

SQL-ASTRA: Alleviating Sparse Feedback in Agentic SQL via Column-Set Matching and Trajectory Aggregation

Anonymous ACL submission

Abstract

Agentic Reinforcement Learning (RL) shows promise for complex tasks, but Text-to-SQL remains mostly restricted to single-turn paradigms. A primary bottleneck is the credit assignment problem. In traditional paradigms, rewards are determined solely by the final-turn feedback, which ignores the intermediate process and leads to ambiguous credit evaluation. To address this, we propose Agentic SQL, a framework featuring a universal two-tiered reward mechanism designed to provide effective trajectory-level evaluation and dense step-level signals. First, we introduce Aggregated Trajectory Reward (ATR) to resolve multi-turn credit assignment. Using an asymmetric transition matrix, ATR aggregates process-oriented scores to incentivize continuous improvement. Leveraging Lyapunov stability theory, we prove ATR acts as an energy dissipation operator, guaranteeing a cycle-free policy and monotonic convergence. Second, Column-Set Matching Reward (CSMR) provides immediate step-level rewards to mitigate sparsity. By executing queries at each turn, CSMR converts binary (0/1) feedback into dense $[0, 1]$ signals based on partial correctness. Evaluations on BIRD show a 5% gain over binary-reward GRPO. Notably, our approach outperforms SOTA Arctic-Text2SQL-R1-7B on BIRD and Spider 2.0 using identical models, propelling Text-to-SQL toward a robust multi-turn agent paradigm.

1 Introduction

In recent years, Agentic Reinforcement Learning (RL) has garnered significant attention (Zhang et al., 2025a), empowering Large Language Models (LLMs) to engage in multi-turn interactions with environments to accomplish more complex tasks, such as deep research (OpenAI, 2025; Xu and Peng, 2025; Huang et al., 2025), Web searching (Jin et al., 2025), and code execution (Jiang et al., 2025; Feng et al., 2025). In most scenarios, the capabilities of these methods are enhanced

through Reinforcement Learning with Verifiable Rewards (RLVR) (Zhang et al., 2025b), where feedback is provided based on the correctness of the final outcome (Guo et al., 2025a; Li et al., 2025b).

However, when applied to complex tasks requiring exploratory reasoning, Agentic RL faces three core challenges: **(1) Paradigm Constraint:** While agents are designed for multi-turn interaction, most existing work in specific domains like **Text-to-SQL** remains confined to a single-turn, static generation paradigm (Ma et al., 2025; Yao et al., 2025; Ali et al., 2025). This framework fails to reflect the dynamic process of human data analysts who use multiple tentative queries to gather context and refine their strategy (Kim et al., 2024; Fürst et al., 2024), which severely limits the model’s ability to solve complex, real-world problems (Huo et al., 2025; Lei et al., 2025a). **(2) Credit Assignment:** In multi-turn trajectories, evaluation signals typically rely exclusively on **final-turn feedback** (Guo et al., 2025c; Feng et al., 2025). This “all-or-nothing” approach treats the interaction sequence as a black box, introducing a pervasive credit assignment problem where the agent cannot distinguish which intermediate steps contributed to the final outcome. **(3) Micro-level Reward Sparsity:** Even when step-level feedback is available, it is often restricted to coarse, **binary (0/1) signals** based on execution success (Ma et al., 2025; Lei et al., 2025b; Yao et al., 2025). Such sparse feedback ignores the rich information in “partially correct” queries, providing insufficient granular guidance and drastically restricting the efficiency and robustness of RL training.

To systematically address these challenges, we propose an innovative methodology:

(1) We construct a multi-turn interactive framework that enables agents to iteratively gather context and refine SQL queries through dynamic database interaction, effectively overcoming the single-turn limitations.

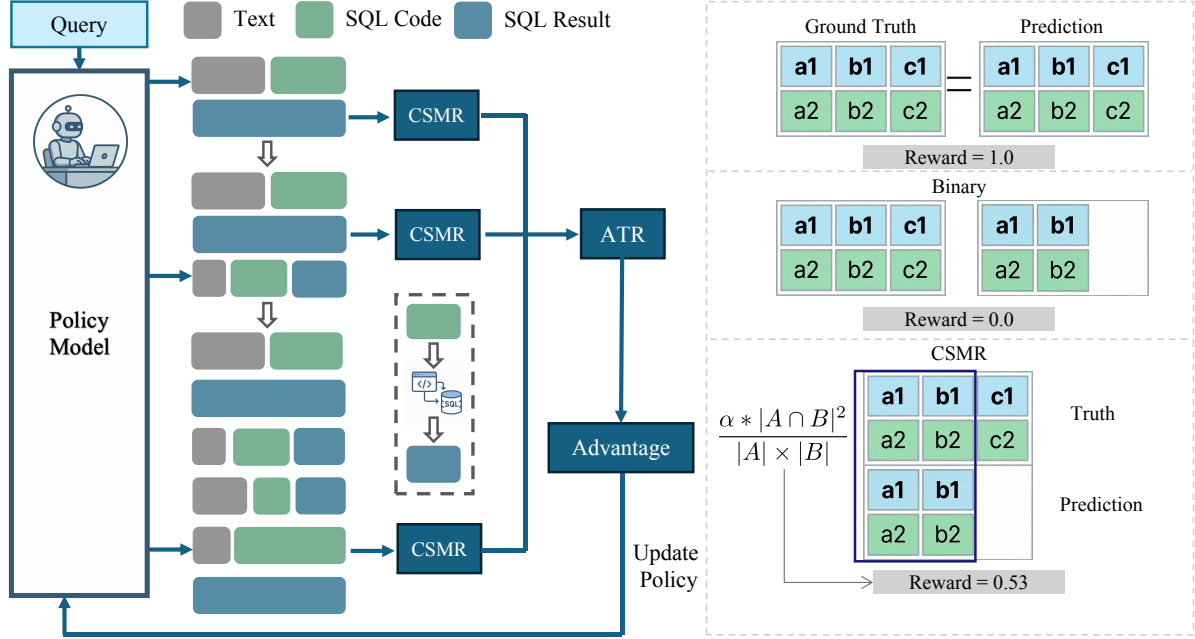


Figure 1: The framework of Agentic SQL. CSMR captures rich signals from error cases, whereas traditional binary rewards overlook this information.

(2) To address the reliance on final-turn feedback, we design the **Aggregated Trajectory Reward (ATR)**. Unlike traditional methods, ATR utilizes an **Asymmetric Transition Matrix** to aggregate signals across the entire reasoning path, explicitly incentivizing strategies that exhibit continuous and monotonic improvement. Crucially, we provide a **rigorous Theoretical Guarantee**: by modeling the reasoning process as a dynamical system, we prove that ATR acts as an energy dissipation operator, mathematically ensuring a cycle-free policy and monotonic convergence based on Lyapunov stability theory.

(3) We propose the **Column-Set Matching Reward (CSMR)** as an immediate, **dense step-level reward**. By evaluating *partial correctness* via column value-set normalization, CSMR transforms 0/1 outcomes into granular signals within the $[0, 1]$ range. This provides precise, step-by-step guidance and serves as the foundational input for ATR trajectory aggregation.

2 Methodology and Convergence Analysis

2.1 Agentic SQL Framework

Traditional Reinforcement Learning (RL) approaches for Large Language Models (LLMs) are mostly confined to a single-turn interaction setting, aiming to maximize the expected reward

$J_{\text{single-turn}}(\theta)$ for a single response.

$$J_{\text{single-turn}}(\theta) = \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_{\theta}(\cdot|s)} [R(s, a)].$$

However, this paradigm is insufficient for complex, exploratory tasks like Text-to-SQL, which require the agent to engage in multi-step interaction, strategy iteration, and error recovery.

To accurately model this dynamic, we formalize the interactive Text-to-SQL task as a Finite-Horizon Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. An agent learns a policy π_{θ} by generating interaction trajectories $\tau = (s_0, a_0, \dots, s_K)$ to maximize the expected cumulative reward based on our process-based reward function \mathcal{R} :

$$J_{\text{multi-turn}}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{K-1} \mathcal{R}(s_t, a_t) \right] \quad (1)$$

The overall framework is illustrated in Figure 1. In each iteration, after generating the SQL code, the agent interacts with the database to retrieve the execution result and subsequently employs CSMR to derive a dense reward. Upon completion of the multi-turn generation, the ATR mechanism aggregates the step-wise CSMRs to formulate the advantage for the entire trajectory.

2.2 Reward Design

2.2.1 Column-Set Matching Reward

As shown on the right side of Figure 1, in Text-to-SQL tasks, traditional reward mechanisms typically employ a sparse binary 0/1 signal. This mechanism requires the execution result of a predicted query, P , to *perfectly match* the execution result of the gold answer, G . Any slight discrepancy (e.g., row ordering, partially missing data) results in a reward of 0. This sparse reward signal significantly hinders RL convergence, as it ignores the substantial “partially correct” information that predicted results may contain.

Algorithm 1 CSMR (Column-Set Matching Reward) with Perfect Match Check

Require: Gold result table G , Predicted result table P , Scaling factor α (e.g., 0.8)

Ensure: Reward score R_{CSMR}

```
1:  $\triangleright$  Step 0: Check for Perfect Match
2: if ISPERFECTMATCH( $G, P$ ) then  $\triangleright$  e.g.,  $\text{set}(G) == \text{set}(P)$ 
3:   return 1.0
4: end if

5:  $\triangleright$  Step 1: Column Value-Set Extraction (for partial score)
6:  $G' \leftarrow \text{DEDUPLICATEROWS}(G)$ 
7:  $P' \leftarrow \text{DEDUPLICATEROWS}(P)$ 
8:  $N_c^G \leftarrow \text{GETNUMCOLUMNS}(G')$ 
9:  $N_c^P \leftarrow \text{GETNUMCOLUMNS}(P')$ 
10:  $D_G \leftarrow \emptyset$   $\triangleright$  Initialize set of gold column-sets
11:  $D_P \leftarrow \emptyset$   $\triangleright$  Initialize set of predicted column-sets
12: for  $i \leftarrow 1$  to  $N_c^G$  do
13:    $C_i^G \leftarrow \text{GETUNIQUEVALUESINCOLUMN}(G', i)$ 
14:    $D_G \leftarrow D_G \cup \{C_i^G\}$ 
15: end for
16: for  $j \leftarrow 1$  to  $N_c^P$  do
17:    $C_j^P \leftarrow \text{GETUNIQUEVALUESINCOLUMN}(P', j)$ 
18:    $D_P \leftarrow D_P \cup \{C_j^P\}$ 
19: end for

20:  $\triangleright$  Step 2: Column-Set Matching
21:  $M \leftarrow 0$   $\triangleright$  Initialize match count
22: for each  $C^G \in D_G$  do
23:   if exists  $C^P \in D_P$  s.t.  $C^G = C^P$  then
24:      $M \leftarrow M + 1$ 
25:   end if
26: end for

27:  $\triangleright$  Step 3: Compute Score
28:  $N_{\text{prod}} \leftarrow N_c^G \times N_c^P$ 
29: if  $N_{\text{prod}} = 0$  then
30:    $R_{\text{CSMR}} \leftarrow 0.0$ 
31: else
32:    $R_{\text{CSMR}} \leftarrow \frac{M^2}{N_{\text{prod}}} \times \alpha$ 
33: end if

34: return  $R_{\text{CSMR}}$ 
```

To address this, we propose Column-Set Matching Reward (CSMR), a novel, denser reward func-

tion designed to measure the meaningful *structural overlap* between P and G . Our core idea is to move away from comparing *rows* (tuples) and instead compare the *sets of values* within each *column*. Specifically, our reward calculation process is detailed in Algorithm 1.

The CSMR mechanism, using product-based normalization, effectively captures *partial correctness*. It provides a dense reward signal for matching column value-sets, even with incorrect row composition, thereby alleviating the sparsity problem of binary 0/1 rewards.

We note that CSMR is insensitive to row-level combinations. For example, $P = [[a1, b2], [a2, b1]]$ would perfectly match the column-sets of $G = [[a1, b1], [a2, b2]]$. To address this, we introduce a scaling factor $\alpha < 1.0$ (e.g., 0.8). This factor caps the maximum reward from CSMR at α , intentionally distinguishing these ‘pseudo-perfect’ (column-only) matches from truly perfect row-level matches.

2.3 Aggregated Trajectory Reward and Theoretical Guarantee

While CSMR provides spatial density, it is insufficient for evaluating the temporal quality of a complete multi-step trajectory τ . In a traditional RL setup, one might employ a *Step-wise Update* strategy, which assigns and backpropagates reward signals r_t immediately after each interaction turn. However, this naive approach suffers from high gradient variance due to **inconsistent signal scaling** across heterogeneous database schemas. Furthermore, it is susceptible to **reward hacking**—a phenomenon where agents learn to exploit the reward function through intentional oscillations.

To overcome these limitations, we design the **Aggregated Trajectory Reward (ATR)** R_{ATR} . Unlike step-wise reinforcement, ATR is a single scalar reward provided only at episode termination to explicitly incentivize **continuous improvement** while suppressing oscillatory behaviors. We provide a detailed ablation study comparing ATR with the *Step-wise Update* in Section 3.3.1.

ATR Calculation The ATR aggregates the trajectory’s utility by weighing both the direction and magnitude of semantic changes. The total reward is computed as:

The step-wise transition function Ψ is derived from an asymmetric matrix \mathcal{M} scaled by the gradient magnitude $|\Delta R_t|$, where $\Delta R_t = R_t - R_{t-1}$.

The core inductive bias is encoded in \mathcal{M} , where each element represents a specific semantic behavior:

$$\begin{aligned} \mathcal{M} &= \begin{pmatrix} R_{\text{Low} \rightarrow \text{Low}} & R_{\text{Low} \rightarrow \text{High}} \\ R_{\text{High} \rightarrow \text{Low}} & R_{\text{High} \rightarrow \text{High}} \end{pmatrix} \\ &= \begin{pmatrix} -0.0 & +1.0 \\ -1.5 & 0.0 \end{pmatrix} \end{aligned}$$

$$\Psi(R_{t-1}, R_t) = \mathcal{M}_{s_{t-1}, s_t} \cdot \begin{cases} |\Delta R_t| & \text{if } s_{t-1} \neq s_t \\ 1.0 & \text{if } s_{t-1} = s_t \end{cases} \quad (2)$$

The binary states (s_{t-1}, s_t) are determined dynamically by the gradient and a stagnation threshold τ :

- (i) **Transition** ($s_{t-1} \neq s_t$): A positive gradient $\Delta R_t > 0$ triggers an improvement transition, while $\Delta R_t < 0$ triggers degradation.
- (ii) **Stagnation** ($s_{t-1} = s_t$): When $\Delta R_t \approx 0$, the state is sustained as High if $R_t > \tau$, otherwise it remains Low.

This design ensures that the agent is rewarded proportionally to the *extent* of improvement ($|\Delta R_t|$), while the asymmetric \mathcal{M} (with $|R_{\text{High} \rightarrow \text{Low}}| > |R_{\text{Low} \rightarrow \text{High}}|$) ensures strict dissipativity.

Theoretical Foundation: Lyapunov-Guided Convergence We demonstrate that the ATR mechanism is not merely heuristic, but acts as a necessary energy dissipation operator, guaranteeing asymptotic stability (i.e., convergence to the correct SQL query) from the perspective of Lyapunov stability theory (Taylor et al., 2018).

1. Semantic Error Energy (Lyapunov Candidate Function): We transform the CSMR score $\Phi(s_t) = R_{\text{CSMR}}(s_t)$ into the **Semantic Error Energy** $V(s_t)$:

$$V(s_t) = 1 - \Phi(s_t) \quad (3)$$

This function satisfies the fundamental Lyapunov properties, where $V(s) = 0$ if and only if the system reaches the equilibrium point s^* (the correct SQL).

2. Stability Criterion and ATR’s Dissipative Role: Asymptotic stability requires the energy sequence to be strictly decreasing, $\Delta V(s_t) = V(s_{t+1}) - V(s_t) < 0$. Our R_{step} is explicitly designed to approximate the negation of the orbital derivative:

$$R_{\text{step}}(s_t, s_{t+1}) \propto -\nabla V(s_t) \approx \Phi(s_{t+1}) - \Phi(s_t) \quad (4)$$

Maximizing the cumulative ATR implicitly optimizes the policy π_θ to maximize the rate of energy dissipation, accelerating convergence towards s^* .

3. Necessity of Asymmetric Shaping: Eliminating Limit Cycles

Multi-turn agents often suffer from *limit cycles*, oscillating between suboptimal states. We prove an **Asymmetric Transition Matrix** \mathcal{M} is necessary to eliminate such cycles. While symmetric rewards satisfy $\sum_{t \in \mathcal{C}} R_t = 0$ for any cycle \mathcal{C} —yielding a merely *Lagrange stable* (Leine, 2010; Zhang and Zeng, 2018) and oscillatory system—ATR imposes $|R_{\text{High} \rightarrow \text{Low}}| > |R_{\text{Low} \rightarrow \text{High}}|$, ensuring a net reward loss ($\sum_{t \in \mathcal{C}} R_t < 0$). This transforms the reasoning into a **strictly dissipative system**, forcing π^* to be cycle-free along a monotonic energy-descent path (Appendix B).

4. Robustness via Quantized Feedback: The threshold τ in the ATR calculation acts as a **quantized feedback filter**, akin to techniques in Sliding Mode Control.

$$S_t = \mathbb{I}(\Phi(s_t) > \tau)$$

Given the stochastic nature of LLMs, the continuous potential $\Phi(s_t)$ may contain high-frequency noise. By discretizing the state transitions, we ensure that the Lyapunov derivative ΔV drives policy updates only when a **significant semantic phase transition** occurs, thereby enhancing the robustness of the learning process.

2.4 GRPO Training with Tool Masking

We use the GRPO algorithm to optimize our policy model (Guo et al., 2025a). For a specific question-answer pair (q, a) , GRPO’s underlying behavior policy, $\pi_{\theta_{\text{old}}}$, generates a group of G individual responses, denoted as $\{o_i\}_{i=1}^G$. The advantage for the i -th response within this ensemble is then calculated by normalizing the rewards specific to that group, $\{R_{\text{ATR}, i}\}_{i=1}^G$:

$$\hat{A}_{i,t} = \frac{R_{\text{ATR}, i} - \text{mean}(\{R_{\text{ATR}, j}\}_{j=1}^G)}{\text{std}(\{R_{\text{ATR}, j}\}_{j=1}^G)}. \quad (5)$$

Thus, every token in the trajectory uses this normalized reward as its advantage. Additionally, we introduce a binary mask,

$$M_{i,t} = \begin{cases} 1 & \text{if } o_{i,t} \text{ is reasoning token} \\ 0 & \text{if } o_{i,t} \text{ is execution token} \end{cases} \quad (6)$$

This loss masking ensures the model focuses on learning the reasoning process. We define

the clipped surrogate objective as $\rho_{i,t}^{\text{clip}}(\theta) = \text{clip}(\rho_{i,t}(\theta), 1 - \varepsilon, 1 + \varepsilon)$. The optimized GRPO loss is:

$$\mathcal{L}_{\text{group}}(\theta) = \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} M_{i,t} \min \left(\rho_{i,t}(\theta) \hat{A}_{i,t}, \rho_{i,t}^{\text{clip}}(\theta) \hat{A}_{i,t} \right)$$

$$\mathcal{L}(\theta) = - \mathbb{E}_{\substack{(q,a) \sim \mathcal{D} \\ o_i \sim \pi_{\theta_{\text{old}}}(\cdot|q)}} \left[\mathcal{L}_{\text{group}}(\theta) \right] \quad (7)$$

where G is the number of sampled trajectories per group, and $\rho_{i,t}(\theta) = \frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q, o_{i,<t})}$ is the per-token importance ratio.

3 Experiment

3.1 Setting

Our experiments consist of two distinct sets of trials. The first set utilizes **Qwen2.5-7B-Instruct** as the base model, which undergoes direct RL training without a cold-start phase to demonstrate the viability of our approach. This configuration is compared with other methods employing Qwen2.5-Coder¹. The second set involves the **OmniSQL** model, which requires a Format-6k fine-tuning step to acquire the tool-calling format; this is then compared with other methods based on the same model. All experiments are conducted on 32 NVIDIA A800-80G GPUs.

In terms of evaluation, we primarily utilize BIRD-Dev (Li et al., 2024a) and Spider (Yu et al., 2018) to assess general SQL capabilities. Furthermore, we evaluate the model’s Agent capabilities using the challenging enterprise-grade Spider-2.0 dataset (Lei et al., 2024). Please refer to Appendix C for comprehensive training details.

3.2 Results

Baseline Our primary baseline is the performance of the same model in a *single-turn SQL* setting. We also include strong proprietary models like GPT-4o (Achiam et al., 2023) and DeepSeek-V3 (Guo et al., 2024), along with several other powerful models based on RL algorithms for SQL tasks, such as strong performers in the single-turn setting: **OmniSQL**, **SQL-R1**, **Reasoning-SQL** and **Arctic-R1**, as well as the multi-turn Agent framework **MTIR-SQL**.

¹Qwen2.5-Coder is overly specialized due to its training regime, leading to insufficient exploratory capabilities that render it unsuitable for this experiment.

Main Results In the Qwen2.5-7B-Instruct experiments, our framework demonstrates excellent performance without requiring a cold start. By framing Actions as parsed signals and embedding a Rethink mechanism into \mathcal{P} , the agent learns to generate and utilize SQL results within one RL loop. Consequently, π_{θ} acquires complex reasoning chains by maximizing $J_{\text{multi-turn}}(\theta)$, enabling RL training to start from scratch. Our experiments also reveal that this approach is highly dependent on the chosen base model retaining a certain level of exploration capability; for instance, it could not be directly initiated on Qwen2.5-Coder, which provides significant inspiration for the future training of agents.

Our main results are presented in Table 1. From the experiments, we can observe that the effectiveness of Agentic SQL is highly significant. It outperforms the *single-turn binary (0/1) GRPO* method by **5.7%** on the BIRD dataset and **3.7%** on the Spider dataset, respectively. In comparison to other methods, although the Qwen2.5-7B-Instruct base model we used has weaker foundational capabilities than Qwen2.5-Coder-7B, after being trained with Agentic SQL, it achieves superior performance. Compared to Reasoning-SQL-7B, which uses the same training dataset, our method completely surpasses it in both in-domain and OOD evaluations. To demonstrate the effectiveness of our method on a more powerful model, we utilized the **OmniSQL-7B** model for agent reinforcement learning generation. It surpassed the single-turn 0/1 reward GRPO on both the BIRD and Spider datasets. Furthermore, it also outperformed SQL-R1 and Arctic-Text2SQL-R1-7B, which are based on OmniSQL-7B, achieving improvements of 2.5% and 1.5% respectively on the BIRD dataset.

Result in Spider2.0 Evaluation on Spider 2.0 typically requires the introduction of more complex workflows for the model to generate the final answer in multiple steps. We demonstrated an even higher level of excellence on the larger and more practical Spider 2.0 dataset. While a series of models using only 0/1 rewards generally converge around 15%, our model achieves 17.7%. This is because, by integrating the CSMR + ATR mechanism, the model can receive more dense signals and learn to move towards better directions in each call, progressively solving the problem.

Model	RL Data	Base Model	Bird (Greedy)	Spider-Dev (Greedy)	Spider2.0-Lite (Mj@8)
GPT-4o	—	—	64.4	80.4	15.6
Deepseek-V3	—	—	62.5	78.3	15.6
Qwen2.5-7B-Instruct (Team, 2024)	—	-	47.5	74.7	5.1
Qwen2.5-Coder-7B (Team, 2024)	—	-	58.2	77.4	2.2
SQL-R1 + Qwen2.5-Coder-7B (Ma et al., 2025)	SynSQL-Complex-5K	Qwen2.5-Coder-7B	63.1	84.5	-
Reasoning-SQL-7B (Ma et al., 2025)	BIRD	Qwen2.5-Coder-7B	64.0	78.7	-
OmniSQL-7B (Li et al., 2025a)	—	Qwen2.5-Coder-7B	64.1	85.0	12.6
SQL-R1 + OmniSQL-7B (Ma et al., 2025)	SynSQL-Complex-5K	OmniSQL-7B	66.6	87.6	14.8
Arctic-Text2SQL-R1-7B (Yao et al., 2025)	BIRD + SPIDER + Gretel*	OmniSQL-7B	67.6	-	15.6
MTIR-SQL-4B (Xu et al., 2025)	BIRD + SPIDER	Qwen3-4B	63.1	82.4	-

Our Results

<i>Binary Reward</i>					
Single-Turn GRPO	BIRD	Qwen2.5-7B-Instruct	58.5	79.2	6.6
Agentic SQL	BIRD	Qwen2.5-7B-Instruct	61.1	80.7	5.9
Agentic SQL + ATR	BIRD	Qwen2.5-7B-Instruct	63.6	81.4	7.4
<i>CSMR Reward</i>					
Single-Turn GRPO + CSMR	BIRD	Qwen2.5-7B-Instruct	59.4	79.3	6.6
Agentic SQL + CSMR	BIRD	Qwen2.5-7B-Instruct	62.5	81.1	8.1
Agentic SQL + CSMR + ATR	BIRD	Qwen2.5-7B-Instruct	64.2	82.9	8.8
<i>Ablation Study on ATR Design</i>					
ATR	BIRD	Qwen2.5-7B-Instruct	64.2	82.9	8.8
w/ Symmetric Matrix	BIRD	Qwen2.5-7B-Instruct	60.1	80.2	7.4
w/ Step-wise Update	BIRD	Qwen2.5-7B-Instruct	61.3	82.0	8.1
<i>Cold-Start Model</i>					
Single-Turn GRPO	BIRD + Spider	OmniSQL-7B	67.4	87.4	14.8
Agentic SQL + CSMR + ATR	BIRD + Spider	OmniSQL-7B	69.1	88.8	17.7

Table 1: For Bird and Spider, we consistently use Greedy decoding, while for Spider 2.0, we use 8-time sampling followed by majority voting on the answers. *Gretel indicates Gretel-Synth-Filtered.

3.3 Ablation Study

Our proposed approach primarily encompasses three core components: the Agentic SQL Framework, CSMR, and ATR. We conducted a thorough ablation study on these three modules sequentially. Table 1 demonstrates that CSMR consistently outperforms the conventional binary reward across all experimental settings, thoroughly validating the effectiveness of its fine-grained reward mechanism. Furthermore, the Agentic SQL Framework proves to be the most significant contributor to performance enhancement, yielding a performance gain of nearly 3% on the Bird dataset. Finally, the ATR Module effectively enhances the signal density within multi-turn conversations, ensuring that sufficient and effective feedback signals are obtained for each generation step.

3.3.1 Ablation Study on ATR Design

We present the validation results regarding the necessity of each ATR component in Table 1.

Necessity of Asymmetric Shaping We compared our asymmetric design with a symmetric setting ($|R_{\text{High} \rightarrow \text{Low}}| = |R_{\text{Low} \rightarrow \text{High}}|$). We observed that the symmetric setting causes the model to engage in repetitive generation, resulting in numerous unnecessary loops and degrading sample learning effi-

ciency. This confirms that asymmetric penalties are crucial for eliminating limit cycles and enforcing the strict dissipativity condition.

Advantage of Trajectory Aggregation In this variant, we assigned the step-wise reward directly as the advantage to the corresponding tokens and removed the group normalization operation. The performance drop observed highlights the efficacy of our aggregated approach in mitigating the credit assignment problem. Under this setting, the prefix context varies at each turn, making it impossible to perform group normalization based on the same original question, thereby compromising the effectiveness of the advantage estimation.

3.3.2 Ablation Study on the Scaling Factor α in CSMR

Sensitivity analysis (Table 4) shows negligible performance variance, confirming CSMR’s robustness to α . This factor distinguishes “perfect” matches from “pseudo-perfect” ones—where column values match but row compositions differ (e.g., $P = [[a1, b2], [a2, b1]]$ vs. $G = [[a1, b1], [a2, b2]]$). By penalizing these subtle mismatches, α prevents false-positive reinforcement and ensures precise semantic alignment.

Input Distribution	ATR Reward
[1]	1.00
[0, 1]	0.90
[0, 0, 1]	0.80
[1, 1]	0.70
[0, 1, 1]	0.60
[1, 1, 1]	0.40
[1, 0, 1]	0.30
[0]	0.00
[0, 0]	-0.10
[0, 0, 0]	-0.20
[1, 0]	-0.60
[0, 1, 0]	-0.70
[1, 0, 0]	-0.70
[1, 1, 0]	-0.90

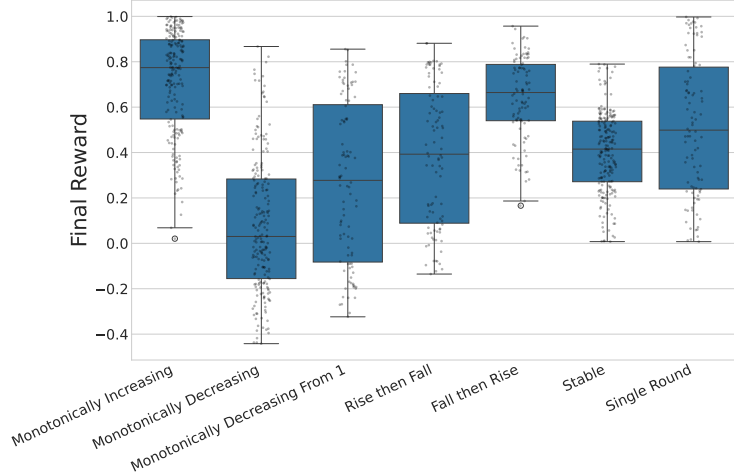


Figure 2: ATR reward analysis, showing the ATR on 0/1 reward (left) and the dense reward distribution by strategy type (right).

4 Analyze

4.1 The Role of CSMR

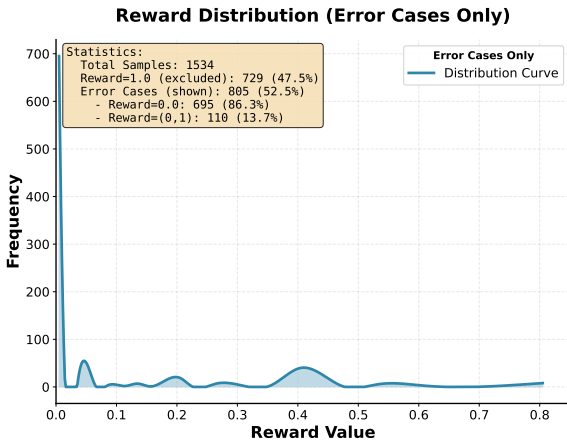


Figure 3: Turn count distribution for models using CSMR on the BIRD-Dev.

We quantified the reward distribution of the base model Qwen2.5-7B-Instruct on the Bird dataset. We can clearly see that in Figure 3, among the 52.5% of data points that are not completely correct, 13.7% of the data contains more information, as its response is partially correct. However, the traditional 0/1 distribution completely missed this information, which the CSMR mechanism can capture.

4.2 The Role of ATR

Effective reward design hinges on a clear **reward shape**, enabling the model to gauge performance and optimize strategically. We theoretically ex-

amine ATR’s guidance toward optimal trajectories, demonstrated under two distinct signals in Figure 2.

Under the **binary signal**, we prioritize reaching the correct solution efficiently (i.e., fewer steps or simpler logic); here, a persistent incorrect trajectory (e.g., $[0, 0]$) receives a -0.10 baseline, which is significantly higher than the reward for a **regressive trajectory** (e.g., $[1, 0]$, -0.60). This disparity penalizes abandoning validated correct states, effectively suppressing policy oscillations and enforcing stability across turns. Conversely, under the **dense signal**, we analyze trajectories within a three-turn window, rewarding their morphological “shape” (e.g., monotonic increase, rise-then-fall, or stagnation). Results show that “**Monotonically Increasing**” trajectories receive the highest average reward, encouraging beneficial exploration toward convergence. Crucially, high-quality **Dense Rewards** are indispensable for internalizing gradual refinement, underscoring the synergy between **CSMR** (providing high-fidelity signals) and **ATR** (integrating signals into a coherent trajectory evaluation).

4.3 Training Efficiency Comparison

We use the async setup in Verl, where, in each step, it connects vLLM inference and tool calls to execute a unified rollout, and then calculates the $\log p$ to update the gradients. Therefore, we compared the rollout time and the total time for the Agentic and single-turn methods at each step, as shown in Figure 4. The execution time for the Agentic rollout is almost twice that of the single-turn method. The rollout phase accounts for the

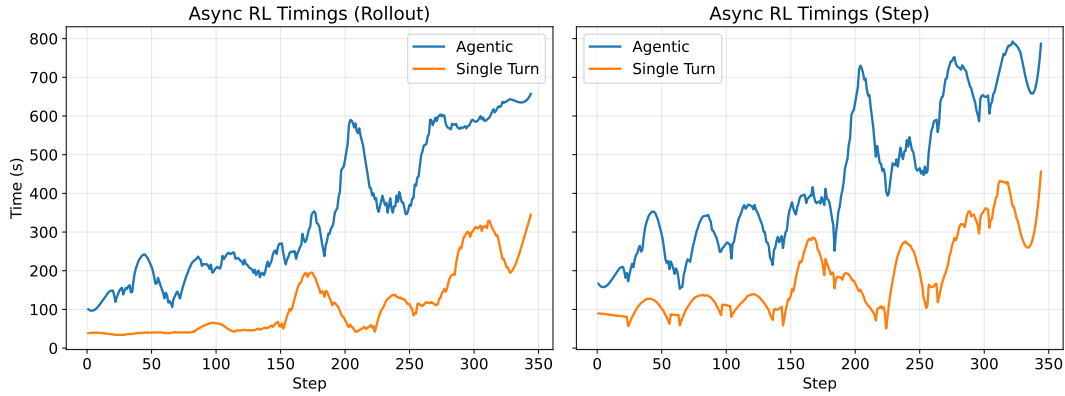


Figure 4: Agentic vs. Single-Turn training speed comparison.

longest duration. The update phase is also longer for Agentic because it involves a longer response, resulting in a total time that is twice that of the single-turn method.

5 Related Work

RLVR for Single-Turn Text-to-SQL A more recent development is RLVR (Yue et al., 2025), a promising strategy that is part of a broader trend of boosting LLM reasoning (e.g., OpenAI’s O-series (OpenAI, 2025), DeepSeek-R1 (Guo et al., 2025b), and Kimi K1.5 (Team, 2025)), especially in areas like mathematics, coding, and reasoning (Shao et al., 2024; Team, 2024; Li et al., 2024b). Recent reasoning-oriented models have primarily targeted single-turn Text-to-SQL tasks. STaR-SQL (He et al., 2025) utilizes rationale-based SFT. Reasoning-SQL (Pourreza et al., 2025) and SQL-R1 (Ma et al., 2025) apply reinforcement learning to ensure logical and execution consistency; notably, Reasoning-SQL (Pourreza et al., 2025) also designed an n-gram-based reward mechanism to provide more fine-grained feedback. These methods fail to surpass single-turn bottlenecks due to a lack of interactive environment verification.

Multi-turn Agent RL Early multi-turn LLM methods relied heavily on prompting: ACT-SQL (Zhang et al., 2023) rewrites multi-turn queries into single-turn inputs via Chain-of-Thought prompting, while CoE-SQL (Zhang et al., 2024) incrementally edits the prior SQL. Both of these methods depend on closed-source GPT models and lack database verification or self-correction mechanisms. In contrast, Agent models like Search-R1 (Jin et al., 2025), WebAgent-R1 (Wei et al., 2025), verltool (Jiang et al., 2025) and Re-

tool (Feng et al., 2025) extend reasoning capabilities through multi-turn interactions with an environment. Applying this paradigm to the SQL domain, MTSQL-R1 (Guo et al., 2025c) has adopted multi-turn execution to call SQL statements, using reinforcement learning to train the model’s interaction with the external database. However, these reinforcement learning methods mostly rely on a binary reward based on the final answer. In a long-horizon, multi-turn trajectory, this binary reward signal is extremely sparse, making it difficult for the model to effectively distinguish the quality of the reasoning process (Huo et al., 2025; Cui et al., 2025; Wu et al., 2023; Luo et al., 2024).

6 Conclusion

In this paper, we addressed the critical bottleneck of sparse-feedback-driven credit assignment in Agentic RL, focusing our experimental validation on the Text-to-SQL task domain. We proposed Agentic SQL along with a universal two-tiered reward mechanism, which not only resolves the challenges associated with sparse reward signals but also advances the SOTA in Text-to-SQL generation. A key contribution of this work is the integration of *Lyapunov stability theory* into the reinforcement learning reward design. We rigorously proved that our asymmetric transition matrix acts as an energy dissipation operator, mathematically guaranteeing a cycle-free policy and ensuring asymptotic stability throughout the reasoning process. Detailed ablation experiments highlight the specific contribution of each design component. These results validate that dense, process-oriented signals are essential for bridging the gap between LLMs reasoning and real-world database interactions.

531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579

Limitations

Although our framework significantly improves the performance of agentic SQL, several limitations remain to be addressed in future work:

Computational Overhead and Latency The transition from a single-turn paradigm to a multi-turn agentic framework inevitably increases computational costs. Our experimental analysis shows that the execution time for the agentic rollout is nearly twice that of the single-turn method, with the rollout phase accounting for the longest duration. This increased latency, combined with longer response lengths during the update phase, poses challenges for real-time deployment in latency-sensitive environments.

Fixed Interaction Horizon Our current implementation adopts a finite-horizon MDP with a system prompt that limits the agent to a maximum of three tool calls. While this prevents infinite loops and controls costs, it may restrict the model’s ability to solve extremely complex, multi-step reasoning tasks in enterprise-grade workflows that require more extensive exploration.

Hyperparameter Dependency in Reward Design While we leverage Lyapunov stability theory to provide theoretical guarantees for convergence, the practical implementation still relies on several heuristic hyperparameters. These parameters include the stagnation threshold τ within the ATR mechanism. Although our ablation experiments demonstrate that the system exhibits a certain degree of robustness, these parameters may require further fine-tuning when the framework is applied to database domains with significantly different architectural complexities or data distributions.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alnur Ali, Ashutosh Baheti, Jonathan Chang, Ta-Chung Chi, Brandon Cui, Andrew Drozdov, Jonathan Frankle, Abhay Gupta, Pallavi Koppol, Sean Kulinski, and 1 others. 2025. A state-of-the-art sql reasoning model using rlvr. *arXiv preprint arXiv:2509.21459*.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen,

Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, and 6 others. 2025. *Process reinforcement through implicit rewards*. *Preprint*, arXiv:2502.01456.

Minghang Deng, Ashwin Ramachandran, Canwen Xu, Lanxiang Hu, Zhewei Yao, Anupam Datta, and Hao Zhang. 2025. *RefoRCE: A text-to-SQL agent with self-refinement, format restriction, and column exploration*. In *ICLR 2025 Workshop: VerifAI: AI Verification in the Wild*.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. *Retool: Reinforcement learning for strategic tool use in llms*. *arXiv preprint arXiv:2504.11536*.

Jonathan Fürst, Catherine Kosten, Farhad Nooralahzadeh, Yi Zhang, and Kurt Stockinger. 2024. *Evaluating the data model robustness of text-to-sql systems based on real user queries*. *arXiv preprint arXiv:2402.08349*.

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

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

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, and 1 others. 2024. *Deepseek-coder: When the large language model meets programming—the rise of code intelligence*. *arXiv preprint arXiv:2401.14196*.

Taicheng Guo, Hai Wang, ChaoChun Liu, Mohsen Golaikhani, Xin Chen, Xiangliang Zhang, and Chandan K Reddy. 2025c. *Mtsql-r1: Towards long-horizon multi-turn text-to-sql via agentic training*. *arXiv preprint arXiv:2510.12831*.

Mingqian He, Yongliang Shen, Wenqi Zhang, Qiuying Peng, Jun Wang, and Weiming Lu. 2025. *Star-sql: Self-taught reasoner for text-to-sql*. *Preprint*, arXiv:2502.13550.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, and 1 others. 2025. *Deep research agents: A systematic examination and roadmap*. *arXiv preprint arXiv:2506.18096*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajuan Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang,

636	Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report . <i>Preprint</i> , arXiv:2409.12186.	692
637		693
638		694
639	Nan Huo, Xiaohan Xu, Jinyang Li, Per Jacobsson, Shipai Lin, Bowen Qin, Binyuan Hui, Xiaolong Li, Ge Qu, Shuzheng Si, and 1 others. 2025. Bird-interact: Re-imagining text-to-sql evaluation for large language models via lens of dynamic interactions. <i>arXiv preprint arXiv:2510.05318</i> .	695
640		696
641		697
642		698
643		699
644		700
645	Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, Tianyu Pang, and Wenhui Chen. 2025. Verl-tool: Towards holistic agentic reinforcement learning with tool use . <i>Preprint</i> , arXiv:2509.01055.	701
646		702
647		703
648		704
649		705
650	Bowen Jin, Hansi Zeng, Zhenrui Yue, and 1 others. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. <i>arXiv preprint arXiv:2503.09516</i> .	706
651		707
652		708
653		709
654		710
655	Heegy Kim, Taeyang Jeon, Seunghwan Choi, Seungtaek Choi, and Hyunsouk Cho. 2024. Flex: Expert-level false-less execution metric for reliable text-to-sql benchmark. <i>arXiv preprint arXiv:2409.19014</i> .	711
656		712
657		713
658		714
659	Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, SU Hongjin, ZHAOQING SUO, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, and 1 others. 2025a. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. In <i>The Thirteenth International Conference on Learning Representations</i> .	715
660		716
661		717
662		718
663		719
664		720
665	Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, and 1 others. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. <i>arXiv preprint arXiv:2411.07763</i> .	721
666		722
667		723
668		724
669		725
670		726
671		727
672	Fangyu Lei, Jinxiang Meng, Yiming Huang, Tinghong Chen, Yun Zhang, Shizhu He, Jun Zhao, and Kang Liu. 2025b. Reasoning-table: Exploring reinforcement learning for table reasoning. <i>arXiv preprint arXiv:2506.01710</i> .	728
673		729
674		730
675		731
676		732
677	Remco I Leine. 2010. The historical development of classical stability concepts: Lagrange, poisson and lyapunov stability. <i>Nonlinear Dynamics</i> , 59(1):173–182.	733
678		734
679		735
680	Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tiejing Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025a. Omnisql: Synthesizing high-quality text-to-sql data at scale . <i>Proc. VLDB Endow.</i> , 18(11):4695–4709.	736
681		737
682		738
683		739
684		740
685		741
686	Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2024a. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. <i>Advances in Neural Information Processing Systems</i> , 36.	742
687		743
688		744
689		745
690		746
691		
	Long Li, Jiaran Hao, Jason Klein Liu, Zhijian Zhou, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, and 1 others. 2025b. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. <i>arXiv preprint arXiv:2509.07430</i> .	
	Long Li, Xuzheng He, Haozhe Wang, Linlin Wang, and Liang He. 2024b. How do humans write code? large models do it the same way too . In <i>Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing</i> , pages 4638–4649, Miami, Florida, USA. Association for Computational Linguistics.	
	Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision . <i>ArXiv</i> , abs/2406.06592.	
	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 . <i>arXiv preprint arXiv:2504.08600</i> .	
	OpenAI. 2025. Openai o3 and o4-mini system card. Accessed: 2025-05-15.	
	Mohammadreza Pourreza, Shayan Taleai, 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. <i>arXiv preprint arXiv:2503.23157</i> .	
	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. <i>arXiv preprint arXiv:2402.03300</i> .	
	Adrien Taylor, Bryan Van Scoy, and Laurent Lessard. 2018. Lyapunov functions for first-order methods: Tight automated convergence guarantees . In <i>Proceedings of the 35th International Conference on Machine Learning</i> , volume 80 of <i>Proceedings of Machine Learning Research</i> , pages 4897–4906. PMLR.	
	K Team. 2025. Kimi k1.5: Scaling reinforcement learning with llms. <i>arXiv preprint arXiv:2501.12599</i> .	
	Qwen Team. 2024. Qwen2.5: A party of foundation models .	
	Zhepei Wei, Wenlin Yao, Yao Liu, and 1 others. 2025. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. <i>arXiv preprint arXiv:2505.16421</i> .	
	Zequi Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training . In <i>Advances in Neural</i>	

747		original natural language question q , the database	800
748	<i>Information Processing Systems</i> , volume 36, pages	schema $\mathcal{D}_{\text{schema}}$, and the sequence of past actions	801
	59008–59033. Curran Associates, Inc.	and observations.	802
749	Renjun Xu and Jingwen Peng. 2025. A comprehensive	Action Space (\mathcal{A}): The structured tool-use sig-	803
750	survey of deep research: Systems, methodologies,	nals parsed from the complete response gener-	804
751	and applications. <i>arXiv preprint arXiv:2506.12594</i> .	ated by the policy π_θ . Primary tools include	805
752	Zekun Xu, Siyu Xia, Chuhuai Yue, Jiajun Chai,	<code>execute_sql(query: str)</code> and <code>finish(answer:</code>	806
753	Mingxue Tian, Xiaohan Wang, Wei Lin, Haoxuan	<code>str)</code> .	807
754	Li, and Guojun Yin. 2025. Mtir-sql: Multi-turn tool-	Transition Dynamics (\mathcal{P}) and Rethink Mecha-	808
755	integrated reasoning reinforcement learning for text-	nism: The dynamics are determined by the agent-	809
756	to-sql . <i>Preprint</i> , arXiv:2510.25510.	environment interaction and, crucially, a rethink	810
757	Zhewei Yao, Guoheng Sun, Lukasz Borchmann, Zheyu	mechanism activated upon receiving the database	811
758	Shen, Minghang Deng, Bohan Zhai, Hao Zhang, Ang	observation o_t . This mechanism enables the agent	812
759	Li, and Yuxiong He. 2025. Arctic-text2sql-r1: Simple	to autonomously evaluate the result and decide	813
760	rewards, strong reasoning in text-to-sql. <i>arXiv</i>	whether to continue the iteration or terminate.	814
761	<i>preprint arXiv:2505.20315</i> .	Process-based Reward Function (\mathcal{R}): Unlike	815
762	Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga,	other agent frameworks, the core of our ap-	816
763	Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning	proach is a dense, process-based reward function ,	817
764	Yao, Shanelle Roman, and 1 others. 2018. Spider:	$\mathcal{R}(s_t, a_t)$. This function has two distinguishing	818
765	A large-scale human-labeled dataset for complex and	properties: first, it provides fine-grained feedback	819
766	cross-domain semantic parsing and text-to-sql task.	for scoring the outcome, rather than a single binary	820
767	In <i>2018 Conference on Empirical Methods in Natural</i>	reward; second, we introduce a fused, composite	821
768	<i>Language Processing, EMNLP 2018</i> , pages 3911–	reward signal that offers a fine-grained reflection	822
769	3921. Association for Computational Linguistics.	of our trajectory’s intermediate steps, rather than	823
770	Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai	judging based solely on the final result. This ap-	824
771	Wang, Yang Yue, Shiji Song, and Gao Huang. 2025.	proach effectively mitigates the credit assignment	825
772	Does reinforcement learning really incentivize rea-	problem inherent in sparse, outcome-only rewards.	826
773	soning capacity in LLMs beyond the base model? In	B Proof of Theorem 1 (Strict	827
774	<i>2nd AI for Math Workshop @ ICML 2025</i> .	Dissipativity)	828
775	Fanghai Zhang and Zhigang Zeng. 2018. Multiple la-	To theoretically justify how ATR prunes limit cy-	829
776	grange stability under perturbation for recurrent neu-	cles, we model the agent’s reasoning as a discrete-	830
777	ral networks with time-varying delays. <i>IEEE Trans-</i>	time dynamical system on a semantic state space	831
778	<i>actions on Systems, Man, and Cybernetics: Systems</i> ,	\mathcal{S} .	832
779	50(6):2029–2041.	Definition 1 (Lagrange Stability). <i>A reasoning sys-</i>	833
780	Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin,	<i>tem is Lagrange stable if every trajectory $\tau =$</i>	834
781	Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li,	<i>$\{s_1, s_2, \dots\}$ remains within a bounded subset of</i>	835
782	Xiangyuan Xue, Yijiang Li, and 1 others. 2025a. The	<i>\mathcal{S}. In the context of Agentic RL, this represents a</i>	836
783	landscape of agentic reinforcement learning for llms:	<i>state where the agent oscillates indefinitely (limit</i>	837
784	A survey. <i>arXiv preprint arXiv:2509.02547</i> .	<i>cycles) without diverging or converging.</i>	838
785	Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen	Definition 2 (Strict Dissipativity). <i>The system sat-</i>	839
786	Xu, and Kai Yu. 2023. Act-sql: In-context learning	<i>satisfies the Strict Dissipativity Condition if for any</i>	840
787	for text-to-sql with automatically-generated chain-of-	<i>non-trivial closed cycle \mathcal{C}, the accumulated reward</i>	841
788	thought . <i>Preprint</i> , arXiv:2310.17342.	<i>is strictly negative: $\sum_{t \in \mathcal{C}} R_t < 0$. This implies</i>	842
789	Hanchong Zhang, Ruisheng Cao, Hongshen Xu,	<i>a net loss of “energy” over any repetitive path,</i>	843
790	Lu Chen, and Kai Yu. 2024. Coe-sql: In-context	<i>preventing the agent from remaining in a loop.</i>	844
791	learning for multi-turn text-to-sql with chain-of-	Theorem 1 (Elimination of Limit Cycles). <i>The</i>	845
792	editions . <i>Preprint</i> , arXiv:2405.02712.	<i>optimal policy π^* is cycle-free if and only if the</i>	846
793	Yuhao Zhang, Shaoming Duan, Jinhang Su, Chuanyi		
794	Liu, and Peiyi Han. 2025b. Spft-sql: Enhancing large		
795	language model for text-to-sql parsing by self-play		
796	fine-tuning. <i>arXiv preprint arXiv:2509.03937</i> .		
797	A Key Components of the MDP		
798	State Space (\mathcal{S}): The state $s_t \in \mathcal{S}$ encapsu-		
799	lates the complete interaction history, including the		

reward transition function satisfies the strict dissipativity condition.

Proof. Consider a potential limit cycle \mathcal{C} consisting of a **regression** step $s_{\text{High}} \rightarrow s_{\text{Low}}$ (with reward $R_{\text{High} \rightarrow \text{Low}}$) and a **restoration** step $s_{\text{Low}} \rightarrow s_{\text{High}}$ (with reward $R_{\text{Low} \rightarrow \text{High}}$).

1. **Symmetric Case:** If the reward scheme is symmetric, $|R_{\text{High} \rightarrow \text{Low}}| = |R_{\text{Low} \rightarrow \text{High}}|$, then $\sum_{t \in \mathcal{C}} R_t = 0$. The system is Lagrange stable but lacks a driving force to exit the loop.
2. **Asymmetric Case (ATR):** Our ATR design follows the matrix \mathcal{M} , imposing $|R_{\text{High} \rightarrow \text{Low}}| > |R_{\text{Low} \rightarrow \text{High}}|$. Let $R_{\text{High} \rightarrow \text{Low}} = -k \cdot r$ and $R_{\text{Low} \rightarrow \text{High}} = r$ where $k > 1$. The total cycle return becomes:

$$\sum_{t \in \mathcal{C}} R_t = (1 - k) \sum r < 0 \quad (8)$$

3. **Conclusion:** Per Lyapunov stability theory, since every cycle incurs a net cost, the system acts as an **energy dissipation operator**. This net loss forces the trajectory to terminate at the global maximum, effectively pruning all limit cycles from the policy space.

C Training Detail

Table 2: Hyperparameters for RL Training

Hyperparameter	Value
Batch Size	128
Learning Rate	1e-6
Rollout Temperature	1.0
Rollout Top-p	0.95
Validation Temperature	0.6 / 1.0
Validation Top-p	0.95
PPO Epochs	1
Max Response Length	2048 / 4096
Number of Rollouts	8
Training Epochs	5
GPU Con	32 * A800
Training Data	8,958/17614
B	2
α	0.8
C_{turn}	0.0001
τ	0.6

We utilize the **BIRD** training set to train our models. This dataset comprises 9,428 question-SQL pairs from 70 databases across diverse domains such as airlines, movies, and sales. We filtered out samples where the gold SQL query failed to execute, resulting in a final set of 8,958 training instances. For evaluation, we primarily use

the **BIRD-Dev** set to assess in-domain capabilities. We introduce the **Spider** benchmark to test the model’s out-of-distribution (OOD) generalization. Additionally, we leverage the challenging enterprise-grade **Spider-2.0** dataset to evaluate the model’s *Agent* capabilities. This dataset targets highly difficult real-world scenarios, representing a strong indicator of practical Text-to-SQL deployment readiness. Specifically, we extract a subset of 135 locally executable examples from Spider-2.0-Lite. To ensure a fair comparison, we evaluate both SQL-R1 and Arctic-Text2SQL-R1-7B using the identical configuration. Throughout all evaluations, we adopt a 0/1 reward mechanism, where a query is considered correct only if all rows in the execution result match the ground truth exactly.

Our experiments employed two sets of trials. The first set involved the Qwen2.5-7B-Instruct model, which underwent Reinforcement Learning (RL) training directly without cold start. The second set involved the OmniSQL model, which required the Format-6k fine-tuning step to teach it the tool-calling format. Format-6k denotes that a subset of data was randomly sampled from the training set and formatted into the tool-calling format.

During our experimentation, we observed that general-purpose code models, such as Qwen2.5-Coder (Hui et al., 2024), are overly specialized. This specialization leads to poor *instruction-following* performance, rendering them unable to correctly learn the required Agent format. Even when SFT techniques were applied to correct their output format, the model failed to exhibit strong exploratory capabilities in subsequent multi-turn generation tasks.

For the RL framework, we utilize the open-source library VERL; for the Spider 2.0 evaluation, we employ the open-source repository ReFoRCE (Deng et al., 2025).

The hyperparameter configurations are detailed in Table 2. A single Reinforcement Learning experiment for Agentic SQL takes approximately 24 hours to complete. Specifically, $\alpha = 0.8$ serves as the scaling factor for the CSMR to cap structural matches at 0.8, effectively distinguishing them from truly perfect matches. The parameter $B = 2$ defines the reward clipping boundary to constrain the total reward within $[-2, 2]$ for training stability. Additionally, $C_{\text{turn}} = 0.0001$ introduces a turn penalty to incentivize concise reasoning, while $\tau = 0.6$ acts as a stagnation threshold to determine state transitions for the ATR calculation.

D Turn Correct Distribution

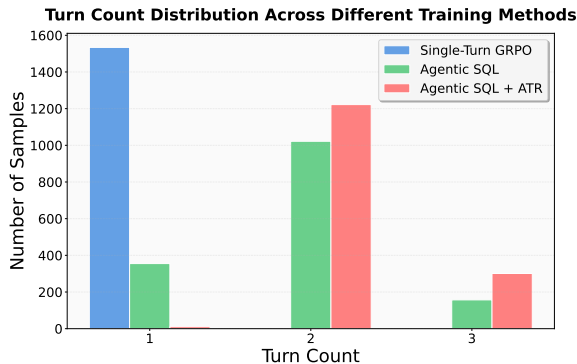


Figure 5: Turn count distribution for models using CSMR on the BIRD-Dev.

Table 3: Model Performance on Turn- n Interactions.

Model	Turn 1	Turn 2	Turn 3
Single-Turn GRPO	59.4	-	-
Agentic SQL	62.2	66.0	39.5
Agentic SQL + ATR	72.7	71.0	36.2

Figure 5 presents the turn-number distribution (i.e., the percentage of generations using 2 turns, 3 turns, etc.) for the RL-trained models. Conversely, Table 3 reports the accuracy achieved by these models at a specified number of N turns.

We observe that different reward functions result in markedly distinct distributions. The model trained without ATR exhibits a more balanced distribution, generating outputs across 1, 2, and 3 turns. In contrast, the model employing ATR almost entirely abandons single-turn generation, instead focusing heavily on 2-turn and 3-turn generations, where it simultaneously achieves the highest overall accuracy.

E Additional Experiment

For experimental simplicity, we conduct these comparative trials using a single-turn GRPO baseline. The primary objective of α is to provide a clear distinction between truly perfect samples and “pseudo-perfect” matches—cases where the column values match but the row-level compositions are incorrect, such as $P = [[a1, b2], [a2, b1]]$ and $G = [[a1, b1], [a2, b2]]$. By incorporating α , the model effectively learns to differentiate between these two scenarios, thereby validating the efficacy of our approach. While “pseudo-perfect” matches may be introduced, this mechanism en-

sures they are appropriately penalized to avoid misleading the learning process.

Table 4: Sensitivity Analysis of the Scaling Factor α in CSMR.

Model	Bird-Dev
Single-Turn GRPO	
$\alpha = 0.6$	59.2
$\alpha = 0.7$	60.1
$\alpha = 0.8$	59.4

F Case Study

In this section, we present a complete multi-turn case study. It illustrates how the agent uses information about missing columns returned by the executor to iteratively refine its plan, ultimately leading to the correct result.

System Prompt

You **are** a helpful SQL assistant. You will solve the user's question by calling a tool to execute SQL code and query a SQL database. After each tool call, you must evaluate the result returned in `<tool_response>` to determine its relevance to the question, and decide for yourself whether to make another query. You may also use a tool to inspect the database schema first before providing your final SQL code. You can call the tool a maximum of 3 times.

User Prompt

Task Overview:
You **are** a data science expert. Below, you **are** provided with a database schema **and** a **natural** language question. Your task is to understand the schema **and** generate a valid SQL query to answer the question.

Database Engine:

SQLite

Database Schema:

```
CREATE TABLE frpm (
  CDSCode text, -- example:
  ['01100170109835', '01100170112607']
  `Academic Year` text, -- example:
  ['2014-2015']
  `County Code` text, -- example: ['01',
  '02']
  `District Code` integer, -- example:
  [10017, 31609]
  `School Code` text, -- example:
  ['0109835', '0112607']
)
```

```

`County Name` text, -- example: ['
Alameda', 'Alpine']
`District Name` text, -- example: ['
Alameda County Office of Education',
'California School for the Blind (
State S']
`School Name` text, -- example: ['FAME
Public Charter', 'Envision Academy
for Arts & Technology']
`District Type` text, -- example: ['
County Office of Education (COE)', '
State Special Schools']
`School Type` text, -- example: ['K-12
Schools (Public)', 'High Schools (
Public)']
`Educational Option Type` text, --
example: ['Traditional', 'Juvenile
Court School']
`NSLP Provision Status` text, --
example: ['Breakfast Provision 2', '
Provision 2']
`Charter School (Y/N)` integer, --
example: [1, 0]
`Charter School Number` text, --
example: ['0728', '0811']
`Charter Funding Type` text, --
example: ['Directly funded', 'Locally
funded']
IRC integer, -- example: [1, 0]
`Low Grade` text, -- example: ['K',
'9']
`High Grade` text, -- example: ['12',
'8']
`Enrollment (K-12)` real, -- example:
[1087.0, 395.0]
`Free Meal Count (K-12)` real, --
example: [565.0, 186.0]
`Percent (%) Eligible Free (K-12)`
real, -- example: [0.5197, 0.4708]
`FRPM Count (K-12)` real, -- example:
[715.0, 186.0]
`Percent (%) Eligible FRPM (K-12)`
real, -- example: [0.6577, 0.4708]
`Enrollment (Ages 5-17)` real, --
example: [1070.0, 376.0]
`Free Meal Count (Ages 5-17)` real, --
example: [553.0, 182.0]
`Percent (%) Eligible Free (Ages 5-17)
` real, -- example: [0.5168, 0.4840]
`FRPM Count (Ages 5-17)` real, --
example: [702.0, 182.0]
`Percent (%) Eligible FRPM (Ages 5-17)
` real, -- example: [0.6560, 0.4840]
`2013-14 CALPADS Fall 1 Certification
Status` integer, -- example: [1]
PRIMARY KEY (CDSCode),
CONSTRAINT fk_frpm_cdscode FOREIGN KEY
(CDSCode) REFERENCES schools (
CDSCode)
);
-- (Other tables satscores and schools are
omitted for brevity, but follow same
pattern)

```

```

CREATE TABLE satscores (
cds text, -- example:

```

```

['1010108000000', '10101080109991']
rtype text, -- example: ['D', 'S']
sname text, -- school name, example:
['FAME Public Charter', 'Envision
Academy for Arts & Technology']
dname text, -- district name, example:
['Alameda County Office of Education
', 'Alameda Unified']
cname text, -- county name, example:
['Alameda', 'Amador']
enroll12 integer, -- enrollment (1st
-12nd grade), example: [398, 62]
NumTstTskr integer, -- Number of Test
Takers, example: [88, 17]
AvgScrRead integer, -- average scores
in Reading, example: [418, 503]
AvgScrMath integer, -- average scores
in Math, example: [418, 546]
AvgScrWrite integer, -- average scores
in writing, example: [417, 505]
NumGE1500 integer, -- Number of Test
Takers Whose Total SAT Scores Are
Greater or Equal to 1500, example:
[14, 9]
PRIMARY KEY (cds),
CONSTRAINT fk_satscores_cds FOREIGN
KEY (cds) REFERENCES schools (CDSCode)
);
CREATE TABLE schools (
CDSCode text, -- example:
['01100170000000', '01100170109835']
NCESDist text, -- National Center for
Educational Statistics school
district identification number,
example: ['0691051', '0600002']
NCESSchool text, -- National Center
for Educational Statistics school
identification number, example:
['10546', '10947']
StatusType text, -- example: ['Active
', 'Closed']
County text, -- example: ['Alameda', '
Alpine']
District text, -- example: ['Alameda
County Office of Education', '
California School for the Blind (
State S']
School text, -- example: ['FAME Public
Charter', 'Envision Academy for Arts
& Technology']
Street text, -- example: ['313 West
Winton Avenue', '39899 Balentine
Drive, Suite 335']
StreetAbr text, -- street address,
example: ['313 West Winton Ave.',
'39899 Balentine Dr., Ste. 335']
City text, -- example: ['Hayward', '
Newark']
Zip text, -- example: ['94544-1136',
'94560-5359']
State text, -- example: ['CA']
MailStreet text, -- example: ['313
West Winton Avenue', '39899 Balentine
Drive, Suite 335']
MailStrAbr text, -- mailing street

```

address, example: ['313 West Winton Ave.', '39899 Balentine Dr., Ste. 335']

MailCity text, -- mailing city, example: ['Hayward', 'Newark']

MailZip text, -- mailing zip, example: ['94544-1136', '94560-5359']

MailState text, -- mailing state, example: ['CA']

Phone text, -- example: ['(510) 887-0152', '(510) 596-8901']

Ext text, -- extension, example: ['130', '1240']

Website text, -- example: ['www.acoe.org', 'www.envisionacademy.org/']

OpenDate **date**, -- example: ['2005-08-29', '2006-08-28']

ClosedDate **date**, -- example: ['2015-07-31', '2015-06-30']

Charter **integer**, -- example: [1, 0]

CharterNum text, -- example: ['0728', '0811']

FundingType text, -- example: ['Directly funded', 'Locally funded']

DOC text, -- District Ownership Code, example: ['00', '31']

DOCType text, -- The District Ownership Code Type, example: ['County Office of Education (COE)', 'State Special Schools']

SOC text, -- School Ownership Code, example: ['65', '66']

SOCType text, -- School Ownership Code Type, example: ['K-12 Schools (Public)', 'High Schools (Public)']

EdOpsCode text, -- Education Option Code, example: ['TRAD', 'JUV']

EdOpsName text, -- Educational Option Name, example: ['Traditional', 'Juvenile Court School']

EILCode text, -- Educational Instruction Level Code, example: ['ELEMHIGH', 'HS']

EILName text, -- Educational Instruction Level Name, example: ['Elementary-High Combination', 'High School']

GSoffered text, -- grade span offered, example: ['K-12', '9-12']

GSserved text, -- grade span served., example: ['K-12', '9-12']

Virtual text, -- example: ['P', 'N']

Magnet **integer**, -- example: [0, 1]

Latitude real, -- example: [37.658212, 37.521436]

Longitude real, -- example: [-122.09713, -121.99391]

AdmFName1 text, -- administrator's first name 1, example: ['L Karen', 'Laura']

AdmLName1 text, -- administrator's last name 1, example: ['Monroe', 'Robell']

AdmEmail1 text, -- administrator's email address 1, example: ['lkmonroe@aco.org', 'laura@envisionacademy.org']

AdmFName2 text, -- administrator's first name 2, example: ['Sau-Lim (Lance)', 'Jennifer']

AdmLName2 text, -- administrator's last name 2, example: ['Tsang', 'Koelling']

AdmEmail2 text, -- administrator's email address 2, example: ['stsang@unityhigh.org', 'jkoelling@efcps.net']

AdmFName3 text, -- administrator's first name 3, example: ['Drew', 'Irma']

AdmLName3 text, -- administrator's last name 3, example: ['Sarratore', 'Munoz']

AdmEmail3 text, -- administrator's email address 3, example: ['dsarratore@vincentacademy.org', 'gmunoz@piedmont.k12.ca.us']

LastUpdate **date**, -- example: ['2015-06-23', '2015-09-01']

PRIMARY KEY (CDSCode)

);

This schema describes the database's structure, including tables, columns, primary keys, foreign keys, and any relevant relationships or constraints.

Question:

Charter schools refers to `Charter School (Y/N)` = 1 in the frpm

Please list the phone numbers of the direct charter-funded schools that are opened after 2000/1/1.

Instructions:

- Make sure you only output the information that is asked in the question. If the question asks for a specific column, make sure to only include that column in the SELECT clause, nothing more.
- The generated query should return all of the information asked in the question without any missing or extra information.
- Before generating the final SQL query, please think through the steps of how to write the query.

Output Format:

In your answer, please enclose the generated SQL query in a code block:

```
```sql
-- Your SQL query
```
```

Take a deep breath and think step by step to find the correct SQL query.

For each round's response, please **first** think **step-by-step**, **and then** make the tool call according to this format:

```
<tool_call>
{
  "name": "run_sql_remote",
  "arguments": {
    "query": "```sql\n-- Your SQL query\n```",
```

```

        "db_id": "california_schools/
        california_schools.sqlite"
    }
}
</tool_call>

```

Response-Turn 1

To find the phone numbers of the direct charter-funded schools that **are** opened after 2000/1/1, we need to follow these steps:

1. Filter the schools to include **only** those with `Charter School (Y/N)` = 1 (charter schools).
2. Ensure the schools **are** direct charter-funded, which can be checked **from** the `Charter Funding Type` **column** in the `frpm` **table**.
3. Verify that the schools were opened after 2000/1/1 **using** the `OpenDate` **column** in the `schools` **table**.

Let's write the SQL query step by step:

1. Join the `frpm` table with the `schools` table using the `CDSCode` column.
2. Apply the filter for charter schools (`Charter School (Y/N)` = 1).
3. Check for direct charter funding (`Charter Funding Type` = 'Directly funded').
4. Ensure the schools were opened after 2000/1/1 (`OpenDate` > '2000-01-01').
5. Select the phone numbers from the `schools` table.

Now, let's **write** the SQL query:

```

<tool_call>
{
  "name": "run_sql_remote",
  "arguments": {
    "query": "```sql\nSELECT s.Phone\nFROM schools s\nJOIN frpm f ON s.\nCDSCode = f.CDSCode\nWHERE f.`Charter\nSchool (Y/N)` = 1 AND s.`Charter\nFunding Type` = 'Directly funded' AND\ns.OpenDate > '2000-01-01'\n```",
    "db_id": "california_schools/\ncalifornia_schools.sqlite"
  }
}
</tool_call>

```

SQL Executor Interaction Results

```

<tool_response>
error: no such column: s.Charter Funding
Type
</tool_response>

```

Response-Turn 2

I apologize for the oversight. The `Charter Funding Type` **column** is

actually **in** the `frpm` **table**, **not** the `schools` **table**. Therefore, we should **join** the `frpm` **table** with the `schools` **table** and apply the filters **using** the columns **from** the `frpm` **table**.

Let's correct the query:

1. Join the `frpm` table with the `schools` table using the `CDSCode` column.
2. Apply the filter for charter schools (`Charter School (Y/N)` = 1).
3. Check for direct charter funding (`Charter Funding Type` = 'Directly funded') in the `frpm` table.
4. Ensure the schools were opened after 2000/1/1 (`OpenDate` > '2000-01-01').
5. Select the phone numbers from the `schools` table.

Now, let's **write** the corrected SQL query:

```

<tool_call>
{
  "name": "run_sql_remote",
  "arguments": {
    "query": "```sql\nSELECT s.Phone\nFROM schools s\nJOIN frpm f ON s.\nCDSCode = f.CDSCode\nWHERE f.`Charter\nSchool (Y/N)` = 1 AND f.`Charter\nFunding Type` = 'Directly funded' AND\ns.OpenDate > '2000-01-01'\n```",
    "db_id": "california_schools/\ncalifornia_schools.sqlite"
  }
}
</tool_call>

```

SQL Executor Interaction Results

```

<tool_response>
[[None], ['(510) 596-8901'], [None], ['(510) 686-4131'], ['(510) 452-2063'], ['(510) 842-1181'], ['(510) 748-4008'], ['(510) 748-4017'], ['(510) 995-4300'], ['(510) 748-4314'], ['(510) 809-9800'], ['(510) 809-9800'], ['(510) 300-1340'], ['(510) 300-1560'],
</tool_response>

```

Response-Turn 2

The phone numbers of the direct charter-funded schools that **are** opened after 2000/1/1 **are** as follows:

```

```sql
SELECT s.Phone
FROM schools s
JOIN frpm f ON s.CDSCode = f.CDSCode
WHERE f.`Charter School (Y/N)` = 1 AND f.`Charter Funding Type` = 'Directly funded' AND s.OpenDate > '2000-01-01'
...

```