, Rebecca Schnall, and Chunhua Weng. 2022. Combining human and machine intelligence for clinical trial eligibility querying. *Journal of the American Medical Informatics Association*, 29(7):1161–1171.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Moshe Hazoom, Vibhor Malik, and Ben Bogin. 2021. Text-to-sql in the wild: A naturally-occurring dataset based on stack exchange data. *arXiv preprint arXiv:2106.05006*.

Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. 2024. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint arXiv:2406.08426*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Pengcheng Jiang, Jiacheng Lin, Lang Cao, Runchu Tian, SeongKu Kang, Zifeng Wang, Jimeng Sun, and Jiawei Han. 2025. Deepretrieval: Hacking real search engines and retrievers with large language models via reinforcement learning. *arXiv preprint arXiv:2503.00223*.

Dongjun Lee, Choongwon Park, Jaehyuk Kim, and Heesoo Park. 2024. Mcs-sql: Leveraging multiple prompts and multiple-choice selection for text-to-sql generation. *arXiv preprint arXiv:2405.07467*.

Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, and 1 others. 2025a. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, and 1 others. 2025b. Omnisql: Synthesizing high-quality text-to-sql data at scale. *arXiv preprint arXiv:2503.02240*.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36:42330–42357.

Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2024. A survey of nl2sql with large language models: Where are we, and where are we going? *arXiv preprint arXiv:2408.05109*.

Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. 2025. Sql-r1: Training natural language to sql reasoning model by reinforcement learning. *arXiv preprint arXiv:2504.08600*.

Eduardo R Nascimento, Grettel García, Yenier T Izquierdo, Lucas Feijó, Gustavo MC Coelho, Aiko R de Oliveira, Melissa Lemos, Robinson LS Garcia, Luiz AP Paes Leme, and Marco A Casanova. 2025. Llm-based text-to-sql for real-world databases. *SN Computer Science*, 6(2):130.

Vaishali Pal, Andrew Yates, Evangelos Kanoulas, and Maarten de Rijke. 2023. Multitabqa: Generating tabular answers for multi-table question answering. *arXiv preprint arXiv:2305.12820*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eicu collaborative research database, a freely available multi-center database for critical care research. *Scientific data*, 5(1):1–13.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36:36339–36348.

Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, Sercan Arik, and 1 others. 2025. Reasoning-sql: Reinforcement learning with sql tailored partial rewards for reasoning-enhanced text-to-sql. *arXiv preprint arXiv:2503.23157*.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and 1 others. 2024. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to sql queries. *arXiv preprint arXiv:2010.11246*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Josefa Lia Stoisser, Marc Boubnovski Martell, Kaspar Märtens, Lawrence Phillips, Stephen Michael Town, Rory Donovan-Maiye, and Julien Fauqueur. 2025. Query, don't train: Privacy-preserving tabular prediction from ehr data via sql queries. *arXiv preprint arXiv:2505.21801*.

Yuan Sui, Jiaru Zou, Mengyu Zhou, Xinyi He, Lun Du, Shi Han, and Dongmei Zhang. 2023. Tap4llm: Table provider on sampling, augmenting, and packing semi-structured data for large language model reasoning. *arXiv preprint arXiv:2312.09039*.

Ruoxi Sun, Sercan Ö Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, and 1 others. 2023. Sql-palm: Improved large language model adaptation for text-to-sql (extended). *arXiv preprint arXiv:2306.00739*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Ping Wang, Tian Shi, and Chandan K Reddy. 2020. Text-to-sql generation for question answering on electronic medical records. In *Proceedings of The Web Conference 2020*, pages 350–361.

Shuo Wang and Carlos Crespo-Quinones. 2023. Natural language models for data visualization utilizing nvbench dataset. *arXiv preprint arXiv:2310.00832*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xeron Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, and 1 others. 2025. Tablebench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25497–25506.

Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. 2024. Deepseek-prover: Advancing theorem proving in llms through large-scale synthetic data. *arXiv preprint arXiv:2405.14333*.

Peiying Yu, Guoxin Chen, and Jingjing Wang. 2025. Table-critic: A multi-agent framework for collaborative criticism and refinement in table reasoning. *arXiv preprint arXiv:2502.11799*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Zheng Yuan, Hao Chen, Zijin Hong, Qinggang Zhang, Feiran Huang, and Xiao Huang. 2025. Knapsack optimization-based schema linking for llm-based text-to-sql generation. *arXiv preprint arXiv:2502.12911*.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, and 1 others. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *arXiv preprint arXiv:2307.08674*.

Zhehao Zhang, Xitao Li, Yan Gao, and Jian-Guang Lou. 2023. Crt-qa: A dataset of complex reasoning question answering over tabular data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2131–2153.

Xueliang Zhao, Wei Wu, Jian Guan, and Lingpeng Kong. 2025. Promptcot: Synthesizing olympiad-level problems for mathematical reasoning in large language models. *arXiv preprint arXiv:2503.02324*.

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

A Summary of Clinton Dataset

We conduct part of our evaluation using the `Clinton/Text-to-sql-v1` dataset,[4] a large-scale compilation of natural language to SQL examples spanning a broad set of domains. This benchmark includes 26 individual datasets, covering academic records, medical databases, entertainment metadata, government statistics, and more.

Each example in the dataset consists of a natural language query, an associated database schema, and a corresponding SQL statement. Some subsets also include table content or ground-truth execution results. The diversity in schema complexity and domain coverage makes this benchmark well-suited for evaluating both generalization and transfer in Text-to-SQL and tabular reasoning models.

Key datasets include:

- **Spider** (Yu et al., 2018) – Complex, cross-domain Text-to-SQL benchmark.
- **WikiSQL** (Zhong et al., 2017) – Large-scale dataset with simple queries over Wikipedia tables.
- **ATIS** (Hemphill et al., 1990) – Airline travel information with traditional semantic parsing annotations.
- **MIMICSQL** (Wang et al., 2020) and **eICU** (Pollard et al., 2018) – Clinical databases for medical question answering.

We also include lesser-known and synthetic datasets such as `Criteria2SQL` (Fang et al., 2022), `SEDE` (Hazoom et al., 2021), `SQuALL` (Shi et al., 2020), and `NVBench` (Wang and Crespo-Quinones, 2023), along with public domain tabular corpora like IMDb, Yelp, and historical sports or wildfire datasets.

This variety allows us to test the ability of LLMs to reason across database schemas, interact with realistic tabular structures, and generalize beyond fixed SQL templates.

[4] https://huggingface.co/datasets/Clinton/Text-to-sql-v1

B Prompts

B.1 Creating Synthetic CoT

This section outlines the structure of prompts designed for SQL query generation tasks. Each prompt features SQL table schemas and clear instructions, facilitating the generation of valid SQL queries using SQLite syntax. The expert guidance within the prompts emphasizes the requirement to articulate the reasoning behind the constructed SQL queries. By utilizing this approach, we aim to train models that can effectively understand the context of relational data and generate precise queries that meet specific operational goals, thereby enhancing the overall interpretability and accuracy of automated SQL generation.

You are a SQL expert. Below are SQL table schemas paired with instructions that describe a specific task. Using valid SQLite syntax, write a response that appropriately completes the request for the provided tables.
SCHEMA: schema
INSTRUCTIONS: specific task instructions
When answering, provide reasoning for the SQL query you create using the following template:
<sql> Write the SQL query here, ensuring it adheres to SQLite syntax and effectively accomplishes the task described in the instructions. </sql>

B.2 Evaluation of Synthetic CoT

This section specifies a prompt for evaluating the correctness of SQL queries based on a defined schema and a reference SQL query. The prompt clearly delineates the evaluation task for the SQL expert, presenting the query to be evaluated, the relevant schema, and the correct SQL reference. The evaluator is instructed to determine whether the provided SQL query is correct or incorrect, with responses limited to "Correct" or "Wrong." This structured approach facilitates precise assessment of SQL queries, contributing to the development of robust models capable of generating and validating SQL syntax effectively.

You are an SQL expert, and your task is to evaluate whether the SQL query below is correct based on the provided schema and the correct SQL reference.
SQL Query: ans.sql
Schema: schema
Correct SQL: correct_sql
Return ONLY "Correct" or "Wrong".

B.3 LLM Judge for Execution Based Reward

For our Execution Reward in Group Relative Policy Optimization (GRPO) the LLM judge is instructed to count the number of orthographic changes required to convert each predicted query into the corresponding correct query. The reward is computed using the following equation:

$$R_{\text{exec}} = \frac{1}{x+1}, \tag{3}$$

where x is the number of detected changes. This methodology provides a more continuous measure of execution accuracy, crucial for refining the model's performance.

You are an SQL expert. Count how many changes you need to make to get the following predicted queries correct.
Predicted Queries (one per line): queries_to_rank
For reference, use this Schema: schema.
Here is the correct query: true_query
You should count the number of Orthographic elements you need to change from the predicted queries to the correct query.
ONLY RETURN a JSON object with a single 'scores' field containing a list of **num_queries** numbers reflecting the number of changes needed for each predicted query.

B.4 LLM Judge with Classes

The LLM judge reward is designed to evaluate the quality of predicted SQL queries by comparing them to a reference correct query. In this task, the judge is instructed to assign a grade to each predicted query on a scale from 'Very bad' to 'Excellent.' The grading criteria are explicitly defined, allowing the judge to assess various aspects of the queries, including grammatical correctness, logical accuracy, and overall fidelity to the correct query. This structured grading system enables a nuanced analysis of the model's output quality, providing insights that facilitate targeted improvements in query generation.

Compare these SQL queries to the correct query and grade each one as: 'Very bad', 'Bad', 'Above average', 'Good', or 'Excellent'. Use the following grading system, and the correct query as reference :
Correct Query: true_query
1. Excellent: this is only given when the SQL query is perfect and matches {true_query}
2. Good: This is when there is a grammar mistake in the query
3. Above average: This is when the query is mostly correct but gets a logical step wrong in the query
4. Bad: Makes more than one mistake in the query
5. Very bad: does not produce a query or varies significantly from the correct query
Queries to grade: queries_to_rank
{format_instructions}

C Implementation Details

In our experiments, we utilize VERL[5] for training the 14B models. To enhance efficiency, Unsloth[6] is employed for the 7B and 8B models. Unsloth provides support for QLoRA-style training with Flash Attention 2, bitsandbytes quantization, and PEFT-compatible adapters.

We fine-tuned three pretrained models:

- **Qwen-2.5-7B**, a dense, instruction-tuned model released by Alibaba DAMO, trained in full precision [7].
- **Qwen-2.5-14B**, a larger, dense, instruction-tuned model released by Alibaba DAMO, trained in full precision [8].
- **DeepSeek-R1-Distill LLaMA3-8B**, a 4-bit quantized variant of Meta's LLaMA 3–8B, distilled by DeepSeek AI[9].

Supervised fine-tuning (SFT) was performed on the Clinton dataset using QLoRA adapters, while reinforcement learning with GRPO was applied on the BIRD benchmark. The GRPO setup used candidate comparisons and execution-guided rewards computed via SQLite.

Experiments were conducted on 4×A100 80GB GPUs using mixed-precision (FP16).

[5] https://GitHub.com/volcengine/verl
[6] https://GitHub.com/unslothai/unsloth
[7] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[8] https://huggingface.co/Qwen/Qwen2.5-14B-Instruct
[9] https://huggingface.co/unsloth/DeepSeek-R1-Distill-Llama-8B-unsloth-bnb-4bit