FLEX: Expert-level False-Less EXecution Metric for Reliable Text-to-SQL Benchmark

Heegyu Kim[1], Taeyang Jeon[1], Seunghwan Choi[1], Seungtaek Choi, Hyunsouk Cho[1,2*]
[1]Department of Artificial Intelligence, [2]Department of Software and Computer Engineering,
Ajou University, Suwon 16499, Republic of Korea
`{khk6435, dnwn3311, dexrf, hyunsouk}@ajou.ac.kr, hist0134@naver.com`

Abstract

Text-to-SQL systems have become crucial for translating natural language into SQL queries in various industries, enabling non-technical users to perform complex data operations. The need for accurate evaluation methods has increased as these systems have grown more sophisticated. However, the Execution Accuracy (EX), the most prevalent evaluation metric, still shows many false positives and negatives. Thus, this paper introduces **FLEX (False-Less Execution)**, a novel approach to evaluating text-to-SQL systems using large language models (LLMs) to emulate human expert-level evaluation of SQL queries. Our metric improves agreement with human experts (from 62 to 87.04 in Cohen's kappa) with comprehensive context and sophisticated criteria. Our extensive experiments yield several key insights: (1) Models' performance increases by over 2.6 points on average, substantially affecting rankings on Spider and BIRD benchmarks; (2) The underestimation of models in EX primarily stems from annotation quality issues; and (3) Model performance on particularly challenging questions tends to be overestimated. This work contributes to a more accurate and nuanced evaluation of text-to-SQL systems, potentially reshaping our understanding of state-of-the-art performance in this field.

1 Introduction

Text-to-SQL systems, which translate natural language questions into SQL queries, have become vital across various industries by democratizing data access and facilitating data-driven decision-making (Shi et al., 2024; Hong et al., 2024). As these systems grow more complex, accurate and efficient evaluation methods are increasingly important. While initial evaluations relied on human experts, this approach proved too time-consuming and costly for large-scale assessments. To address this, benchmarks like Spider (Yu et al., 2018) introduced the Execution Accuracy (EX) metric, which has been widely adopted by others such as BIRD (Li et al., 2023c).

Table 1: BIRD top 10 leaderboard re-ranked by **FLEX**. Δ denotes underestimation error (FLEX − EX).

Rank	Model	FLEX	EX	Δ
1 (↑2)	SuperSQL 👑	**64.08**	57.37	+6.71
2 (↓1)	CHESS-GPT-4o-mini	62.71	**59.13**	+3.59
3 (↑2)	TA-ACL	59.97	55.67	+4.30
4 (↑3)	DAIL_SQL_9-SHOT_MP	59.26	53.52	+5.74
5 (↑4)	DAIL_SQL_9-SHOT_QM	58.47	53.06	+5.41
5 (↓3)	DTS-SQL-BIRD-GPT4o	58.47	58.08	+0.39
7 (↓3)	SFT_CodeS_15B_EK	56.98	56.52	+0.46
8 (↓2)	SFT_CodeS_7B_EK	53.59	54.89	−1.30
9 (↓1)	SFT_CodeS_3B_EK	53.26	53.46	−0.20
10 (↑2)	DAIL_SQL	51.83	45.89	+5.93

However, the EX metric, which evaluates queries based on their execution result, has significant limitations that can lead to assessments diverging from human experts. Our analysis reveals that EX incorrectly validates queries due to coincidental database states or unfairly penalizes correct ones because of ambiguous questions, thus misestimating a model's ability to generate correct and valid SQL queries (Pourreza and Rafiei, 2023b).

Although LLM-based evaluation methods (Kim et al., 2023; Zheng et al., 2023; Zhao et al., 2024) have been proposed, they exhibit suboptimal performance due to limited context. They fail to utilize crucial contextual information, struggle with noisy ground truth issues, and employ ambiguous criteria inappropriate for text-to-SQL evaluation. Consequently, they even underperform the traditional EX metric when evaluated against established text-to-SQL benchmarks.

To address these issues, we propose **FLEX (False-Less EXecution)**, a novel approach leveraging LLMs to emulate expert-level evaluation of SQL queries. **FLEX** is specifically designed for evaluating text-to-SQL tasks and

---

[*] Corresponding author

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4448–4475
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

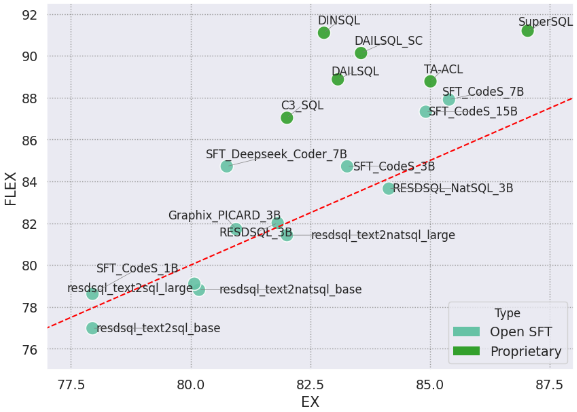


Figure 1: Performance comparison of EX vs. FLEX metrics on Spider benchmark. The red identity line shows an equivalent score.

overcomes existing limitations through comprehensive context analysis, considering the natural language question, database schema, and external knowledge. It also exploits elaborately devised evaluation criteria with detailed guidelines for assessing query correctness. Our approach is validated through strong consistency between **FLEX** judgments and human expert evaluations, demonstrating significantly higher agreement (Cohen's kappa of 87.04) than the existing EX metric (62.00) and outperforming previous LLM-based methods.

To demonstrate the practical impact of **FLEX**, we re-evaluated 50 publicly available text-to-SQL models on the Spider and BIRD[1] benchmarks, revealing several key insights. As shown in Table 1, significant shifts in model rankings, particularly for BIRD, provide a more accurate assessment of model capabilities. Additionally, **FLEX** categorized the error cases and identified noisy annotations as a primary factor in underestimating model performance in current benchmarks. As illustrated in Fig. 1, **FLEX** scores are generally higher than EX scores, suggesting that **FLEX** captures previously underestimated aspects of model capability. Furthermore, **FLEX** detected models are overestimated in BIRD's challenging questions, highlighting areas for future research focus.

Our key contributions are as follows: 1) we identify key limitations in current text-to-SQL evaluation metrics, especially the Execution Accuracy (EX) metric, and 2) we introduce **FLEX**, a novel LLM-based evaluation method that aligns with human expert reasoning and shows better agreement with expert assessments. 3) By re-evaluating 50 text-to-SQL models using **FLEX**, we uncover significant ranking changes and provide a clearer understanding of benchmarks and state-of-the-art models, emphasizing the need for advanced evaluation methods. **FLEX** framework is publicly available[2].

2 Related Work

Exact Matching (EM) and Execution Accuracy (EX) were proposed to evaluate text-to-SQL systems in the Spider (Yu et al., 2018). EM assesses the syntax-level equivalence of two queries; however, it is prone to high false negative rates because the same logical intent can be expressed in various query formulations. Therefore, EX has been utilized as a primary metric for query accuracy in BIRD (Li et al., 2023c), focusing on the execution results rather than the syntactic form. Despite this improvement, EX still suffers from false negatives due to annotation quality issues and may produce false positives by rewarding incorrect queries that coincidentally yield the same results as the ground truth. These limitations underscore the necessity for more robust semantic evaluation approaches.

LLM-based evaluation has gained popularity as an alternative to human evaluators for assessing human preference in natural language generation models. Frameworks like MT-Bench (Zheng et al., 2023) and AlpacaEval (Li et al., 2023e) utilize LLMs as judges to provide faster and more cost-effective evaluations than human assessments. However, these frameworks primarily rely on the proprietary GPT-4 (OpenAI et al., 2024), which poses reproducibility risks since providers may alter or discontinue models without notice. In response, Kim et al. introduced the open-source LLM judge *Prometheus*, offering a reproducible and cost-free alternative at GPT-4's performance level. Nonetheless, *Prometheus* is mainly focused on general human preference evaluations rather than specialized tasks like text-to-SQL evaluation. To specifically leverage LLMs for text-to-SQL evaluation, Zhao et al. proposed *LLM-SQL-Solver*. Without distinguishing between the false positives and false negatives, this approach showcases the capability of LLMs to assess execution equivalence between two queries using two prompting strategies: *Miniature-and-Mull*, evaluating seman-

---

[1] Our study utilized the 20240627 version of the BIRD-dev set, yielding different EX scores from prior studies due to modified ground truth.

[2] `https://github.com/HeegyuKim/FLEX`

tic equivalence between two queries, and *Explain-and-Compare*, asking LLMs to compare their significant logical differences. Despite its advancements, *LLM-SQL-Solver* remains insufficient to fully replace EX as an evaluation metric. We will investigate these detailed limitations in section 4.3.

3 Preliminaries

This section introduces the notation and definitions used throughout the paper. We begin by establishing the following conventions: Let D denote a database, S its schema, and X a set of natural language questions. For each question $x \in X$, we define contextual information $\mathbb{C}$ to assist in the evaluation of metrics, including $Q_{gt}(x)$ as the ground truth SQL query for question x, and $Q_{gen}(x)$ as the SQL query generated by a text-to-SQL model for x. We then execute these queries on the database D to obtain their execution result sets. Specifically, we define $R_{gt}(x) = \text{Execute}(Q_{gt}(x), D)$ as the result of the ground truth query and $R_{gen}(x) = \text{Execute}(Q_{gen}(x), D)$ as the result of the generated query. We define evaluation metrics: EM, EX, and LLM-based evaluation (LX) as:

$$\text{EM} = \frac{1}{|X|} \sum_{x \in X} \mathbb{I}\left(Q_{gen}(x) = Q_{gt}(x)\right) \tag{1}$$

$$\text{EX} = \frac{1}{|X|} \sum_{x \in X} \mathbb{I}\left(R_{gen}(x) = R_{gt}(x)\right) \tag{2}$$

$$\text{LX} = \frac{1}{|X|} \sum_{x \in X} \mathbb{I}\left(x \underset{\mathbb{C}}{\leftrightarrow} Q_{gen}(x)\right) \tag{3}$$

where $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition inside the parentheses is true, and 0 otherwise. The notation $x \underset{\mathbb{C}}{\leftrightarrow} Q_{gen}(x)$ denotes that the generated query $Q_{gen}(x)$ semantically matches the question x under the contextual information $\mathbb{C}$ based on an LLM.

For EM, the condition checks if $Q_{gen}(x)$ exactly matches the $Q_{gt}(x)$ syntactically. For EX, the condition checks if the $R_{gen}(x)$ exactly matches the $R_{gt}(x)$. However, LX handles semantic match between $Q_{gen}(x)$ and question x under contextual information. $\mathbb{C}$ may vary depending on the evaluation method. These metrics produce a score between 0 and 1, where 1 indicates perfect performance (100% accuracy) and 0 indicates complete failure (0% accuracy).

Table 2: Illustrated examples of false positive / negative.

Question: Who has the highest score?
Ground Truth Query

SELECT fname, lname FROM student
ORDER BY score DESC LIMIT 1
> Emily, Carter

Example 1: False Positive

SELECT fname, lname FROM student
WHERE age < 19 # Unnecessary condition.
ORDER BY score DESC LIMIT 1
> Emily, Carter

Example 2: False Negative

SELECT **lname, fname** FROM student
ORDER BY score DESC LIMIT 1
> Carter, Emily # The column order is different

Example 3: False Negative

SELECT fname, lname FROM student
WHERE score == (SELECT MAX (score) FROM student)
> Emily, Carter | Liam, Thompson
# Two students have the highest score. 	

4 Analyzing the Limitations of Current Text-to-SQL Evaluation Methods

In this section, we address key research questions to investigate the shortcomings of existing text-to-SQL evaluation metrics. We aim to understand the types of errors these metrics produce and explore whether LLMs can offer a better alternative.

4.1 What Types of Errors Occur in Text-to-SQL Evaluation?

To identify the limitations of current evaluation methods, we first examine the errors that arise when using the Execution Accuracy (EX) metric: false positives (FP) and false negatives (FN) using Table 2.

False Positives ($x \underset{\mathbb{C}}{\not\leftrightarrow} Q_{gen}(x)$, but $R_{gen}(x) = R_{gt}(x)$): The generated query $Q_{gen}(x)$ with semantically different structures or logic may produce the same execution results due to the current database states, leading to overestimating system performance. *Example 1* includes an unnecessary condition `age < 19` but coincidentally produces the same results as the ground truth because the database has no students older than 18. Unlike humans, EX does not assess whether the generated SQL accurately represents the intent of the natural language question, potentially rewarding queries that yield correct results through incorrect means.

False Negatives ($x \underset{\mathbb{C}}{\leftrightarrow} Q_{gen}(x)$, but $R_{gen}(x) \neq R_{gt}(x)$): Semantically correct queries that accurately translate the natural language questions may produce different results from ground truth queries, underestimating system capabilities. Prior studies (Pourreza and Rafiei, 2023b; Wretblad et al., 2024) have raised concerns about annotation quality, potentially leading to false negatives during evaluation. They reported that the Spider and BIRD benchmarks have many annotation issues—at most 49% at specific databases—including ambiguous questions and incorrectly annotated ground truths.

Ambiguous questions can be translated into multiple queries due to limited instruction about the output structure. For instance, *Example 2* is marked incorrect simply because `lname` precedes `fname` in the column order. This structural rigidity in evaluation can lead to false negatives, potentially overlooking semantically correct queries. Additionally, *Example 3* shows noisy ground truth, which can occur when multiple students have the same highest score. The query handles this issue correctly but is evaluated as incorrect because its execution result differs from the ground truth.

These limitations reduce reliability and hinder the development of more robust text-to-SQL systems, potentially leading to deployed systems that fail in real-world scenarios. To address these issues, we must explore evaluation methods that align more closely with human judgment.

4.2 How Closely Does EX Align with Human Expert Evaluations?

Building on our analysis of identified errors, we conducted a human evaluation study to assess the alignment between EX and human experts in evaluating SQL query correctness. We randomly sampled 200 query pairs from the BIRD dataset, generated by TA-ACL (Qu et al., 2024) and SuperSQL (Li et al., 2024a). The samples were evenly divided between queries that produced the same result as the ground truth (*Equivalent set*) and those with different results (*Not equivalent set*). Three SQL experts with over three years of experience independently evaluated these queries for semantic correctness. Discrepancies were resolved through consensus. Before reaching a consensus, the inter-annotator agreement measured by Fleiss' kappa was 79.32, indicating strong agreement.

We employed Cohen's kappa and accuracy score to measure the agreement between human experts and EX. The agreement between human consensus and EX yielded a Cohen's kappa of 62.0, indicating substantial agreement but leaving much room for improvement, where human annotators identified 21 false positives in the *Not equivalent set* and 17 false negatives in the *Equivalent set*.

Table 3: Agreement between human consensus and other evaluation methods. **Acc** denotes overall accuracy score, **EQ** denotes *Equivalent set* accuracy score, **NEQ** denotes *Not equivalent set* accuracy score.

Model	Kappa	Acc	EQ	NEQ
EX	**62.00**	**81.0**	**79**	**83**
LLM-SQL-Solver	52.29	76.5	70	**83**
Prometheus-2-7B	61.14	80.5	78	**83**
Prometheus-2-8x7B	60.66	80.0	78	82

These findings raise concerns about the reliability of the EX metric and underscore the need for evaluation methods that better reflect human judgment by considering semantic correctness and providing a more expert-level assessment of query accuracy. Appendix C provides real examples illustrating these discrepancies.

4.3 Can LLMs Replace EX in Evaluating Text-to-SQL Systems?

Given EX's limitations and misalignment with human judgment, we explore LLMs as a potential alternative for evaluating text-to-SQL systems. We thus employed two LLM-based evaluation methods: 1) *Prometheus-2* (Kim et al., 2024), an open LM judge, and 2) *LLM-SQL-Solver* (Zhao et al., 2024) with GPT-4o (OpenAI et al., 2024), a prompting method utilizing a proprietary LLM.

Human Experts vs. LLMs: Table 3 reveals that existing LLM-based evaluation methods did not surpass EX performance. *Prometheus-2* showed a marginally lower agreement than EX, with a Cohen's kappa of 61.14, while *LLM-SQL-Solver* performed considerably worse, even when using a flagship proprietary LLM. We propose three main factors contributing to the underperformance of current LLM-based evaluation methods:

- **Limited Context and Noisy Ground Truth:** *LLM-SQL-Solver* neglects contextual information $\mathbb{C}$, including the natural language question, external knowledge [3], and execution results. It solely compares queries $Q_{gen}(x)$ to $Q_{gt}(x)$ under schema S, disregarding whether $Q_{gen}(x)$ accurately represents question x, thus limiting its ability to detect false positives and negatives. This issue is particularly critical in the Spider and BIRD benchmarks due to annotation quality

---

[3] External knowledge indicates the *evidence* field of the BIRD, assisting text-to-SQL systems to predict accurately.

concerns discussed in section 4.1.

- **Ambiguous Criteria:** The *LLM-SQL-Solver* may underperform even with advanced LLMs due to unspecified evaluation criteria. Its *Explain-and-Compare* strategy uses vague prompts like "Are there significant differences?" Similarly, *Prometheus-2* employs a 1-5 rubric scoring system, but designing optimal rubrics becomes heuristic and ambiguous, especially for scores between 2 and 4 in binary decisions (correct and incorrect)[4] We claim that LLM should evaluate SQL queries using deeper reasoning steps and several binary criteria; otherwise, the LLM shows low evaluation performance from the ambiguous criteria.

Our study empirically demonstrates that prior LLM-based evaluation methods are insufficient compared to EX for evaluating text-to-SQL systems. This highlights the need for a more advanced evaluation paradigm that leverages LLMs effectively while addressing their current limitations.

5 Our Proposed Metric: FLEX

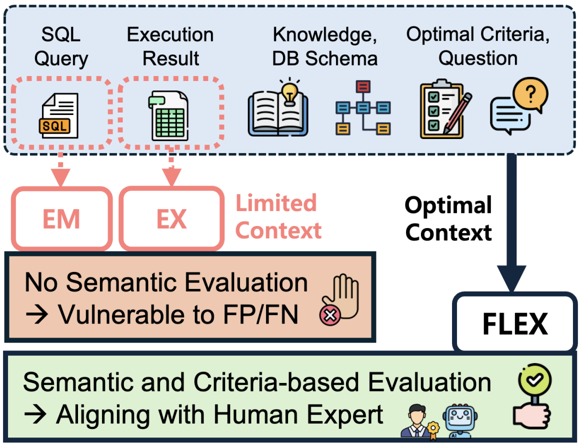


Figure 2: Compared to conventional EM and EX, **FLEX** evaluates semantic equivalence between question and query based on holistic, contextual information.

This section introduces our proposed evaluation metric **FLEX** (**False-Less EXecution**). **FLEX** addresses the limitations of existing metrics by leveraging LLMs to provide a more accurate and human-like assessment of generated SQL queries.

5.1 Evaluation Process

FLEX instructs the LLM with the question, $Q_{gen}(x)$ and optimal contextual information $\mathbb{C}_{FLEX}$, including the question x, generated query $Q_{gen}(x)$, ground truth query $Q_{gt}(x)$, execution results $R_{gt}(x)$ and $R_{gen}(x)$, schema S, external knowledge K and two criteria T_{EQ} and T_{NEQ}[5]. In other words, **FLEX** prompts LLM to focus on detecting false positives or false negatives after obtaining EX results whether $R_{gen}(x) = R_{gt}(x)$.

$$
\begin{aligned}
\mathbb{C}_{FLEX} &= \begin{cases} \mathbb{C}_{EQ} & \text{if } R_{gen}(x) = R_{gt}(x) \\ \mathbb{C}_{NEQ} & \text{if } R_{gen}(x) \neq R_{gt}(x) \end{cases} \\
\mathbb{C}_{base} &= \{x, S, K, Q_{gt}(x), Q_{gen}(x)\} \\
\mathbb{C}_{EQ} &= \mathbb{C}_{base} \cup \{T_{EQ}\} \\
\mathbb{C}_{NEQ} &= \mathbb{C}_{base} \cup \{T_{NEQ}, R_{gt}(x), R_{gen}(x)\}
\end{aligned} \tag{4}
$$

5.2 Optimal Context ($\mathbb{C}_{FLEX}$)

1) Execution Results Match ($\mathbb{C}_{EQ}$)

If $R_{gen}(x) = R_{gt}(x)$, there is a possibility of false positives—queries that produce correct results by coincidence or through incorrect logic. To evaluate false positives, 1) $\mathbb{C}_{EQ}$ does not contain $R_{gt}(x)$ and $R_{gen}(x)$. LLM occasionally judges false positives as correct due to the equivalent execution results. 2) $\mathbb{C}_{EQ}$ utilized the following criteria (T_{EQ}) to evaluate semantic correctness between the questions and $Q_{gen}(x)$:

- **Schema Alignment**: Whether the tables and columns used in $Q_{gen}(x)$ align with the intent of question q and are consistent with the schema S.
- **Correct Filtering Conditions**: Whether the WHERE clauses in $Q_{gen}(x)$ accurately reflect the conditions specified in the question x.
- **Handling of Nullable Columns**: Whether $Q_{gen}(x)$ properly handles cases where the nullable columns in aggregation functions (SUM, COUNT, AVG) of WHERE clauses, as improper handling can lead to unexpected results.
- **Accounting for Multiple Rows**: Whether $Q_{gen}(x)$ correctly handles cases where multiple rows satisfy the query conditions such as one-to-many relation and multiple min/max tuples.
- **Abusing of Clauses**: Whether clauses like GROUP BY, HAVING, ORDER BY, or DISTINCT are used appropriately, avoiding unnecessary complexity that could alter the intended results.

The LLM analyzes differences between $Q_{gen}(x)$ and $Q_{gt}(x)$ to detect logical discrepan-

---

[4] Therefore, we empirically selected rubrics with a threshold (≥ 4) yielding the highest agreement scores.

[5] Detailed prompts are provided in Appendix F.1.

cies that might not affect the current execution result but could lead to incorrect outcomes if the database state changes.

2) Execution Results Differ ($\mathbb{C}_{NEQ}$)

Although $R_{gen}(x) \neq R_{gt}(x)$, $Q_{gen}(x)$ may still be correct due to ambiguities in the question or annotation issues with the ground truth query. In this scenario, $\mathbb{C}_{NEQ}$ contains $R_{gt}(x)$, $R_{gen}(x)$, and T_{NEQ} which is designed to evaluate semantic correctness between the question and $Q_{gen}(x)$ by considering:

- **Acceptable Output Structure Variations**: Whether differences in the output structure (e.g., column order, extra or missing columns) are acceptable given the way x is phrased.
- **Representation of Values**: Whether differences in value formats (e.g., numerical precision, percentage representation, boolean values as YES/NO) are acceptable for readability and do not alter the meaning.
- **Multiple Answers Available**: Whether x is interpretable in multiple valid ways, each leading to different correct queries.
- **Incorrect Ground Truth**: Whether the ground truth query $Q_{gt}(x)$ is incorrect or suboptimal, while $Q_{gen}(x)$ correctly answers the question.

The LLM thus considers the flexibility required due to ambiguous or underspecified questions and assesses whether $Q_{gen}(x)$ is a valid translation of the question. By focusing on semantic correctness and closely aligning with human expert judgments, **FLEX** provides a more reliable evaluation of text-to-SQL models than traditional execution-based metrics. It effectively mitigates issues with false positives and false negatives, offering a more accurate assessment of model performance.

6 Experiments

To validate the effectiveness of our **FLEX** approach, we conducted a comprehensive study comparing the agreement between LLM judgments and human evaluations. We tested various state-of-the-art language models to determine which aligns most closely with human judgment.

6.1 Does FLEX Outperform Other Metrics?

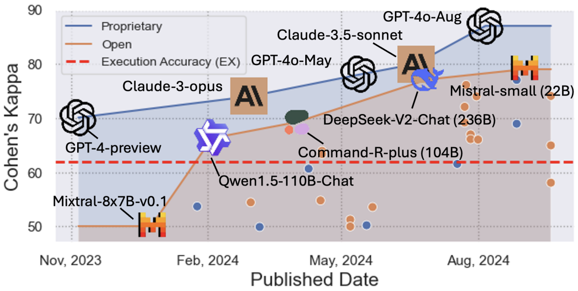


Figure 3: Agreements between human evaluation and **FLEX** across LLM models over time. The red line shows EX metric agreement. Dots represent other LLMs, illustrating lower agreement than previous SOTA. Details are illustrated in Fig. 8.

As shown in Fig. 3, **FLEX** generally outperforms the traditional EX metric across various LLMs, with GPT-4o achieving the highest agreement with human judgments. Open-source models like Mistral-small-Instruct-2409 (Jiang et al., 2023) show substantial improvement over the EX metric, indicating that expert-level evaluations are achievable without reliance on proprietary solutions. LLM performance has improved significantly over time, with more recent models generally demonstrating higher Cohen's Kappa scores. Although open-source models have consistently underperformed proprietary models such as GPT-4o, they are steadily advancing and closing the gap. For instance, DeepSeek-V2-Chat (DeepSeek-AI et al., 2024) shows close performance to Claude-3.5-sonnet (Anthropic, 2024), a proprietary LLM in its contemporary time.

FLEX also offers substantial efficiency gains in both time and cost. GPT-4o evaluated the BIRD dev set (1,534 instances) in under 20 minutes at approximately \$6, whereas volunteers required an average of 2 hours for 200 instances. These results demonstrate **FLEX**'s viability as an efficient (42x faster) and cost-effective alternative to both EX and human evaluation for text-to-SQL tasks. Using Batch API [6] can reduce the input processing cost 75% and more number of threads can reduce the processing time of the **FLEX**.

6.2 What Factors are Beneficial?

Table 4: Comparison of human agreement results depending on execution results.

$\mathbb{C}_{EQ}$		$\mathbb{C}_{NEQ}$					
$\mathbf{R_{gen}}(x)$	$\mathbf{R_{gt}}(x)$	$\mathbf{R_{gen}}(x)$	$\mathbf{R_{gt}}(x)$	**Kappa**	**Acc**	**EQ**	**NEQ**
		✓	✓	**87.04**	**93.5**	**88**	**99**
				82.06	91.0	**88**	94
✓	✓	✓	✓	81.08	90.5	82	**99**

We conducted an ablation study to validate the

---

[6] `https://platform.openai.com/docs/guides/batch`

Table 5: Comparison of human agreement results using different contextual information.

Ablation Settings	Kappa	Acc	EQ	NEQ
w/o Question	80.10	90.0	84	96
w/o Knowledge	79.09	89.5	82	97
w/o Criteria	74.08	87.0	81	93
w/o Ground Truth	29.36	64.0	72	56

efficacy of the optimal context $\mathbb{C}_{FLEX}$ and each contextual information. As shown in Table 4, including execution results only in $\mathbb{C}_{NEQ}$ yields the best performance across all metrics. When execution results are included in both $\mathbb{C}_{EQ}$ and $\mathbb{C}_{NEQ}$, performance is strong in the *Equivalent set* (true positives + false positives) but declines in the *Not equivalent set* (true positives + false negatives). Excluding execution results from both contexts shows the opposite trend. The inclusion of execution results impacts the model's ability to detect errors. Equivalent execution results may hinder false positive detection even if the generated query has incorrect logic. Conversely, for non-equivalent queries, including execution results helps reduce false negatives, as the model can directly compare minor differences in execution results. Therefore, the optimal context selection approach enhances evaluation for $Q_{gen}(x)$, aligning it more closely with human expert assessments.

Table 5 shows the impact of removing different components from the evaluation context. Removing the ground truth query and results is the most significant factor, causing a substantial performance drop across all metrics. This underscores the importance of a reliable reference point in LLM-based text-to-SQL evaluations. Removing the natural language question or external knowledge also decreases notable performance, emphasizing their role in providing a comprehensive evaluation context. The removal of evaluation criteria also impacts performance, highlighting the importance of well-defined guidelines. This ablation study demonstrates that **FLEX**'s performance stems from the synergy of all its components, each playing a vital role in achieving high-quality text-to-SQL evaluation.

7 Leaderboard Re-evaluation

In this section, we evaluated the published result in the Spider and BIRD benchmark using robust capabilities of the **FLEX**. Three key findings were observed to demonstrate crucial limitations of current text-to-SQL evaluation. We first introduce experimental settings and demonstrate three crucial limitations.

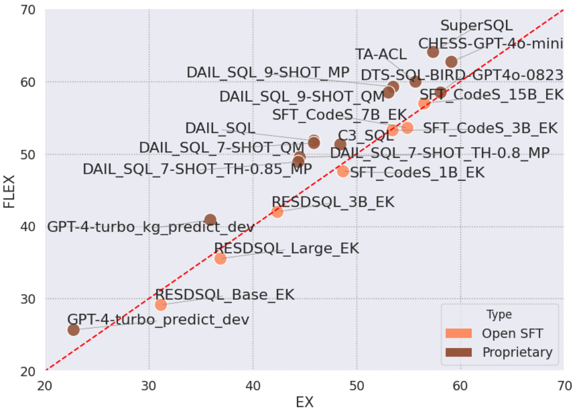


Figure 4: Performance comparison of EX vs. FLEX metrics on BIRD benchmark. The red identity line shows an equivalent score.

7.1 Experiment Setup

For sections 7.2 and 7.4, we conducted experiments using all of the publicly available 50 models on both the Spider and BIRD benchmarks. For section 7.3, we examine error categories on the top ten models in BIRD and Spider benchmarks. Our study utilizes the dev set of the BIRD benchmark because it has not disclosed their test set. For a fair comparison, we employed a dev set of the Spider benchmark; most studies published their prediction results of the dev set. GPT-4o was employed for the query evaluation as an LLM judge, which shows an outperforming agreement in our human evaluation.

7.2 Findings 1: Leaderboard Shifts

The change in the performance of text-to-SQL models in our re-evaluation leads to a significant shift in the leaderboard. We found that the EX metric mostly underestimates the models. As illustrated in Fig. 1 and 4, the average increases from EX to **FLEX** are $+2.63$ and $+2.6$ in Spider and BIRD benchmarks, respectively. This result causes an average rank change of about 1.7 and 2.2 in both benchmarks. Specifically, the maximum rank changes are as large as five positions in BIRD and eight in Spider. These findings demonstrate that the EX metric obscures models' true performance, leading to inaccurate leaderboard rankings.

The extent of these shifts varies across different model types: Proprietary, open pre-trained language models (open PLMs), and open language models with supervised fine-tuning (open SFT). As described in Fig. 5, proprietary models and

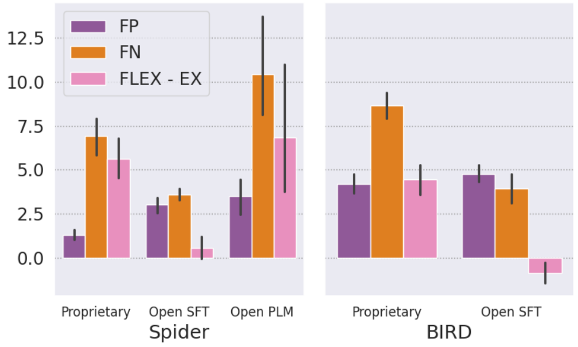


Figure 5: Average model performances and error ratios across different model types.

open PLMs show significant improvements when re-evaluated using the **FLEX**, whereas open SFT models display only marginal changes. A closer examination of the error types explains this difference. Open SFT models have a lower false negative ratio than proprietary and open PLM models. In contrast, all model types show relatively similar false positive ratios.

We hypothesize that this phenomenon can be attributed to differences in the training approaches. The supervised fine-tuning process enables SFT models to predict SQL queries structurally similar to those in the training dataset (Li et al., 2024a). This similarity results in execution results closely matching those of the ground-truth queries, reducing false negatives. In contrast, proprietary and open PLM models are more likely to generate more diverse SQL queries. This diversity increases the likelihood of structural differences between the execution results of the predicted and ground-truth queries. Consequently, these models are more susceptible to false negatives under the EX metric, which **FLEX** can correct.

7.3 Findings 2: Models are Underestimated due to Annotation Quality

We analyze the causes of false negatives, which lead to underestimation of text-to-SQL models' performance, by leveraging a categorized error summary from our framework[7]. As illustrated in Fig. 6, the primary causes of false negatives are different output structures (Struct) and incorrect ground truth (GT). The inherent ambiguity in questions allows for several possible answers, leading to many false negatives due to different output structures (58.13% in BIRD). Incorrect ground truth SQL queries also account for a significant portion of false negatives

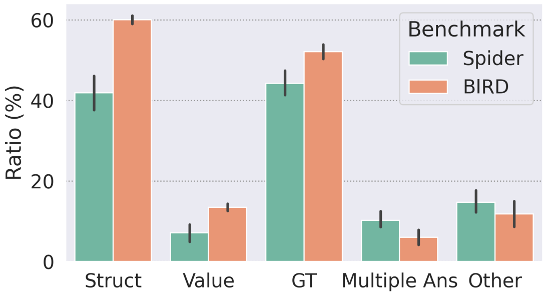


Figure 6: Categorized result of FN ratios in top 10 models. Struct denotes an acceptable output structure variation, Value denotes a different representation of value, GT denotes incorrect ground truth, and Multiple Ans denotes multiple answers available.

(54.54% in BIRD). These results support the observation of previous works (Wang et al., 2023a; Wretblad et al., 2024) and show that this limitation remains unaddressed even in state-of-the-art models.

7.4 Findings 3: Models are Overestimated in Difficult Questions

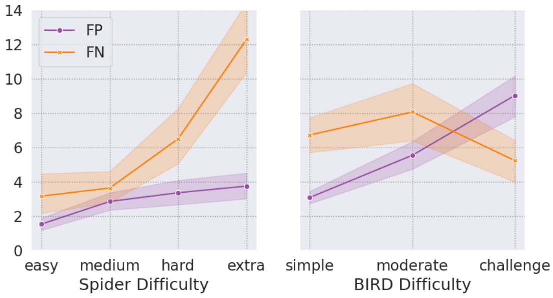


Figure 7: The models' average FP and FN ratios across difficulty levels. The y-axis denotes the percentage (%). In the Spider, text-to-SQL models tend to be underestimated as the questions become more difficult. Conversely, in the BIRD, models are overestimated with more challenging questions.

Our study reveals that text-to-SQL models demonstrate lower **FLEX** score than EX in BIRD's challenging questions while higher in easier ones. This discrepancy can be elucidated through an analysis of false positive and false negative ratios illustrated in Fig. 7. In Spider, considered less complex than BIRD, the false positive ratio is consistently lower than the false negative ratio for text-to-SQL models, indicating that text-to-SQL models show higher **FLEX** than EX. This trend persists for simple and moderate questions in the BIRD. However, in challenging questions of BIRD, the false positive ratio exceeds the false negative ratio, leading to a

---

[7] Detailed process is provided in Appendix B.3.

decrease in the model's performance. These results underscore the significant impact of false positives and false negatives in text-to-SQL evaluation.

8 Discussion

FLEX demonstrates promising results in LLM-based text-to-SQL evaluation, yet it faces limitations primarily due to insufficient contextual information. To enhance the accuracy of LLM-based evaluations, the future text-to-SQL benchmarks should incorporate multiple ground-truth queries per question, more comprehensive schema details, and additional context regarding database relationships and constraints. GPT-4o exhibits potential biases in the human-AI agreement. To address this issue in the future, we suggest implementing advanced prompting techniques such as employing an ensemble of LLM judges. These strategies aim to mitigate individual model biases and enhance LLM-based evaluations' overall reliability and fairness. Continuous fine-tuning of the evaluation process based on human expert feedback is crucial for maintaining and improving the system's performance over time.

FLEX uncovered a substantial performance gap between proprietary models like GPT-4 and open alternatives such as CodeS (Li et al., 2024b) and RESDSQL (Li et al., 2023a). While proprietary models exhibit superior performance, their use in real-world text-to-SQL applications raises concerns, particularly regarding data contamination (Ranaldi et al., 2024). **FLEX** could be employed for diverse and high-quality data augmentation, potentially enhancing open SFT models. This will narrow the performance disparity between open-source and proprietary models and advance the field of text-to-SQL translation.

9 Conclusion

In this paper, we found the limitations in text-to-SQL evaluation, the vulnerability of the EX metric to false positives and negatives, which can result in inaccurate model assessments. Although LLM-based evaluations have emerged as alternatives to human evaluators, they still reveal several limitations. To overcome this, we introduce a novel metric, **FLEX (False-Less EXecution)**. It utilizes the comprehensive context and elaborate criteria based on the advanced language understanding of LLMs to achieve more expert-level evaluations. Our comprehensive experiments show the effectiveness of our proposed metric and provide several empirical findings to promote future studies on text-to-SQL evaluation.

10 Limitations

While our **FLEX** evaluation approach offers significant improvements over traditional metrics like Execution Accuracy (EX), there are some limitations to note: **FLEX** has a strong agreement; Cohen's kappa of 87.04 is imperfect and could be further improved through advanced reasoning (Wang et al., 2022; Madaan et al., 2024) or fine-tuning (Wang et al., 2023b; Zhu et al., 2023) of the LLM evaluator. Our evaluation is currently limited to the Spider and BIRD benchmarks, necessitating further testing on other text-to-SQL datasets and real-world enterprise databases to validate **FLEX**'s effectiveness fully. **FLEX**'s reliance on LLMs makes it computationally intensive and more time-consuming than simple execution-based metrics, which may limit scalability for million-scale evaluations. Moreover, proprietary LLMs pose reproducibility risks, as providers can change or deprecate models without notice. While **FLEX** provides a more comprehensive assessment, it does not eliminate the need for human review, especially in critical applications. Addressing these limitations presents opportunities for future work to refine further and expand LLM-based evaluation approaches for text-to-SQL systems.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2022-0-00680, Abductive inference framework using omni-data for understanding complex causal relations), the National R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (RS-2024-00407282 and RS-2024-00444182), and the Artificial Intelligence Convergence Innovation Human Resources Development Program (IITP-2025-RS-2023-00255968) funded by the Korean government (MSIT).

References

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

Anthropic. 2024. Introducing claude 3.5 sonnet.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, and et al. Luo Ji. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Cohere For AI. 2024. c4ai-command-r-plus-08-2024.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, et al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, and Aobo Yang et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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*.

Gemini Team, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, and Roy Frostig et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Gemma Team, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, and Léonard Hussenot et al. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *Preprint*, arXiv:2401.14196.

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. *Preprint*, arXiv:2406.08426.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego De Las Casas, Emma Bou Hanna, Florian Bressand, and et al. Lengyel. 2024. Mixtral of experts.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. 2023. Mistral 7b (2023). *arXiv preprint arXiv:2310.06825*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing fine-grained evaluation capability in language models. *Preprint*, arXiv:2310.08491.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. In *The Twelfth International Conference on Learning Representations*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. 2024a. The dawn of natural language to sql: Are we fully ready? *Preprint*, arXiv:2406.01265.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. *Preprint*, arXiv:2302.05965.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024b. Codes: Towards building open-source language models for text-to-sql. *Proceedings of the ACM on Management of Data*, 2(3):1–28.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023b. Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13076–13084.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023c. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *arXiv:2305.03111*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and Qian Liu et al. 2023d. Starcoder: may the source be with you! *Preprint*, arXiv:2305.06161.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023e. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, and Igor Babuschkin et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Mohammadreza Pourreza and Davood Rafiei. 2023a. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Mohammadreza Pourreza and Davood Rafiei. 2023b. Evaluating cross-domain text-to-sql models and benchmarks. *arXiv preprint arXiv:2310.18538*.

Mohammadreza Pourreza and Davood Rafiei. 2024. Dts-sql: Decomposed text-to-sql with small large language models. *Preprint*, arXiv:2402.01117.

Ge Qu, Jinyang Li, Bowen Li, Bowen Qin, Nan Huo, Chenhao Ma, and Reynold Cheng. 2024. Before generation, align it! a novel and effective strategy for mitigating hallucinations in text-to-sql generation. *Preprint*, arXiv:2405.15307.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Federico Ranaldi, Elena Sofia Ruzzetti, Dario Onorati, Leonardo Ranaldi, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, and Fabio Massimo Zanzotto. 2024. Investigating the impact of data contamination of large language models in text-to-sql translation. *arXiv preprint arXiv:2402.08100*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and et al. Jérémy Rapin. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Liang Shi, Zhengju Tang, and Zhi Yang. 2024. A survey on employing large language models for text-to-sql tasks. *arXiv preprint arXiv:2407.15186*.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. Chess: Contextual harnessing for efficient sql synthesis. *arXiv preprint*.

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

Bing Wang, Yan Gao, Zhoujun Li, and Jian-Guang Lou. 2023a. Know what i don't know: Handling ambiguous and unknown questions for text-to-sql. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5701–5714.

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*.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. 2023b. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and et al. Joe Davison. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Niklas Wretblad, Fredrik Gordh Riseby, Rahul Biswas, Amin Ahmadi, and Oskar Holmström. 2024. Understanding the effects of noise in text-to-sql: An examination of the bird-bench benchmark. *arXiv preprint arXiv:2402.12243*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, and et al. Haoran Wei. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

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

Fuheng Zhao, Lawrence Lim, Ishtiyaque Ahmad, Divyakant Agrawal, and Amr El Abbadi. 2024. Llm-sql-solver: Can llms determine sql equivalence? *Preprint*, arXiv:2312.10321.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*.

A Experiment Setting

A.1 Frameworks and Hardware

We utilized NVIDIA A6000x4 GPUs with Huggingface transformers (Wolf et al., 2020) and VLLM (Kwon et al., 2023) for generating outputs from open LLMs. To evaluate open models larger than 30B, we utilize the Huggingface Inference API, TogetherAI Inference, and AI/ML API.

A.2 Generation Hyperparameter

We set the temperature to 0 in every experiment to ensure consistent results for each attempt and the maximum number of tokens to 2,048.

A.3 Execution Results in the LLM Prompt

The execution results in the LLM prompt were converted to a markdown table. However, the SQL query can produce extensive rows, increasing the prompt length and API costs. To reduce the overhead, we truncated the middle of the table with a mark as '...' and left 50 heads and tails for more than 100 rows. The shape of the table is appended to the markdown table to help LLM's judgment. Some TEXT-type columns can also be longer. After the 50 characters, we truncated them and appended '... *k chars*', where k indicates an original number of characters.

A.4 Source of Prediction Results

The prediction results were obtained from various sources, including NL2SQL360 (Li et al., 2024a) frameworks, published studies, and reproduced predictions. Table 7 provides an overview of these sources. In DAIL-SQL (Gao et al., 2023), we simplified model results in their repository[8] for readability, as shown in Table 6.

A.5 Human Annotators

Three of the authors are annotators. They have taken database classes at the university and have more than three years of experience operating the database system.

B Additional Experiment Result

B.1 FLEX Human Agreement across Various LLMs

Fig. 8 shows the human-LLM agreement results. We employed the following cutting-edge LLMs to evaluate human-LLM agreements. For proprietary models, we employed OpenAI's GPT models (Achiam et al., 2023; OpenAI et al., 2024), Anthropic's Claude models (Anthropic, 2024), and Google's Gemini models (Gemini Team et al., 2024).

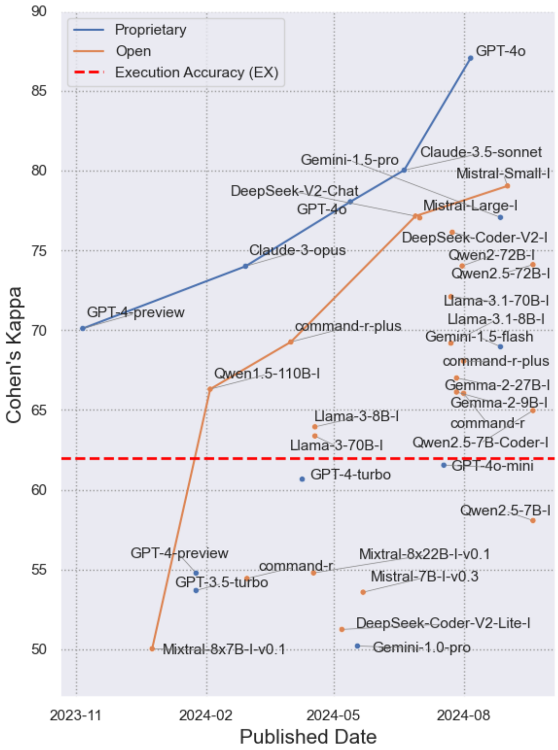


Figure 8: Detailed results of Agreement between Human Evaluation and FLEX across LLM models. The red line indicates the agreement of EX metric. "-I" denotes an instruction-tuned model

For open LMs, we employed Meta's Llama-3 and Llama-3.1 models (Dubey et al., 2024), MistralAI's models (Jiang et al., 2023, 2024), Google's Gemma-2 models (Gemma Team et al., 2024), Qwen-1.5 (Bai et al., 2023), Qwen-2 (Yang et al., 2024) and Qwen-2.5 models (Qwen Team, 2024) DeepSeek AI's DeepSeek-V2 (DeepSeek-AI et al., 2024) and DeepSeek-Coder (Guo et al., 2024), Cohere's Command-R (Cohere For AI, 2024).

B.2 Full FLEX results

The full leaderboard of BIRD-dev and Spider-dev is described in Table 8 and 9, respectively.

B.3 FP and TN Error Analysis

We generate a comprehensive evaluation report after obtaining judgments for all generated queries in a dataset. This report includes an overall accuracy

---

[8] `https://github.com/BeachWang/DAIL-SQL/tree/main/results`

Table 6: Simplified names for DAIL-SQL model results

Simplified Name	**Original File Name**
DAIL_SQL_7-SHOT_TH-0.85_MP	BIRD_WITH_EVIDENCE-TEST_SQL_7-SHOT_EUCDISMASKPRESKLSIMTHR_QA-EXAMPLE_CTX-150_ANS-4096_TH-0.85.txt
DAIL_SQL_7-SHOT_TH-0.8_MP	BIRD_WITH_EVIDENCE-TEST_SQL_7-SHOT_EUCDISMASKPRESKLSIMTHR_QA-EXAMPLE_CTX-150_ANS-4096_TH-0.8.txt
DAIL_SQL_9-SHOT_MP	BIRD_WITH_EVIDENCE-TEST_SQL_9-SHOT_EUCDISMASKPRESKLSIMTHR_QA-EXAMPLE_CTX-150_ANS-4096.txt
DAIL_SQL_9-SHOT_QM	BIRD_WITH_EVIDENCE-TEST_SQL_9-SHOT_EUCDISQUESTIONMASK_QA-EXAMPLE_CTX-150_ANS-4096.txt
DAIL_SQL	QUESTIONMASK_GPT-4_for_bird.txt

Table 7: Overview of prediction sources

Category	**Frameworks / Studies**
NL2SQL360	DIN-SQL (Pourreza and Rafiei, 2023a), C3_SQL (Dong et al., 2023), CodeS (Li et al., 2024b), RESDSQL (Li et al., 2023a), Graphix (Li et al., 2023b), DeepSeek Coder (Guo et al., 2024), Llama-2 (Touvron et al., 2023), Llama-3 (Dubey et al., 2024), Code Llama (Rozière et al., 2024), StarCoder (Li et al., 2023d)
Published	TA-ACL (Qu et al., 2024), DAIL-SQL (Gao et al., 2023), BIRD's baseline (GPT-4) (OpenAI et al., 2024)
Reproduced	DTS-SQL (GPT-4o) (Pourreza and Rafiei, 2024), CHESS (GPT-4o-mini) (Talaei et al., 2024)

Table 8: Spider full result

Rank	Model	FLEX	EX	Δ
1 (-)	SuperSQL	91.20	87.04	+4.16
2 (↑7)	DINSQL	91.10	82.79	+8.32
3 (↑3)	DAILSQL_SC	90.14	83.56	+6.58
4 (↑4)	DAILSQL	88.88	83.08	+5.80
5 (↓2)	TA-ACL	88.78	85.01	+3.77
6 (↓4)	SFT_CodeS_7B	87.91	85.40	+2.51
7 (↓3)	SFT_CodeS_15B	87.33	84.91	+2.42
8 (↑2)	C3_SQL	87.04	82.01	+5.03
9 (↑5)	SFT_Deepseek_Coder_7B	84.72	80.75	+3.97
10 (↓2)	SFT_CodeS_3B	84.72	83.27	+1.45
11 (↓6)	RESDSQL_NatSQL_3B	83.66	84.14	−0.48
12 (-)	RESDSQL_3B	82.01	81.82	+0.19
13 (-)	Graphix_PICARD_3B	81.72	80.95	+0.77
14 (↓4)	resdsql_text2natsql_large	81.43	82.01	−0.58
15 (↑1)	resdsql_text2sql_large	79.11	80.08	−0.97
16 (↓1)	resdsql_text2natsql_base	78.82	80.17	−1.35
17 (-)	SFT_CodeS_1B	78.63	77.95	+0.68
18 (↑6)	pretrained_deepseek_coder_7b	78.34	64.22	+14.12
19 (-)	SFT_Llama3_8B	78.05	76.11	+1.93
20 (↓3)	resdsql_text2sql_base	76.98	77.95	−0.97
21 (↓1)	SFT_CodeLlama_7B	76.21	74.08	+2.13
22 (↓1)	Deepseek-Coder-7B	75.15	73.50	+1.64
23 (↓1)	SFT_StarCoder_7B	72.44	72.05	+0.39
24 (↑3)	pretrained_llama3_8b	68.86	60.44	+8.41
25 (↓2)	SFT_Llama2_7B	64.41	65.28	−0.87
26 (↓1)	CodeLlama-7B	60.35	60.93	−0.58
27 (↑1)	pretrained_starcoder_7b	59.96	55.51	+4.45
28 (↓2)	Llama2-7B	59.67	60.83	−1.16
29 (-)	pretrained_codellama_7b	55.71	51.64	+4.06
30 (-)	pretrained_llama2_7b	24.08	20.99	+3.09

Table 9: BIRD full result

Rank	Model	FLEX	EX	Δ
1 (↑2)	SuperSQL	64.08	57.37	+6.71
2 (↓1)	CHESS-GPT-4o-mini	62.71	59.13	+3.59
3 (↑2)	TA-ACL	59.97	55.67	+4.30
4 (↑3)	DAIL_SQL_9-SHOT_MP	59.26	53.52	+5.74
5 (↑4)	DAIL_SQL_9-SHOT_QM	58.47	53.06	+5.41
5 (↓3)	DTS-SQL-BIRD-GPT4o-0823	58.47	58.08	+0.39
7 (↓3)	SFT_CodeS_15B_EK	56.98	56.52	+0.46
8 (↓2)	SFT_CodeS_7B_EK	53.59	54.89	−1.30
9 (↓1)	SFT_CodeS_3B_EK	53.26	53.46	−0.20
10 (↑2)	DAIL_SQL	51.83	45.89	+5.93
11 (↑1)	DAIL_SQL_7-SHOT_QM	51.50	45.89	+5.61
12 (↓1)	C3_SQL	51.30	48.44	+2.87
13 (↑1)	DAIL_SQL_7-SHOT_TH-0.8_MP	49.54	44.52	+5.02
14 (↑1)	DAIL_SQL_7-SHOT_TH-0.85_MP	48.89	44.39	+4.50
15 (↓5)	SFT_CodeS_1B_EK	47.59	48.70	−1.11
16 (-)	RESDSQL_3B_EK	41.98	42.37	−0.39
17 (↑1)	GPT-4-turbo_kg_predict_dev	40.87	35.92	+4.95
18 (↓1)	RESDSQL_Large_EK	35.53	36.90	−1.37
19 (-)	RESDSQL_Base_EK	29.14	31.16	−2.02
20 (-)	GPT-4-turbo_predict_dev	25.68	22.75	+2.93

score, a breakdown of error types, and detailed explanations for each incorrect query. The LLM is prompted to categorize the error case based on its judgment and other contextual information. Judgments are categorized according to the criteria described in the previous subsection, and complex errors can belong to more than one category. Researchers can utilize this evaluation report to identify systematic errors in their text-to-SQL models and guide future improvements. Fig. 12 describes the system prompt to categorize false positives, and Fig. 13 describes the system prompt to categorize false negatives. Fig. 9 demonstrates the error categorization results of the top 10 BIRD-dev and Spider-dev models.

C EX Error Case Study

We sampled the representative examples from our human study. Table 10, 11, 12 and 13 demonstrate the false positive examples. Table 14, 15, 16, 17 demonstrate the false negative examples. Table 18 demonstrates the example, including different column ordering and incorrect ground truth.

D Artifacts

We confirmed that all models, datasets, and frameworks are allowed for research use.

E AI Assitant

We use Microsoft Copilot as a coding assistant and Grammarly and Writefull as a writing assistant.

F LLM Prompt

F.1 FLEX

We distinguished the prompt into system prompt and user prompt. As described in Fig. 14, the user prompt includes natural language questions, generated queries, ground truth queries, and execution results. The system prompt includes task instruction and criteria as described in Fig. 10 and 11. In the categorization, the user prompt includes natural language questions, generated queries, and ground truth queries. All system prompts instruct the LLM to output using Chain-of-Thought (Wei et al., 2022) prompting followed by JSON formatted result to extract judgment from LLM output.

F.2 Prometheus-2

Prometheus-2 (Kim et al., 2024) requires instruction, response, reference answer, criteria, and score rubrics for evaluation. Fig. 15 and 16 show optimal criteria for equivalent and not equivalent sets, respectively. We deemed scores of four and five as correct to achieve the highest level of agreement.

Table 10: Equivalent set #3 (BIRD #9): An incorrect schema linking case demonstrating a generated query that mistakenly uses ’schools.FundingType’ instead of ’T2.Charter Funding Type’. The human judgment accurately identifies this schema linking error, further corroborated by the detailed FLEX w/ GPT-4o analysis.

Question	Among the schools with the average score in Math over 560 in the SAT test, how many schools are directly charter-funded?

Ground Truth Query	SELECT COUNT(T2.School Code) FROM satscores AS T1 INNER JOIN frpm AS T2 ON T1.cds = T2.CDSCode
	WHERE T1.AvgScrMath 560 AND T2.Charter Funding Type = `Directly funded’
	> 2

Generated Query	SELECT COUNT(*) FROM satscores INNER JOIN schools ON satscores.cds = schools.CDSCode
	WHERE satscores.AvgScrMath 560 AND schools.FundingType = `Directly funded’;
	> 2

Human Judgment	Incorrect Schema Linking: The generated query uses incorrect columns schools.funding type The generated query is incorrect

FLEX *w/ GPT-4o*
(Correct)
	The Prediction Query and the Ground Truth Query both aim to count the number of schools that meet certain criteria.
	However, they differ in the tables and columns they use to achieve this.
	1. Tables and Joins:
	- The Prediction Query uses the satscores and schools tables, joining them on satscores.cds = schools.CDSCode.
	- The Ground Truth Query uses the satscores and frpm tables, joining them on T1.cds = T2.CDSCode.
	2. Conditions:
	- Both queries filter for schools with an average Math score greater than 560.
	- The Prediction Query checks if schools.FundingType is ’Directly funded’.
	- The Ground Truth Query checks if frpm.Charter Funding Type is ’Directly funded’.
	3. Schema Considerations:
	- The hint indicates that schools.FundingType and frpm.Charter Funding Type are not equivalent,
	which means they may not represent the same data or criteria.
	- The Ground Truth Query uses frpm.Charter Funding Type, which is the correct column
	to check for charter funding type according to the question.
	4. Logical Correctness:
	- The Prediction Query uses a different column (schools.FundingType) to determine if a school is directly charter-funded,
	which is not equivalent to frpm.Charter Funding Type.
	This could lead to incorrect results if the data in these columns differ.
	
	Given these points, the Prediction Query does not logically answer the question as it uses an incorrect column
	to determine the charter funding type. Therefore, the Prediction Query is incorrect.
	```json
	{"correct": false}
	```

Table 11: Equivalent set #28 (BIRD #68): An incorrect condition filtering case demonstrating a generated query that erroneously adds ’StatusType = ’Closed” and uses ’SOC = ’11” as a string instead of an integer. The human judgment correctly identifies the missing ’StatusType = ’Closed” condition, while FLEX w/ GPT-4o provides a more comprehensive analysis, highlighting both the SOC value type discrepancy and the missing StatusType condition. This example emphasizes the critical nature of precise condition filtering in SQL queries.

Question	Which county reported the most number of school closure in the 1980s with school wonership code belonging to Youth Authority Facilities (CEA)?
Ground Truth Query	SELECT schools.County FROM schools WHERE schools.SOC = `11’ AND strftime(`%Y’, schools.ClosedDate) BETWEEN `1980’ AND ’1989’ GROUP BY schools.County ORDER BY COUNT(schools.County) DESC LIMIT 1; > Sacramento
Generated Query	SELECT County FROM schools WHERE strftime(`%Y’, ClosedDate) BETWEEN `1980’ AND `1989’ AND StatusType = `Closed’ AND SOC = 11 GROUP BY County ORDER BY COUNT(School) DESC LIMIT 1 > Sacramento
Human Judgment	Incorrect Condition Filtering: The generated query misses a filtering condition StatusType = `Closed’ The generated query is incorrect

FLEX *w/ GPT-4o*
(Correct)
	```json
	{"correct": false}
	```
	The Prediction Query is incorrect for the following reasons:
	1. SOC Value Type: The Prediction Query uses SOC = ’11’ with ’11’ as a string, while the Ground Truth Query
	uses SOC = 11 as an integer. This could lead to different results if the SOC column is of integer type.
	2. StatusType Condition: The Prediction Query does not include the condition StatusType = ’Closed’,
	which is present in the Ground Truth Query. This condition is crucial to ensure that only closed schools are considered,
	which directly affects the result.
	These differences mean that the Prediction Query could potentially include incorrect data, leading to different results
	if the data set includes schools that are not closed or if the SOC column is of integer type.

Table 12: Equivalent set #66 (BIRD #45): This case showcases a generated query that unnecessarily includes a GROUP BY clause. Human judgment correctly identifies this issue, noting that it can yield incorrect results if there are multiple schools with the same name. FLEX w/ GPT-4o incorrectly assesses both queries as equivalent, overlooking the potential problems caused by the superfluous GROUP BY clause. This example highlights the importance of careful query structure analysis and the challenges in detecting subtle logical errors in SQL query generation and evaluation.

Question	What is the average writing score of each of the schools managed by Ricci Ulrich? List the schools and the corresponding average writing scores.

Ground Truth Query	SELECT T2.School, T1.AvgScrWrite FROM satscores AS T1
	INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode
	WHERE T2.AdmFName1 = `Ricci’ AND T2.AdmLName1 = `Ulrich’
	> | Buchanan High | 507.0 |

Generated Query	SELECT T1.sname, AVG(T1.avgscrwrite) FROM satscores AS T1
	INNER JOIN schools AS T2 ON T1.cds = T2.cdscode
	WHERE T2.admfname1 = `Ricci’ AND T2.admlname1 = `Ulrich’ GROUP BY T1.sname
	> | Buchanan High | 507.0 |

Human Judgment	The generated query misses a GROUP BY clause. It can yield different results If there are several schools with the same name. The generated query is incorrect.

FLEX *w/ GPT-4o*
(Incorrect)
	To evaluate the correctness of the Prediction Query, let’s analyze both the Prediction Query
	and the Ground Truth Query in the context of the provided schema and question.
	Objective: The question asks for the average writing score of each school managed by Ricci Ulrich,
	listing the schools and their corresponding average writing scores.
	Schema Understanding:
	- satscores table contains cds and avgscrwrite.
	- schools table contains cdscode, sname, admfname1, and admlname1.
	Ground Truth Query Analysis:
	- It selects School and AvgScrWrite from the satscores and schools tables.
	- It joins these tables on T1.cds = T2.CDSCode.
	- It filters for schools managed by Ricci Ulrich using T2.AdmFName1 = ’Ricci’ AND T2.AdmLName1 = ’Ulrich’.
	Prediction Query Analysis:
	- It selects T1.sname and the average of T1.avgscrwrite.
	- It joins the satscores and schools tables on T1.cds = T2.cdscode.
	- It filters for schools managed by Ricci Ulrich using T2.admfname1 = ’Ricci’ and T2.admlname1 = ’Ulrich’.
	- It groups the results by T1.sname to calculate the average writing score for each school.
	Comparison:
	- The Prediction Query correctly joins the satscores and schools tables and filters for the correct administrator.
	- The use of AVG(T1.avgscrwrite) and GROUP BY T1.sname in the Prediction Query is appropriate
	to calculate the average writing score for each school.
	- The Ground Truth Query does not perform an aggregation,
	which suggests it assumes precomputed average scores are available in T1.AvgScrWrite.
	Result Consistency:
	- Both queries produce the same results, indicating that the Prediction Query correctly calculates the average writing scores.
	Given the analysis, the Prediction Query logically answers the question and produces the correct results.
	Therefore, the Prediction Query is correct.
	```json
	{"correct": true}
	```

Table 13: Equivalent set #73 (BIRD #64): A case highlighting the importance of handling nullable columns in SQL queries. The human correctly identifies that COUNT(*) and COUNT(charternum) treat null values differently, potentially affecting the result if charternum is nullable. FLEX w/ GPT-4o incorrectly deems the queries equivalent, overlooking this crucial distinction. This example reveals a limitation in the AI system's query analysis, emphasizing the need for careful consideration of null value handling in SQL evaluation.

Question	What is the total number of non-chartered schools in the county of Los Angeles with a percent (%) of eligible free meals for grades 1 through 12 that is less than 0.18%?
Ground Truth Query	SELECT COUNT(charternum) FROM schools WHERE mailcity = `Hickman’ AND charternum = `00D4’ > 3
Generated Query	SELECT COUNT(*) FROM schools WHERE CharterNum = `00D4’ AND MailCity = `Hickman’ > 3
Human Judgment	Missing handling of the nullable column: charternum is the nullable column in the schema COUNT(*) counts the null values while COUNT(charternum) does not. The generated query is incorrect.

FLEX *w/ GPT-4o*
(Incorrect)
	The Prediction Query and the Ground Truth Query are essentially the same in terms of logic and execution.
	Both queries aim to count the number of schools in the 'schools' table where the 'MailCity' is 'Hickman' and the 'CharterNum' is '00D4'.
	Key points to consider:
	- Both queries use the same filtering conditions: mailcity = 'Hickman' and charternum = '00D4'.
	- The difference in column case sensitivity (charternum vs CharterNum and mailcity vs MailCity) is not an issue in SQLite,
	as it is case-insensitive with respect to column names.
	- The use of COUNT(charternum) in the Prediction Query versus COUNT(*) in the Ground Truth Query does not affect the result
	as long as charternum is not nullable. If charternum were nullable, it could potentially affect the count, but the question implies that
	it is not an issue here since both queries produce the same result.
	Given these considerations, the Prediction Query correctly answers the question.
	```json
	{"correct": true}
	```

Table 14: Not equivalent set #90 (BIRD #63): A case showcases FLEX's ability to evaluate SQL queries with different column selections correctly. The Generated Query includes extra columns for additional administrator names (admname2, admname3) that are not present in the Ground Truth Query. Despite this difference, FLEX accurately judges that both queries are logically equivalent in answering the question about administrators in chartered schools with charter number '0002'. This demonstrates FLEX's sophisticated understanding that the presence of additional, non-conflicting information does not alter the fundamental correctness of a query in meeting the specified requirements.

Question	In chartered schools with charter number 00D2, what are the names of all the administrators? Include the name of the school and the city to which it belongs

Ground Truth Query	SELECT AdmFName1, AdmLName1, School, City FROM schools
	WHERE Charter = 1 AND CharterNum = `00D2’

Generated Query	SELECT school, city, admfname1, admlname1, admfname2, admlname2, admfname3, admlname3 FROM schools
	WHERE charter = 1 AND charternum = `00D2’

Human Judgment	Extra Column: admfname2, admlname2, admfname3, admlname3, but the other logic is correct. The generated query is correct.

FLEX *w/ GPT-4o*
(Correct)
	The task is to find the names of all administrators in chartered schools with charter number '00D2',
	including the name of the school and the city to which it belongs.
	Let's compare the Prediction Query and the Ground Truth Query:
	Filtering Condition:
	- Both queries correctly filter the schools where Charter = 1 and CharterNum = '00D2'.
	- This part of the query is consistent with the requirement to focus on chartered schools with the specified charter number.
	Selected Columns:
	- The Ground Truth Query selects AdmFName1, AdmLName1, School, and City.
	- This directly answers the question by providing the first administrator's first and last name, along with the school and city.
	- The Prediction Query selects school, city, admfname1, admlname1, admfname2, admlname2, admfname3, admlname3.
	- This includes additional columns for the second and third administrators' names, which are not required by the question.
	However, this does not affect the logical correctness of the query as it still provides the necessary information.
	Output Structure:
	- The Ground Truth Result has four columns,
	while the Prediction Result has eight columns due to the inclusion of additional administrator name fields.
	- The extra columns in the Prediction Result do not impact the ability to answer the question, as the required information is still present.
	Result Comparison:
	- The results from both queries contain the same schools and cities, and the first administrator's names match.
	- The additional columns in the Prediction Result are filled with None for the second and third administrators,
	which does not affect the correctness of the answer to the question.
	- In conclusion, the Prediction Query logically answers the question by providing the necessary information
	about the first administrator's name, school, and city, even though it includes additional columns.
	- The extra columns do not introduce any errors or inconsistencies in the context of the question.
	```json
	{"correct": true}
	```

Table 15: Not equivalent set #17 (BIRD #37): A case where FLEX correctly judges two SQL queries as logically equivalent despite a difference in column order. The Generated Query selects columns as T2.Street, T2.City, T2.State, T2.Zip, while the Ground Truth Query selects them as Street, City, State, Zip. Despite this variation in the SELECT clause, both queries aim to find the complete address of the school with the lowest excellence rate. FLEX accurately determines that the column order difference does not affect the logical correctness of the query in answering the question, demonstrating its ability to evaluate query semantics beyond superficial syntax.

Question	What is the complete address of the school with the lowest excellence rate? Indicate the Street, City, Zip and State.

Ground Truth Query	SELECT schools.Street, schools.City, schools.Zip, schools.State FROM schools
	INNER JOIN satscores ON schools.CDSCode = satscores.cds
	WHERE satscores.NumTstTakr != 0
	ORDER BY (satscores.NumGE1500 * 1.0 / satscores.NumTstTakr) ASC LIMIT 1;
	> | 1111 Van Ness Avenue | Fresno | CA | 93721-2002 |

Generated Query	SELECT T2.Street, T2.City, T2.State, T2.Zip FROM satscores AS T1
	INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode
	ORDER BY CAST(T1.NumGE1500 AS REAL) / T1.NumTstTakr ASC LIMIT 1
	> | 1111 Van Ness Avenue | Fresno | 93721-2002 | CA |

Human Judgment	Only the column order is different, so the generated query is correct.

FLEX *w/ GPT-4o*
(Correct)
	The task is to find the complete address of the school with the lowest excellence rate,
	defined as NumGE1500 / NumTstTakr. The complete address should include the Street, City, State, and Zip code.
	
	Let's compare the Prediction Query and the Ground Truth Query:
	
	Join Condition:
	- Both queries correctly join the schools and satscores tables using the CDSCode and cds columns, respectively.
	
	Excellence Rate Calculation:
	- Both queries calculate the excellence rate as NumGE1500 / NumTstTakr.
	- The Ground Truth Query explicitly casts NumGE1500 as REAL to ensure floating-point division,
	while the Prediction Query multiplies NumGE1500 by 1.0 to achieve the same effect. -
	Both methods are valid for ensuring floating-point division in SQLite.
	
	Filtering:
	- The Prediction Query includes a condition satscores.NumTstTakr != 0 to avoid division by zero,
	which is a good practice.
	- The Ground Truth Query does not explicitly include this condition,
	but it is implied that NumTstTakr should not be zero for a valid excellence rate calculation.
	
	Ordering and Limiting:
	- Both queries order the results by the calculated excellence rate in ascending order and limit the result to the first entry,
	which corresponds to the school with the lowest excellence rate.
	
	Selected Columns:
	- The Prediction Query selects Street, City, Zip, State, while the Ground Truth Query selects Street, City, State, Zip.
	The order of columns in the SELECT clause does not affect the correctness of the query in terms of answering the question.
	
	Result Discrepancy:
	- The Prediction Result and Ground Truth Result differ in the order of the State and Zip columns.
	However, this does not affect the logical correctness of the query in terms of providing the complete address.
	- In conclusion, the Prediction Query logically answers the question by providing the complete address of the school with the lowest excellence rate.
	- The difference in column order in the result does not impact the correctness of the query.
	```json
	{"correct": true}
	```

Table 16: Not equivalent set #16 (BIRD #32): A different output value case demonstrates the accuracy of FLEX judgment in SQL query evaluation. While the Generated Query used the ROUND function to present results as percentages, the Ground Truth Query expressed them as decimal values. Despite this difference in output format, FLEX correctly determined that both queries produce logically equivalent results. This showcases FLEX's ability to understand and evaluate query semantics and functionality beyond simple text comparison.

<table>
<tr><td>Question</td><td>What is the eligible free or reduced price meal rate for the top 5 schools in grades 1-12
with the highest free or reduced price meal count of the schools with the ownership code 66?</td></tr>
<tr><td>Ground Truth Query</td><td>SELECT CAST(T1.FRPM Count (K-12) AS REAL) / T1.Enrollment (K-12) FROM frpm AS T1
INNER JOIN schools AS T2 ON T1.CDSCode = T2.CDSCode WHERE T2.SOC = 66
ORDER BY T1.FRPM Count (K-12) DESC LIMIT 5
> | 0.9993 | 0.9179 | 0.9142 | 0.8965 | 0.8958 |</td></tr>
<tr><td>Generated Query</td><td>SELECT ROUND((frpm.FRPM Count (K-12) / frpm.Enrollment (K-12)) * 100, 2) AS eligible_rate FROM frpm
INNER JOIN schools ON frpm.CDSCode = schools.CDSCode WHERE schools.SOC = `66’
ORDER BY frpm.FRPM Count (K-12) DESC LIMIT 5
> | 99.93 | 91.79 | 91.42 | 89.65 | 89.58 |</td></tr>
<tr><td>Human Judgment</td><td>Different output value representation: The generated query multiples 100 to the result for readability.
Other logic is correct; the generated query is correct</td></tr>
<tr><td><b>FLEX</b> <i>w/ GPT-4o</i>
<b>(Correct)</b></td><td>The Prediction Query and the Ground Truth Query both aim to calculate the eligible free or reduced price meal rate for the top 5 schools with the highest free or reduced price meal count,
specifically for schools with the ownership code 66. However, there are some differences in their execution and results:

Calculation of Eligible Rate:
- The Ground Truth Query calculates the eligible rate as a floating-point number by dividing FRPM Count (K-12) by Enrollment (K-12).
- The Prediction Query also performs this division but multiplies the result by 100 and rounds it to two decimal places,
effectively converting the rate into a percentage.

Result Representation:
- The Ground Truth Result presents the eligible rates as decimal values (e.g., 0.8964987714987716).
- The Prediction Result presents the eligible rates as percentages (e.g., 89.65).

Logical Correctness:
- Both queries logically answer the question by selecting the top 5 schools based on the FRPM Count (K-12) and
calculating the eligible rate for these schools.
- The difference in representation (percentage vs. decimal) does not affect the logical correctness of the Prediction Query.
- The conversion to percentage is a valid transformation of the rate.

SQLite3 Compatibility:
- Both queries are compatible with SQLite3, and the operations performed are valid within this context.
- In conclusion, the Prediction Query is correct in terms of logic and functionality.
- The difference in result representation (percentage vs. decimal) is acceptable and does not impact the query's ability to answer the question.</td></tr>
</table>

Table 17: Not equivalent set #26 (BIRD #53): This case illustrates FLEX’s capability to correctly evaluate queries with different output structures but equivalent logical intent. The Generated Query uses SUM to aggregate the total number of test takers in Fresno schools, while the Ground Truth Query lists individual values for each school. Despite this structural difference, FLEX accurately judges that the Generated Query correctly answers the question, "How many test takers are at the schools whose mailing city address is in Fresno?". This demonstrates FLEX’s understanding that aggregation can be an appropriate method to answer quantitative questions, even when the reference query provides more granular data.

Question	How many test takers are there at the school/s whose mailing city address is in Fresno?
Ground Truth Query	SELECT T1.NumTstTakr FROM satscores AS T1 INNER JOIN schools AS T2 ON T1.cds = T2.CDSCode WHERE T2.MailCity = `Fresno’ > | 5 | 8 | 368 | .... (Sum is 6070)
Generated Query	SELECT SUM(satscores.NumTstTakr) FROM schools INNER JOIN satscores ON schools.CDSCode = satscores.cds WHERE schools.MailCity = `Fresno’; 6070 > 6070
Human Judgment	Different output structure: Two queries have the same JOIN and WHERE structure, but the generated query adds all values while the ground truth does not. The question does not specify it, so the generated query is correct.

FLEX *w/ GPT-4o*
(Correct)
	The task is to determine the total number of test takers at schools whose mailing city address is in Fresno.
	
	Let’s compare the two queries:
	
	Prediction Query:
	- This query calculates the sum of the NumTstTakr column from the satscores table for all schools in Fresno.
	- It uses an INNER JOIN to match records from the schools and satscores tables based on the CDSCode and cds columns, respectively.
	- The result is a single aggregated value representing the total number of test takers in Fresno.
	
	Ground Truth Query:
	- This query retrieves individual NumTstTakr values for each school in Fresno without aggregation.
	- It also uses an INNER JOIN to match records from the schools and satscores tables.
	- The result is a list of NumTstTakr values for each school in Fresno.
	
	Analysis:
	- The Prediction Query provides the total number of test takers by summing up the NumTstTakr values,
	which directly answers the question of "how many test takers are there" in total.
	- The Ground Truth Query lists the number of test takers for each school individually,
	which does not directly answer the question of the total number of test takers.
	
	Conclusion:
	- The Prediction Query correctly answers the question by providing the total number of test takers in Fresno,
	while the Ground Truth Query provides a breakdown by school.
	The difference in results is due to the aggregation in the Prediction Query,
	which is appropriate for the question asked.
	```json
	{"correct": true}
	```

Table 18: Not equivalent set #98 (BIRD #80): A case highlighting the Ground Truth Query's incorrectness due to overlooking multiple rows satisfying the condition. The Ground Truth Query fails to account for potential ties in maximum latitude by using LIMIT 1. Despite this flaw, FLEX w/ GPT-4o incorrectly judges it as correct while correctly identifying the Generated Query's handling of ties. This reveals a limitation in the AI system's ability to recognize errors in Ground Truth Queries, particularly when dealing with maximum values and potential multiple matching rows in SQL evaluation..

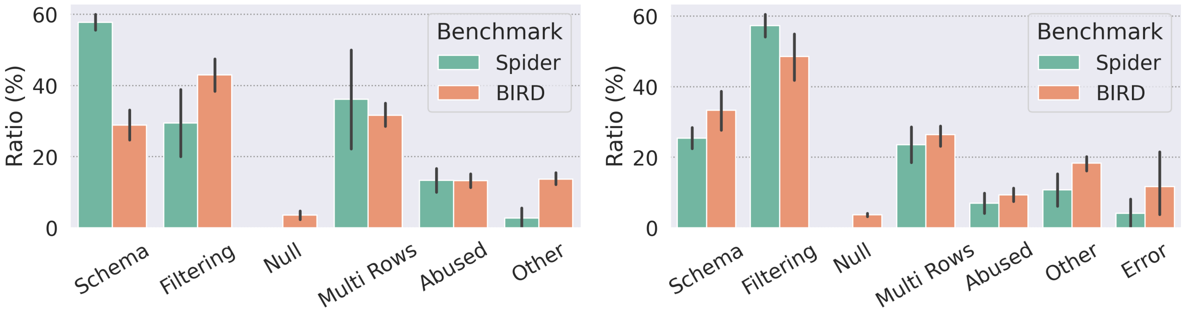
<table>
<tr><td>Question</td><td>What is the school type of the school with the highest latitude?
Indicate the name of the school as well as the latitude coordinates.</td></tr>
<tr><td>Ground Truth Query</td><td>SELECT T1.School Type, T1.School Name, T2.Latitude FROM frpm AS T1
INNER JOIN schools AS T2 ON T1.CDSCode = T2.CDSCode
ORDER BY T2.Latitude DESC LIMIT 1
> | High Schools (Public) | Leggett Valley High | 44.219305 |</td></tr>
<tr><td>Generated Query</td><td>SELECT schools.school, schools.latitude, frpm.school type FROM schools
JOIN frpm ON schools.cdscode = frpm.cdscode
WHERE schools.latitude = (SELECT MAX(latitude) FROM schools)
| Leggett Valley High | 44.219305 | High Schools (Public) |
| Leggett Valley Elementary | 44.219305 | Elementary Schools (Public) |</td></tr>
<tr><td>Human Judgment</td><td>The ground truth does not consider that multiple rows have the same highest latitude.
Moreover, output column order is different, but the generated query is correct.</td></tr>
<tr><td><b>FLEX</b> <i>w/ GPT-4o</i>
<b>(Incorrect)</b></td><td>The task is to find the school type, name, and latitude of the school with the highest latitude.
Let's compare the Prediction Query and the Ground Truth Query to understand their differences and determine if the Prediction Query is correct.

Prediction Query Analysis
- This query joins the schools and frpm tables on the cdscode column.
- It selects the school name, latitude, and school type for schools with the maximum latitude.
- The subquery (SELECT MAX(latitude) FROM schools) correctly identifies the maximum latitude.
- The WHERE clause filters schools to only those with this maximum latitude.
- The result includes multiple rows if there are ties in the maximum latitude.

Ground Truth Query Analysis
- This query also joins the frpm and schools tables on the CDSCode column.
- It orders the results by latitude in descending order and limits the output to the top row,
- - - effectively selecting the school with the highest latitude.
- It selects the school type, school name, and latitude.

Comparison
- Result Set: The Prediction Query returns two rows because there are two schools with the same maximum latitude,
while the Ground Truth Query returns only one row due to the LIMIT 1 clause.
- Columns: Both queries select the same logical information: school type, school name, and latitude, though the order of columns differs.
- Handling Ties: The Prediction Query correctly handles ties by returning all schools with the maximum latitude,
which is logically consistent with the question's requirement to identify the school with the highest latitude.

Conclusion - The Prediction Query logically answers the question by identifying all schools with the maximum latitude,
which is a valid interpretation of the task.
- The Ground Truth Query limits the result to one school, which is another valid approach.
- The difference in the number of rows is due to the handling of ties, which is acceptable given the question's ambiguity.
```json
{"correct": true}
```</td></tr>
</table>



(a) False positives

(b) True negatives

Figure 9: Average error categorization result of top 10 Spider-dev and BIRD-dev models using **FLEX**. The gray line denotes the standard deviation. Schema denotes incorrect schema alignment, Filtering denotes incorrect filtering conditions, Null denotes incorrect handling of nullable columns, Multi Rows denotes incorrect accounting of multiple rows, Abused denotes abused clauses, and Error denotes SQL compile error.

~~~
The Prediction Result matches the Reference Result. However, this does not guarantee that the Prediction Query is correct. Carefully analyze the Prediction Query and evaluate its correctness considering the following criteria:

Correct Prediction Query
- If the Prediction Query missed some tables or columns, it is acceptable if the missing information does not affect the query's ability to answer the Question.
- missing DISTINCT in SELECT is always allowed because we do not consider duplicate rows.

Incorrect Prediction Query
- Does not logically answer the Question or contains significant errors.
- Produces different results due to incorrect filering or missing conditions, JOIN redundancy, or other fatal issues.
- Fails to handle null values, multiple rows, or other critical aspects of the Question.
- Does not consider nullable columns in aggregation functions (SUM, COUNT, AVG) and NULL value can lead to unexpected results.
For example, COUNT(*) in the prediction but COUNT(school) in the ground truth will produce different results if the column school does not have a NOT NULL constraint in the schema.
- Does not produce correct results when multiple rows satisfy the condition (e.g min/max, multiple transactions in a day).
- Abused clauses (LIMIT, GROUP BY) to limit the results when the user didn't request it.

SQLite3 Compatibility
- Both queries are SQLite3 compatible, meaning integer division is not automatically converted to float division, and logical operators, column/table names are case-insensitive.
- comparison between string and integer is allowed in SQLite3.

Analysis Guidelines
1. Compare the Prediction Query with the Ground Truth Query within the context of the provided schema and question.
2. Predict the query's logical correctness based on the criteria mentioned above.

Finally, score the Prediction Query as follows:
```json
{"correct": true or false}
```
~~~

Figure 10: **FLEX** system prompt to judge the generated query producing the same results as ground truth.

~~~
The Prediction Result differs from the Ground Truth Result. However, this does not necessarily mean that the Prediction Query is incorrect. Analyze the differences between the Prediction Query and Reference Query, considering the following:

Correct
- The Prediction Query logically answers the Question, even if the output structure differs from the Ground Truth Query.
- Do not consider column naming, column/row ordering.
- Some extra column or missing column in the output structure is acceptable if it does not affect the query's ability to answer the Question.
- Differences in the representation of values, such as formatting (percentile, YES/NO) or data types, are acceptable if they do not affect the query's logical correctness.
- Ambiguous questions may have multiple correct answers, so the Prediction Query may differ from the Ground Truth Query.
- Multiple rows are acceptable when the calculate the min/max

Incorrect
- The Prediction Query does not logically answer the Question or contains significant errors.
- The Prediction Query produces different results due to incorrect filering or missing conditions, JOIN redundancy, or other fatal issues.
- The Prediction Query fails to handle null values, multiple rows, or other critical aspects of the Question.
- The result of the Prediction Query is significantly different from the Ground Truth Query even its structure is similar.

SQLite3 Compatibility
- Both queries are SQLite3 compatible, meaning integer division is not automatically converted to float division, and logical operators are case-insensitive.
- If the table schema and description are different, follow the schema provided in the prompt.

Provide a detailed comparison of the Prediction Query and Ground Truth Query, focusing on the nature and significance of their differences. If the Prediction Query is incorrect, explain the specific errors and how they affect the query's ability to answer the Question.

Finally, score the Prediction Query as follows:
```json
{”correct”: true/false}
```
~~~

Figure 11: **FLEX** system prompt to judge the generated query producing different results as ground truth.

~~~
You are an expert SQL evaluator responsible for assessing the correctness and quality of SQL queries.
Analyze the judgment thoroughly and provide a categorized evaluation based on the following criteria:
Two queries have no compile-time errors, which means they are both valid SQL queries, integer division is not automatically converted to float division, and logical operators are case-insensitive.

### Criteria
1. Incorrect Schema Linking: Utilized tables and columns in the predicted query do not align with the question and the provided schema. Different columns are permissible if they are described as equivalent in the table description or if the question does not specify exact column names.
2. Incorrect Filtering Conditions: The prediction query incorrectly filters data based on the given conditions, ensuring the WHERE clause is used appropriately to match expected results.
3. Missing Handling of Nullable Column: Check if the query correctly handles nullable columns in aggregation functions (SUM, COUNT, AVG) or other operations, as improper handling can lead to unexpected results. Do not consider NULL values in arithmetic operations.
4. Missing Handling of Multiple Rows: Determine if the query correctly accounts for scenarios where multiple rows might satisfy the condition (e.g min/max, multiple transactions in a day), potentially leading to incorrect answers. Consider primary and foreign keys, as well as unique constraints, which ensure uniqueness.
5. Abused Clauses: Evaluate if SQL clauses like GROUP BY, HAVING, ORDER BY, and DISTINCT are used unnecessarily, which could produce incorrect results.
6. Other Fatal Logical Issues: Identify any additional logical problems not covered by the above criteria.


### Output Format
NOTE: If no issue is found, the explanation can be an empty string. Output only the JSON object containing your evaluation results. Make sure the JSON is properly formatted and valid, with all boolean values in lowercase (true/false) and all strings properly enclosed in double quotes.

```json
{
    "incorrect_schema_linking": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "incorrect_filtering_condition": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "missing_handling_of_nullable_column": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "missing_handling_of_multiple_rows": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "abused_clauses": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "other_fatal_issues": {
        "issued": true/false,
        "explanation": "Your explanation here"
    }
}
```
~~~

Figure 12: **FLEX** system prompt to categorize the error case of false positives.

~~~
You are an expert SQL evaluator responsible for assessing the correctness and quality of SQL queries.
Analyze the judgment thoroughly and provide a categorized evaluation based on the following criteria:
Two queries have no compile-time errors, which means they are both valid SQL queries, integer division is not automatically converted to float division, and logical operators are case-insensitive.

### Criteria
1. Different Output Structure: Column selection, ordering is different, but should not consider output column naming.
2. Different Output Value Representation: Differences in value representation, such as formatting or data types, should not be fatal unless they affect the logical correctness of the query.
3. Incorrect Ground Truth Query: The ground truth query is incorrect and does not logically answer the question.
4. Multiple Answers Available: The question is ambiguous and has multiple correct answers.
5. Other minor issues: Other minor issues that do not affect the logical correctness of the query.

### Output Format
NOTE: If no issue is found, the explanation can be an empty string. Output only the JSON object containing your evaluation results. Make sure the JSON is properly formatted and valid, with all boolean values in lowercase (true/false) and all strings properly enclosed in double quotes.

```json
{
    "different_output_structure": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "different_output_value_representation": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "incorrect_ground_truth_query": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "multiple_answers_available": {
        "issued": true/false,
        "explanation": "Your explanation here"
    },
    "other_minor_issues": {
        "issued": true/false,
        "explanation": "Your explanation here"
    }
```
~~~

Figure 13: **FLEX** system prompt to categorize the error case of false negatives.

```
**Schema**
{schema}

**Question**
{question}

**Prediction Query**
{pred_query}

**Prediction Result**
{pred_result}

**Ground Truth Query**
{gt_query}

**Ground Truth Result**
{gt_result}

**Hint**
{evidence}
```

Figure 14: **FLEX** user prompt. Execution results are excluded for generated queries that do not produce the same results as ground truth. Hint indicates the external knowledge (evidence) that provided in BIRD.

```
Criteria: How accurately does the Prediction Query answer the Question and adhere
to SQLite3 compatibility?
Score 1: The Prediction Query is entirely incorrect, failing to answer the
Question logically or containing critical errors that render it unusable. It may
also be incompatible with SQLite3.
Score 2: The Prediction Query has significant flaws, such as missing essential
tables/columns, incorrect filtering, or JOIN redundancy. It may partially answer
the Question but has major issues. SQLite3 compatibility might be questionable.
Score 3: The Prediction Query somewhat answers the Question but has noticeable
issues. It may fail to handle null values properly, misuse aggregation functions,
or not consider multiple row scenarios. It is mostly SQLite3 compatible but may
have minor issues.
Score 4: The Prediction Query largely answers the Question correctly with only
minor flaws. It handles most scenarios well, including null values and multiple
rows. It is fully SQLite3 compatible. Small oversights like missing DISTINCT in
SELECT are acceptable.
Score 5: The Prediction Query perfectly answers the Question, considering all
aspects including proper handling of null values, multiple rows, and edge cases.
It is fully SQLite3 compatible and optimized. Minor differences like missing
DISTINCT in SELECT are allowed if they don't affect the result.
```

Figure 15: Criteria and rubrics utilized in *Prometheus-2* for the equivalent set.

```
Criteria: How accurately does the Prediction Query answer the Question compared to the Ground Truth Query, considering logical correctness and result similarity?
Score 1: The Prediction Query completely fails to answer the Question logically. It contains significant errors, produces drastically different results from the Ground Truth Query, or fails to handle critical aspects like null values or multiple rows.
Score 2: The Prediction Query partially answers the Question but has major flaws. It may have incorrect filtering, missing conditions, or JOIN redundancy that significantly affect the results. The output differs notably from the Ground Truth Query in ways that impact the answer's accuracy.
Score 3: The Prediction Query generally answers the Question, but with some inconsistencies. It may have minor issues in handling null values or multiple rows. The results differ from the Ground Truth Query in ways that slightly affect the answer's completeness or accuracy.
Score 4: The Prediction Query correctly answers the Question with only minor differences from the Ground Truth Query. It handles null values and multiple rows appropriately. Any differences in output structure or value representation do not significantly impact the answer's accuracy.
Score 5: The Prediction Query perfectly answers the Question, either matching the Ground Truth Query or providing an equally valid alternative. It handles all aspects correctly, including null values and multiple rows. Any differences in output structure, column naming, or value representation are logically justified and do not affect the answer's accuracy
```

Figure 16: Criteria and rubrics utilized in *Prometheus-2* for the not equivalent set.